COMP [56]630– Machine Learning

Lecture 2 –ML Basics

Machine Learning Vocabulary

- **Example:** an object or instance in data used.
- **Features:** the set of attributes, often represented as a vector, associated to an example
- **Labels:**
  - in *classification*, category associated to an object
  - in *regression*, real-valued numbers.

Machine Learning Vocab (contd.)

- **Training data**: data used for training the ML algorithm.
- **Test data**: data exclusively used for testing the ML algorithm.
- Some standard learning scenarios:
  - **supervised learning**: labeled training data.
  - **unsupervised learning**: no *labeled* data.
  - **semi-supervised learning**: both labeled and unlabeled training data

Supervised Learning

- Inputs:
  - Series of data examples: x_1, x_2, x_3,…, x_n and corresponding labels y_1, y_2, y_3,…, y_n
- Goal:
  - Learn to associate patterns from the examples (x_1, x_2, x_3,…, x_n) with their labels (y_1, y_2, y_3,…, y_n)
  - Learn to produce the **desired output** (y) given a **new input** that may or may not be from the training examples
- Labels can be categorical (classification) or continuous (regression)
- Example: SPAM vs NON-SPAM classification, Predicting house prices from details like # of bedrooms, # of bathrooms, size, location, etc.

Unsupervised Learning

- Inputs:
  - Series of data examples: x_1, x_2, x_3,…, x_n
- Goal:
  - Build a model of x that can be used for reasoning, decision making, predicting things, communicating
  - Note that we do not use any labels (y_i) that may or may not be present
- Example: grouping data examples x_i based on features (clustering), learn good, common representations of x_i for other purposes (representation learning)

ML Basics – Linear Algebra

What is linear algebra?

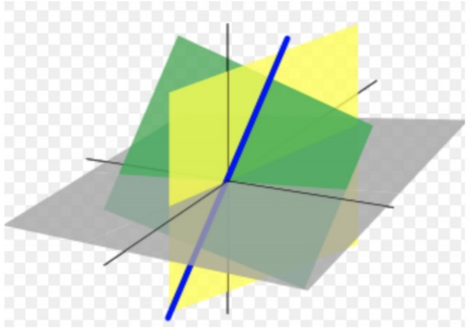
- Branch of mathematics that deal with linear equations such as

$$a_1x_1 + ...+a_nx_n = b$$

- Vector notation: $\mathbf{a}^T\mathbf{x} = b$

  → Called a *linear transformation of the variable x*

- Linear algebra is fundamental to geometry, for defining objects such as lines, planes, rotations



Linear equation $a_1x_1+.....+a_nx_n=b$
defines a plane in $(x_1,..,x_n)$ space
Straight lines define common solutions to equations

Why linear algebra?

- It is based on continuous values.
  - Used throughout engineering for various applications
- Essential for understanding ML algorithms
  - E.g., We convert input vectors (x_1,..,x_n) into outputs by a series of linear transformations
- In this lecture, we cover enough of the basics to understand ML algorithms.

Linear Algebra Topics

- Scalars, Vectors, Matrices and Tensors
- Multiplying Matrices and Vectors
- Identity and Inverse Matrices
- Linear Dependence and Span
- Norms
- Special kinds of matrices and vectors
- Eigen decomposition
- Singular value decomposition
- The Moore Penrose pseudoinverse
- Trace and determinants of a matrix

Scalars, Vectors and Matrix

- Scalar
  - Single number
  - Represented in lower-case italic x
  - They can be real-valued or be integers
    - i.e. slope of a line (real-valued)
- Vector
  - An array of numbers arranged in order
  - Each number can be identified by an index
  - Written in lower-case bold such as $\boldsymbol{x}$
  - We can think of vectors as points in space
    - Each element gives coordinate along an axis

Scalars, Vectors and Matrix

- A vector's elements are in italics lower case, subscripted

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \vdots \\ \boldsymbol{x_n} \end{bmatrix}$$

  - → *n* element vector
- Matrix: 2-D array of numbers
  - So each element identified by two indices
  - Denoted by bold typeface **A**
  - If A has shape of height m and width n with real-values then $\mathbf{A} \in \mathbb{R}^{m \times n}$

