



AI6103 Linear Algebra

Boyang Li, Albert

School of Computer Science and Engineering
Nanyang Technological University

Math Subjects Relevant to Machine Learning

- Calculus, Multivariate Calculus
 - Linear Algebra
 - Probability Theory
 - Information Theory
 - Convex Optimization
 - Differential Equations
 - Random Matrices
 - And so on
- } Reviewed in this course

You don't have to know everything to do machine learning.

But at least you need the first three.



Online Resources

- Slow and accessible video lectures
<https://www.youtube.com/playlist?list=PLHXZ9OQGMqxfUI0tcqPNTJsb7R6BqSLo6>
- Geometric intuition (highly recommended)
https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab
- Interactive textbook
 - Dan Margalit and Joseph Rabinoff. Interactive Linear Algebra.
<https://textbooks.math.gatech.edu/ila/index.html>
- Comprehensive reference manual:
 - X.-D. Zhang, A Matrix Algebra Approach to Artificial Intelligence. Springer. 2020 (Available as e-book in NTU library)

1. Vectors

Math is about connecting simple steps to create something not obvious at all.

Don't overlook the simple stuff!



A vector is a sequence of scalars

- A scalar is a single number, e.g., 5, 1024, 231.2, $\frac{87}{99}$
- A vector is a sequence of scalars
- $\mathbf{a} = \begin{pmatrix} -1.4 \\ 0.07 \\ 3.6 \end{pmatrix}$ (column 3-vector)
- $\mathbf{b} = (1.1, 4.3, 9, 100)$ (row 4-vector)

$$a = \begin{pmatrix} -1.4 \\ 0.07 \\ 3.6 \end{pmatrix}$$

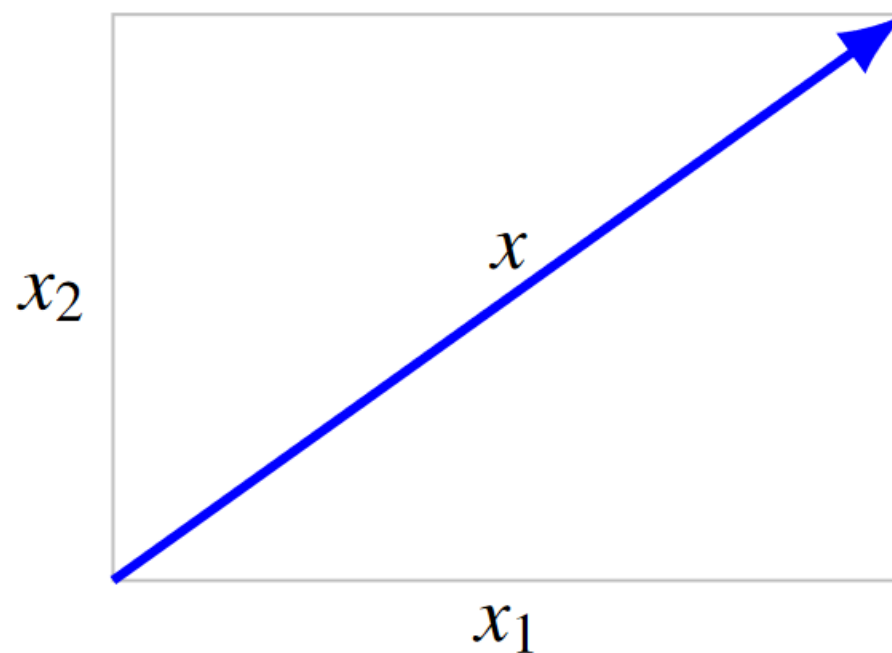
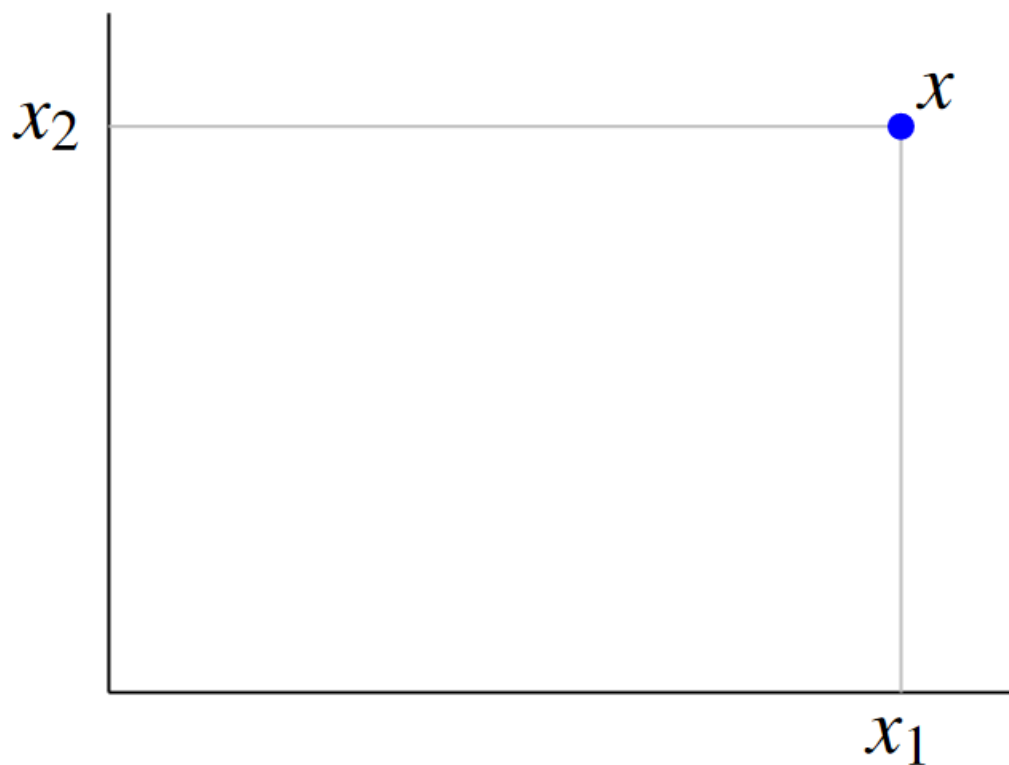
Notations