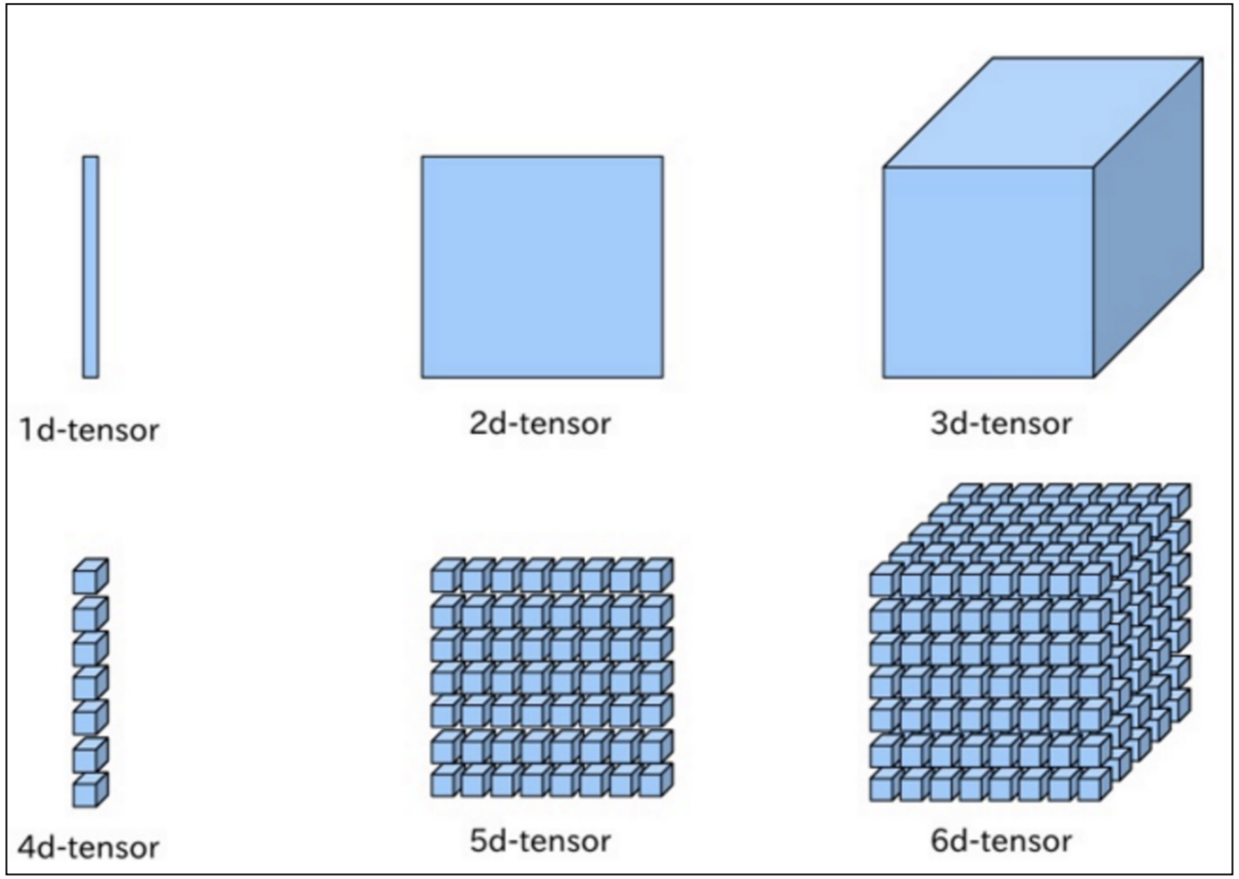
Matrix, Tensors

- Matrix:
  - Elements indicated by name in italic but not bold
  - $\mathbf{A}_{ij}$ represents the element in the i^{th} row and j^{th} column

$$\textbf{Example: } \mathbf{A} = \begin{bmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} \end{bmatrix}$$

- Tensors:
  - Arrays with **more than 2 dimensions**
    - Why?
  - A tensor is an array of numbers arranged on a regular grid with variable number of axes
  - Again, denoted by bold typeface **A**. Elements given by $\mathbf{A}_{ijk}$ for 3-d tensor.

Shapes of tensors



Matrix/Tensor operations

Transpose

- Denoted as $\mathbf{A}^{\mathrm{T}}$
- Defined as

$$(\mathbf{A}^{\mathrm{T}})_{\mathbf{i,j}} = \mathbf{A}_{\mathbf{j,I}}$$

- Mirror image across the (main) diagonal of a matrix
  - Main diagonal -> running down from upper left to the bottom right

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \\ A_{1,3} & A_{2,3} & A_{3,3} \end{bmatrix}$$

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

Vectors/Scalars as matrices

- Vectors → Matrices with one column
- Written as

$$x = [x_1 \quad \dots \quad x_n]^T$$

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \vdots \\ \boldsymbol{x_n} \end{bmatrix} \Rightarrow \boldsymbol{x^T} = [x_1 \quad \dots \quad x_n]^T$$

- Scalar → Matrix with one element

$$a = a^T$$

Matrix Addition

- We can add matrices to each other if they have the same shape, by adding corresponding elements
  - If A and B have same shape (height m, width n)

$$\mathbf{C} = \mathbf{A} + \mathbf{B}$$

$$\mathbf{C}_{i,j} = \mathbf{A}_{i,j} + \mathbf{B}_{i,j}$$

- You can add or multiply a matrix by a scalar

$$\mathbf{D} = a\mathbf{B} + c$$

$$\mathbf{D}_{i,j} = a\mathbf{B}_{i,j} + c$$

- Addition vector to matrix → i.e. **broadcasting** since vector added to each row of **A**

$$\mathbf{C} = \mathbf{A} + b$$

$$\mathbf{C}_{i,j} = \mathbf{A}_{i,j} + b_j$$

Matrix Multiplication

- For product **C = AB** to be defined, **A** has to have the same no. of columns as the no. of rows of **B**
  - If **A** is of shape *mxn* and **B** is of shape *nxp* then matrix product **C** is of shape *mxp*

$$C = AB \Rightarrow C_{i,j} = \sum_k A_{i,k} B_{k,j}$$

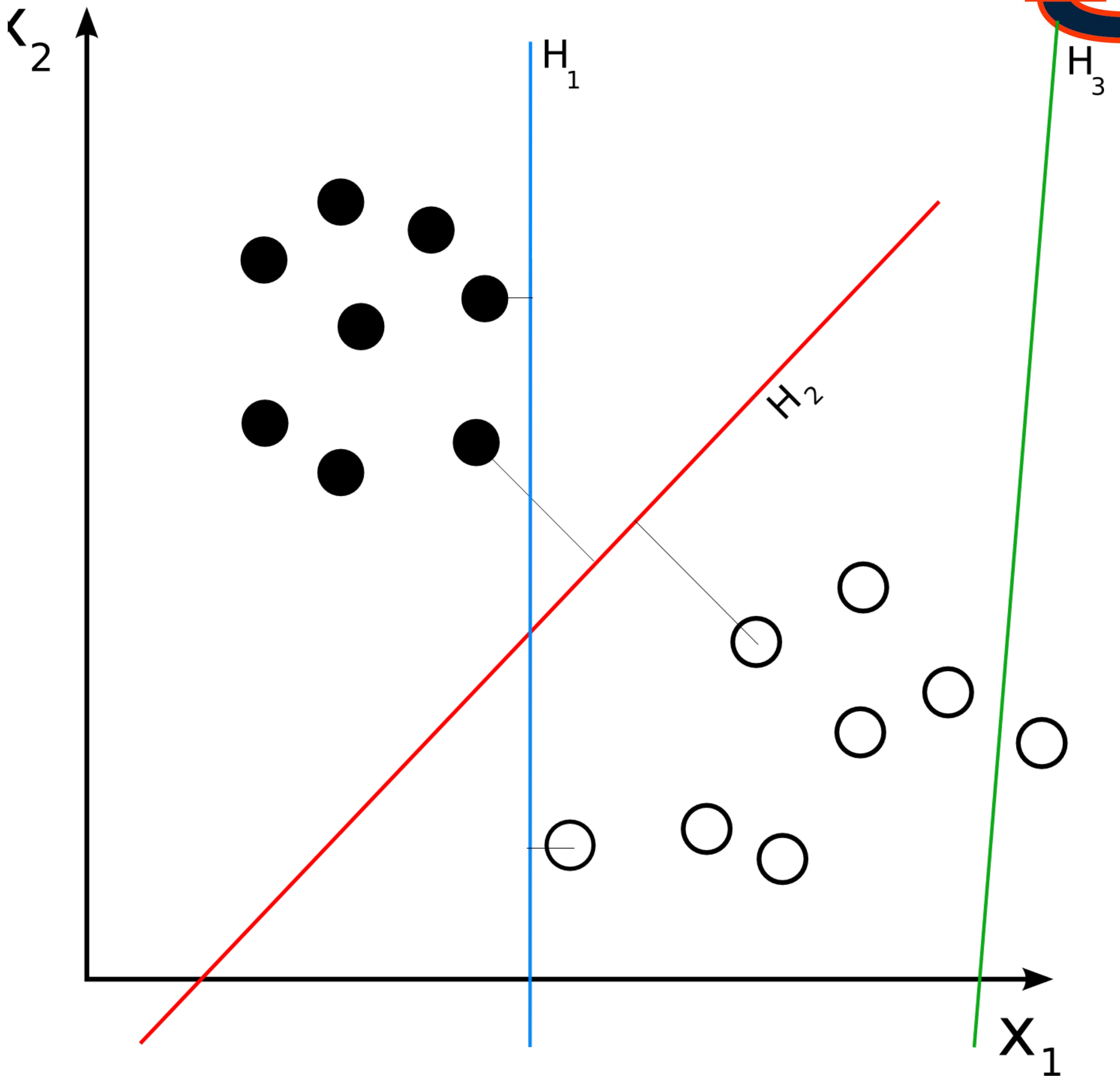
- **Product of two matrices is not the product of their individual elements!**
  - It is called element-wise product or the **Hadamard product** $\boldsymbol{A \odot B}$
  - We can think of matrix product **C** = **AB** as computing $C_{i,j}$ the dot product of row *i* of **A** and column *j* of **B**