- ▶ we'll use symbols to denote vectors, e.g., a , X , p , β , E^{aut}
- ▶ other conventions: \mathbf{g} , \vec{a}
- ▶ i th element of n -vector a is denoted a_i
- ▶ if a is vector above, $a_3 = 3.6$
- ▶ in a_i , i is the *index*
- ▶ for an n -vector, indexes run from $i = 1$ to $i = n$
- ▶ *warning*: sometimes a_i refers to the i th vector in a list of vectors
- ▶ two vectors a and b of the same size are equal if $a_i = b_i$ for all i
- ▶ we overload $=$ and write this as $a = b$

In college-level math and beyond, there are often many different sets of notations. To each mathematician their own.



2-vector (x_1, x_2) can represent a location or a displacement in 2-D



Examples

- color: (R, G, B)
- quantities of n different commodities (or resources), e.g., bill of materials
- portfolio: entries give shares (or \$ value or fraction) held in each of n assets, with negative meaning short positions
- cash flow: x_i is payment in period i to us
- audio: x_i is the acoustic pressure at sample time i (sample times are spaced $1/44100$ seconds apart)
- features: x_i is the value of i th feature or attribute of an entity
- customer purchase: x_i is the total \$ purchase of product i by a customer over some period
- word count: x_i is the number of times word i appears in a document

- ▶ a short document:

Word count vectors are used **in** computer based **document** analysis. Each entry of the **word** count vector is the **number** of times the associated dictionary **word** appears **in** the **document**.

- ▶ a small dictionary (left) and word count vector (right)

word	3
in	2
number	1
horse	0
the	4
document	2

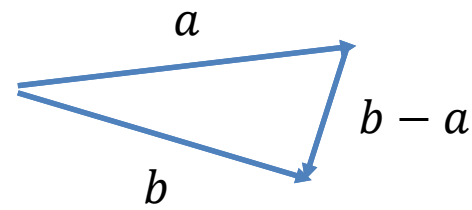
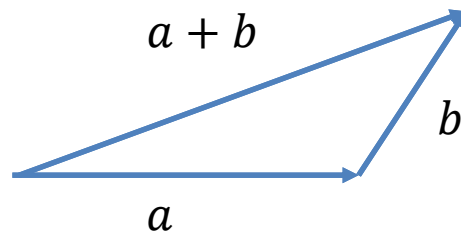
- ▶ dictionaries used in practice are much larger

Vector addition and subtraction

- ▶ n -vectors a and b can be added, with sum denoted $a + b$
- ▶ to get sum, add corresponding entries:

$$\begin{bmatrix} 0 \\ 7 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 9 \\ 3 \end{bmatrix}$$