Matrix Product Properties

- *Distributivity over addition:* **A(B+C)=AB+AC**
- *Associativity:* **A(BC)=(AB)C**
- *NOT commutative:* **AB=BA** is not always true
- *Dot product between vectors is commutative:* $x^Ty=y^Tx$
- *Transpose of a matrix product has a simple form:* $\mathbf{(AB)^T=B^TA^T}$

Linear Classifier

- The simplest ML model
- Makes a *classification* decision based on the value of ***a linear combination*** of the characteristics (features).
- Black and white circles are different labels. H_1, H_2, … represent different *decision boundaries* i.e. linear functions that best map the classification process.
    - ***Goal:*** find the best linear function that has highest accuracy



Linear Classifier (cntd.)

- Mathematically represented as

$$y = \boldsymbol{W}\boldsymbol{x}^{\boldsymbol{T}} + b$$

where y → labels (vector)

W → model parameter matrix

x → feature vector

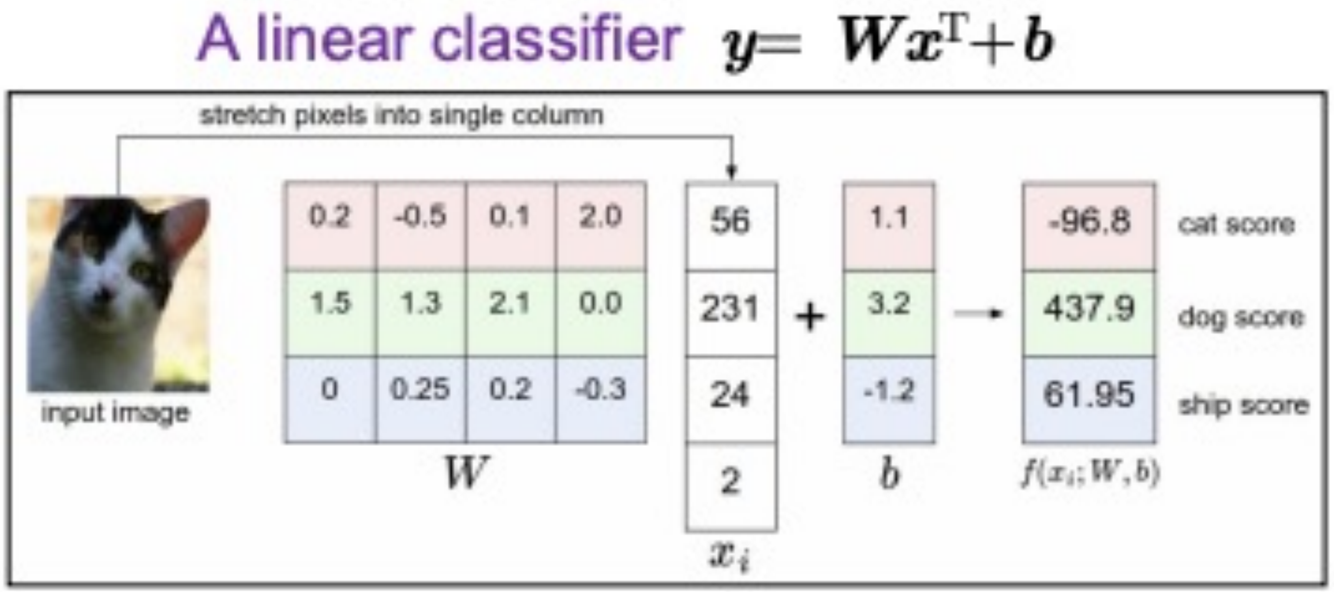
b → bias term (scalar)

- Very similar in the mathematical representation of a line

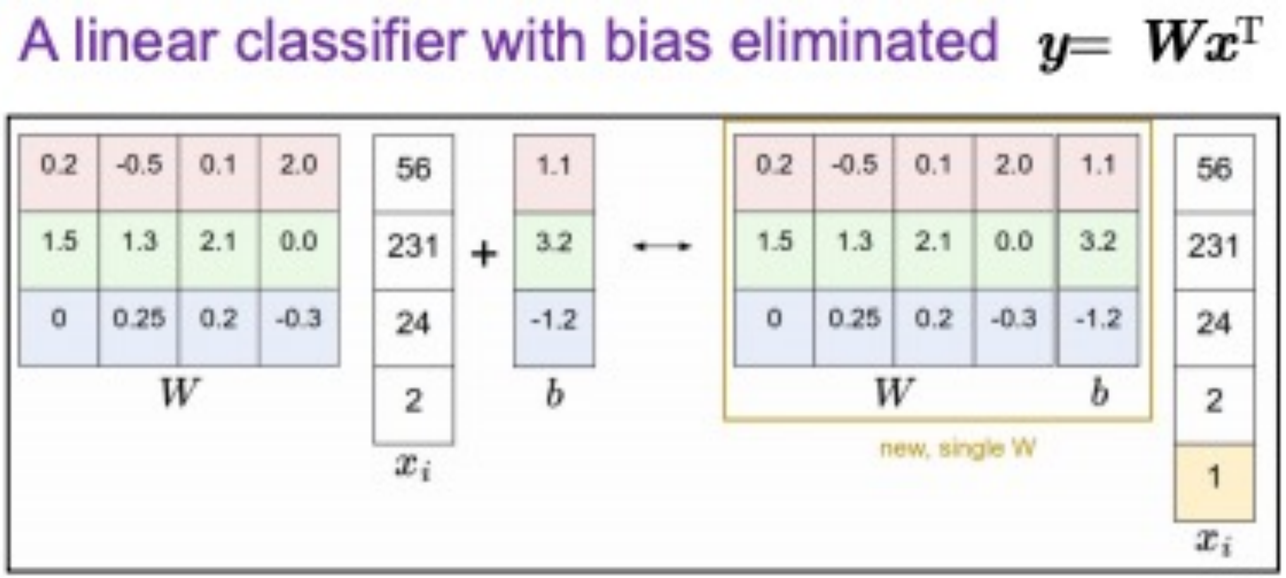
$$y = mx + c$$

→Hence the term ***linear classifier***

Linear Classifier (cntd.)



Linear Classifier (cntd.)



Linear Transformation

$$\boldsymbol{Ax}=\boldsymbol{b}$$

- where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^{n}$
- More explicitly

$$\boxed{\begin{aligned} A_{11}x_1 + A_{12}x_2 + \dots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \dots + A_{2n}x_n &= b_2 \\ \\ A_{n1}x_1 + A_{m2}x_2 + \dots + A_{n,n}x_n &= b_n \end{aligned}}$$

n equations in
n unknowns

Linear Transformation

$\boldsymbol{Ax}=\boldsymbol{b}$

- where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^{n}$
- More explicitly

$$\boxed{\begin{aligned} A_{11}x_1 + A_{12}x_2 + \dots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \dots + A_{2n}x_n &= b_2 \\ \\ A_{n1}x_1 + A_{m2}x_2 + \dots + A_{n,n}x_n &= b_n \end{aligned}}$$