- ▶ subtraction is similar



Scalar-Vector Multiplication

- ▶ scalar β and n -vector a can be multiplied

$$\beta a = (\beta a_1, \dots, \beta a_n)$$

- ▶ also denoted $a\beta$
- ▶ example:

$$(-2) \begin{bmatrix} 1 \\ 9 \\ 6 \end{bmatrix} = \begin{bmatrix} -2 \\ -18 \\ -12 \end{bmatrix}$$

Linear Combination

- ▶ for vectors a_1, \dots, a_m and scalars β_1, \dots, β_m ,

$$\beta_1 a_1 + \dots + \beta_m a_m$$

is a *linear combination* of the vectors

- ▶ β_1, \dots, β_m are the *coefficients*
- ▶ a *very* important concept
- ▶ a simple identity: for any n -vector b ,

$$b = b_1 e_1 + \dots + b_n e_n$$

$$e_1 = (1, 0, 0, \dots, 0), \quad e_2 = (0, 1, 0, \dots, 0), \quad \dots, \quad e_n = (0, 0, 0, \dots, 1)$$

Inner Product

- ▶ *inner product* (or *dot product*) of n -vectors a and b is

$$a^T b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

- ▶ other notation used: $\langle a, b \rangle$, $\langle a|b \rangle$, (a, b) , $a \cdot b$

- ▶ example:

$$\begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0 \\ -3 \end{bmatrix} = (-1)(1) + (2)(0) + (2)(-3) = -7$$

Properties of the Inner Product

- ▶ $a^T b = b^T a$ (commutative)
- ▶ $(\gamma a)^T b = \gamma(a^T b)$ (γ is a scalar)
- ▶ $(a + b)^T c = a^T c + b^T c$ (distributive)

can combine these to get, for example,

$$(a + b)^T (c + d) = a^T c + a^T d + b^T c + b^T d$$

(applying the distributive rule twice)

Some Uses of the Inner Product

- ▶ $e_i^T a = a_i$ (picks out i th entry)
- ▶ $\mathbf{1}^T a = a_1 + \cdots + a_n$ (sum of entries)
- ▶ $a^T a = a_1^2 + \cdots + a_n^2$ (sum of squares of entries)

Examples

- ▶ w is weight vector, f is feature vector; $w^T f$ is score
- ▶ p is vector of prices, q is vector of quantities; $p^T q$ is total cost
- ▶ c is cash flow, d is discount vector (with interest rate r):

$$d = (1, 1/(1 + r), \dots, 1/(1 + r)^{n-1})$$

$d^T c$ is net present value (NPV) of cash flow

- ▶ s gives portfolio holdings (in shares), p gives asset prices; $p^T s$ is total portfolio value

Linear Regression

- ▶ *regression model* is (the affine function of x)

$$\hat{y} = x^T \beta + v$$

- ▶ x is a feature vector; its elements x_i are called *regressors*
- ▶ n -vector β is the *weight vector*
- ▶ scalar v is the *offset*
- ▶ scalar \hat{y} is the *prediction*
(of some actual outcome or *dependent variable*, denoted y)

Linear regression is one of the simplest machine learning models. But it has much in common with more complex techniques.



Linear Regression: An example

- ▶ y is selling price of house in \$1000 (in some location, over some period)
- ▶ regressor is

$x = (\text{house area, \# bedrooms})$

(house area in 1000 sq.ft.)