System of equations

n equations in
n unknowns

Linear Transformation

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

- where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^{n}$
- More explicitly

$$\boxed{\begin{aligned} A_{11}x_1 + A_{12}x_2 + \dots + A_{1n}x_n &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \dots + A_{2n}x_n &= b_2 \\ \\ A_{n1}x_1 + A_{m2}x_2 + \dots + A_{n,n}x_n &= b_n \end{aligned}}$$

System of equations

n equations in
n unknowns

$$A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \vdots & \vdots \\ A_{n,1} & \cdots & A_{nn} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

n x n $\quad$ n x 1 $\quad$ n x 1

Can view A as a *linear transformation* of vector $\boldsymbol{x}$ to vector $\boldsymbol{b}$

Linear Transformation

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

- where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^{n}$
- More explicitly

$$
\begin{aligned}
A_{11}x_1 + A_{12}x_2 + \ldots + A_{1n}x_n &= b_1 \\
A_{21}x_1 + A_{22}x_2 + \ldots + A_{2n}x_n &= b_2 \\
A_{n1}x_1 + A_{m2}x_2 + \ldots + A_{n,n}x_n &= b_n
\end{aligned}
$$

System of equations

n equations in
n unknowns

$$
A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & \vdots & \vdots \\ A_{n,1} & \cdots & A_{nn} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}
$$

n x n $\quad$ n x 1 $\quad$ n x 1

Can view A as a *linear transformation* of vector $\boldsymbol{x}$ to vector $\boldsymbol{b}$

How to solve this?

Linear Transformation (cntd.)

- Matrix Inverse to the rescue!
  - Inverse of a matrix is defined as $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n$
    - $\mathbf{I}_n$ is an identity matrix of dimension n x n
    - $\mathbf{A}$ is a square matrix
- Solving $\mathbf{A}\boldsymbol{x}=\boldsymbol{b}$:

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\mathbf{A}^{-1}\mathbf{A}\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b}$$

$$\mathbf{I}_n\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b}$$

$$\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b}$$

Linear Transformation (cntd.)

- Matrix Inverse to the rescue!
  - Inverse of a matrix is defined as $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n$
    - $\mathbf{I}_n$ is an identity matrix of dimension n x n
    - **A** is a square matrix
- Solving $\mathbf{A}\boldsymbol{x}=\boldsymbol{b}$:

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\mathbf{A}^{-1}\mathbf{A}\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b}$$

$$\mathbf{I}_n\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b}$$

$$\boldsymbol{x} = \mathbf{A}^{-1}\boldsymbol{b}$$

Will this work for all cases?

Linear Transformation (cntd.)

- This depends on being able to find $\mathbf{A}^{-1}$
- If $\mathbf{A}^{-1}$ exists there are several methods for finding it
- **$\mathbf{A}^{-1}$ does not exist for ALL matrices and is possible to find only for square matrices and non-singular matrices**
- **Alternative:** Use *Gaussian elimination* and back-substitution.
  - Transform the matrix **A** into an upper triangular matrix using a series of row-wise operations such as
    - Swapping two rows,
    - Multiplying a row by a nonzero number,
    - Adding a multiple of one row to another row.

Linear Transformation (cntd.)

- Gaussian elimination example:
- Given a system of equations:

$$\begin{aligned} 2x + y - z &= 8 \\ -3x - y + 2z &= -11 \\ -2x + y + 2z &= -3 \end{aligned}$$

- Construct a matrix:

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right]$$

- Perform operations until you transform into upper triangular matrix

$$\begin{aligned} L_2 + \tfrac{3}{2}L_1 &\to L_2 \\ L_3 + L_1 &\to L_3 \end{aligned}$$

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 2 & 1 & 5 \end{array}\right]$$

$$L_3 + -4L_2 \to L_3$$

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$$

Linear Transformation (cntd.)

- Using the final matrix from the previous steps, we can see that the value of z = -1 (last row)
- Using back substitution, we get y = 3 (second row)

  and x = 2 (first row)

$$\left[\begin{array}{ccc|c} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{array}\right]$$

Disadvantages of Gaussian Elimination and Matrix Inverse

- Matrix Inverse:
  - Can only be used if $\mathbf{A}^{-1}$ exists
  - If $\mathbf{A}^{-1}$ exists, the same $\mathbf{A}^{-1}$ can be used for any given $\boldsymbol{b}$
  - But $\mathbf{A}^{-1}$ cannot be represented with sufficient precision
    - It is not used in practice
- Gaussian Elimination:
  - numerical instability (i.e. division by small no.)
  - Complexity if $O(n^3)$ for n x n matrix
- Software solutions use value of $\boldsymbol{b}$ in finding $\boldsymbol{x}$:
  - difference (derivative) between $\boldsymbol{b}$ and prediction is used iteratively
    - ***Least squares solvers***

Solving Ax = b

- Remember Ax = b is a system of equations
- Solution is x = A-1b

  => Ax =b can be solved if A^{-1} exists

  => A^{-1} exists only if exactly one solution exists for each value of b

  - Not always true!
- A system of equations can have no or infinite solutions for some values of b
  - Note: It is not possible to have more than one but fewer than infinitely many solutions
- **Ax = b** is a linear transformation i.e. it is a linear combination of those factors and hence
  - A column of A, i.e., $A_{:i}$ specifies travel in direction *i*
  - How much we need to travel is given by x_i
  - Thus determining whether Ax=b has a solution is equivalent to determining whether b is in the span of columns of A
    - Span of a set of vectors: set of points obtained by a linear combination of those vectors

Norms

- Measures size of a vector $\boldsymbol{x}$
  - i.e. distance from origin to $\boldsymbol{x}$
- Helps maps vectors to non-negative scalar values
- Norm of a vector $\boldsymbol{x} = [\boldsymbol{x_1} \quad \cdots \quad \boldsymbol{x_n}]^T$ is any function that satisfies the triangle inequality

$$f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = 0$$

$$f(\boldsymbol{x}+\boldsymbol{y}) \le f(\boldsymbol{x}) + f(\boldsymbol{y}) \qquad \text{Triangle Inequality}$$

$$\forall \alpha \in R \quad f(\alpha\boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$$

Norms (contd.)

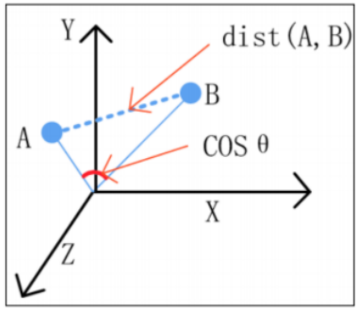
- Different kinds of norms exists and can generally be defined as the L^P norm and is given by

$$\|x\|_p = (\textstyle\sum_i |x_i|^p)^{\frac{1}{p}}$$

- If p=2, then it is called the L-2 norm or Euclidean norm
  - Euclidean distance between origin and x and is represented as $\|x\| = x^T x$
- If p=1, then it is called the L-1 norm.
  - Used when you need to distinguish zero and non-zero vectors
- If $p = \infty$, it is given by $L^\infty = \|x\|_\infty = \max |x_i|$
  - Called the max norm

“Special” Vectors

- Unit vector:
  - A vector with unit norm: $L^2(x) = 1$
- Orthogonal vectors:
  - Vectors x and y are orthogonal if $x^T y = 0$
    - i.e. if the vectors have non-zero norm, they are at 90 degrees to each other
- Orthonormal vector:
  - Vectors are orthogonal and have unit norm
- Dot product of two vectors: $\boxed{\boldsymbol{x}^T \boldsymbol{y} \Rightarrow \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 \cos\theta}$



Distance between two vectors $(\boldsymbol{v}, \boldsymbol{w})$

– $\mathrm{dist}(\boldsymbol{v},\boldsymbol{w}) = \|\boldsymbol{v}-\boldsymbol{w}\|$

$$= \sqrt{(v_1 - w_1)^2 + .. + (v_n - w_n)^2}$$

"Special" Matrices

- **Diagonal Matrix**: mostly zeros, with nonzero entries only in diagonal
  - Eg. Identity matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  - ***diag(v)*** denotes a square diagonal matrix with diagonal elements given by entries of vector ***v***
- **Symmetric matrix:** any matrix A which satisfies $\mathbf{A=A^T}$
- **Singular matrix:** A square matrix that does not have a matrix inverse. i.e. a matrix is singular iff its determinant is 0.
  - Determinant of a matrix: a scalar value that can be computed from the elements of a square matrix and encodes certain properties of the linear transformation described by the matrix

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Matrix Decomposition

- Matrices can be decomposed into factors to learn universal properties
- Many popular algorithms leverage matrix decomposition for solving tasks ranging from data cleaning to label prediction
  - Common applications include
    - Dimensionality reduction
    - Preventing overfitting
    - Finding better features (ignoring clutter, background noise, etc.) by focusing on important aspects of the input
- Many possible ways to matrix decomposition
  - Eigen decomposition
  - QR decomposition
  - Single Value Decomposition

Eigen Decomposition

- We can decompose a matrix **A** as **A=V** *diag*(λ)**V**$^{-1}$
  - Where **V** → eigenvectors, λ → eigenvalues

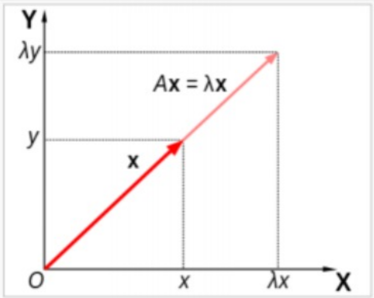
$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ has eigenvalues $\lambda = 1$ and $\lambda = 3$ and eigenvectors V: $v_{\lambda=1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, v_{\lambda=3} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

- An eigenvector of a square matrix **A** is a non-zero vector ***v*** such that multiplication by **A** only changes the scale of ***v***

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

Where **λ** is called the eigenvalue

- If ***v*** is an eigenvector of **A**, so is any rescaled vector ***sv***.
  - Note: ***sv*** still has the same eigen value



Matrix *A* acts by stretching the vector *x*, not changing its direction, so *x* is an eigenvector of *A*.

Wikipedia

What does Eigen Decomp. tell us?

- Provides insights about the matrix:
  - Singular matrix: A matrix is said to be singular ***if & only if*** any eigenvalue is zero
  - Useful to optimize quadratic expressions of form

$$f(x) = x^T A x \text{ such that } \|x\|_2 = 1$$

- Whenever x is equal to an eigenvector, f is equal to the corresponding eigenvalue
- Maximum value of f is max eigen value, minimum value is min eigen value
- This property is very useful in solving several algorithm formulations such as modeling a multivariate Gaussian

$$\boxed{\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right\}}$$

Things to remember

- A matrix whose eigenvalues are all positive is called *positive definite*
  - Positive or zero is called *positive semidefinite*
  - Why is this important?
    - Positive definite matrices guarantee that $x^TAx \geq 0$
- If eigen values are all negative it is negative definite

Single Value Decomposition

- Eigen decomposition of A is of the form
$$A = V\, diag(\lambda) V^{-1}$$
  - If **A** is not square, you cannot do Eigen decomposition
- SVD can help solve this issue.
  - It is of the form $A = UDV^T$
- It is more general than Eigen decomposition
  - Can be used for any matrix
    - Eigen is restricted to symmetric, square matrices
  - All real matrices can be factorized using SVD

SVD (contd.)

- It is of the form $A = UDV^T$
- U and V are orthogonal matrices
- D is a diagonal matrix
  - Not necessarily square
    - Elements of Diagonal of D are called singular values of A
    - Columns of U are called left singular vectors
    - Columns of V are called right singular vectors
- SVD can be represented in terms of Eigen decomposition:
  - Left singular vectors of **A** are eigenvectors of $\mathbf{AA^T}$
  - Right singular vectors of **A** are eigenvectors of $\mathbf{A^TA}$
  - Nonzero singular values of **A** are square roots of eigen values of $\mathbf{A^TA}$. Same is true of $\mathbf{AA^T}$

Moore-Penrose pseudoinverse

- Most useful feature of SVD is that it can be used to generalize matrix inversion to non-square matrices
- **pseudoinverse** of a matrix generalizes the notion of an inverse
  - Not every matrix has an inverse, but every matrix has a pseudoinverse, even non-square matrices.
- Practical algorithms for computing the pseudoinverse of A are based on SVD

$$\mathbf{A^{+}=VD^{+}U^{T}}$$

  - where U,D,V are the SVD of A, ‘+’ refers to the pseudoinverse
    - Pseudoinverse (D^{+}) of D is obtained by taking the reciprocal of its nonzero elements when taking transpose of resulting matrix

If you did not understand it in detail

If you did not understand it in detail

How to do all this in Python?

NumPy

- NumPy is a Python library
- Supports large, multi-dimensional arrays and matrices
- Provides a large collection of high-level mathematical functions to operate on these arrays.
- Runs on CPU
- Highly optimized by computer scientists and mathematicians