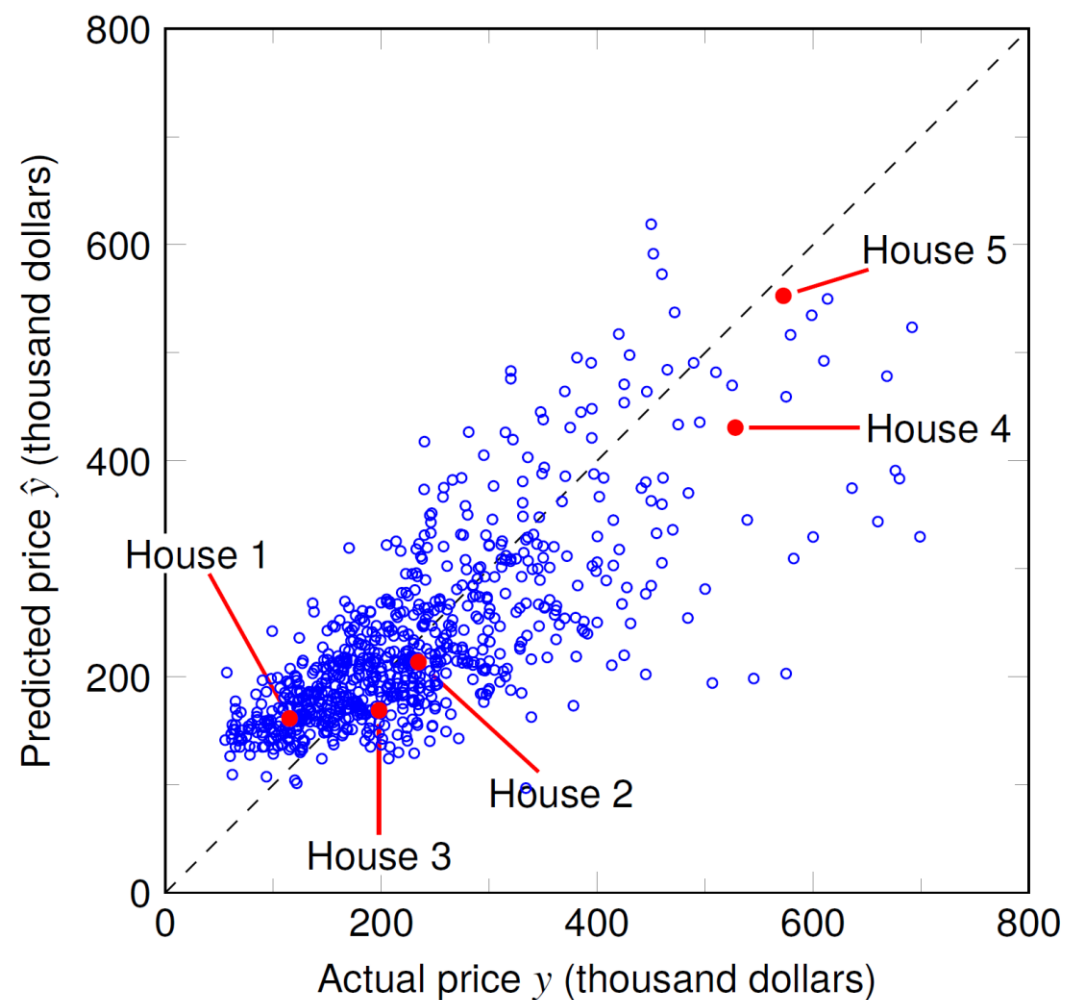
- ▶ regression model weight vector and offset are

$$\beta = (148.73, -18.85), \quad v = 54.40$$

- ▶ we'll see later how to guess β and v from sales data

House	x_1 (area)	x_2 (beds)	y (price)	\hat{y} (prediction)
1	0.846	1	115.00	161.37
2	1.324	2	234.50	213.61
3	1.150	3	198.00	168.88
4	3.037	4	528.00	430.67
5	3.984	5	572.50	552.66

Linear Regression: An example



House	x_1 (area)	x_2 (beds)	y (price)	\hat{y} (prediction)
1	0.846	1	115.00	161.37
2	1.324	2	234.50	213.61
3	1.150	3	198.00	168.88
4	3.037	4	528.00	430.67
5	3.984	5	572.50	552.66

Vector Norm

- ▶ the *Euclidean norm* (or just *norm*) of an n -vector x is

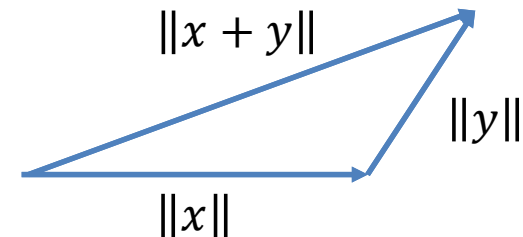
$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{x^T x}$$

- ▶ used to measure the size of a vector
- ▶ reduces to absolute value for $n = 1$
- ▶ (Euclidean) *distance* between n -vectors a and b is

$$\mathbf{dist}(a, b) = \|a - b\|$$

for any n -vectors x and y , and any scalar β

- ▶ *homogeneity*: $\|\beta x\| = |\beta| \|x\|$
- ▶ *triangle inequality*: $\|x + y\| \leq \|x\| + \|y\|$
- ▶ *nonnegativity*: $\|x\| \geq 0$
- ▶ *definiteness*: $\|x\| = 0$ only if $x = 0$



Inner Product and the Angle Between Two Vectors

- ▶ *angle* between two nonzero vectors a, b defined as

$$\angle(a, b) = \arccos \left(\frac{a^T b}{\|a\| \|b\|} \right)$$

- ▶ $\angle(a, b)$ is the number in $[0, \pi]$ that satisfies

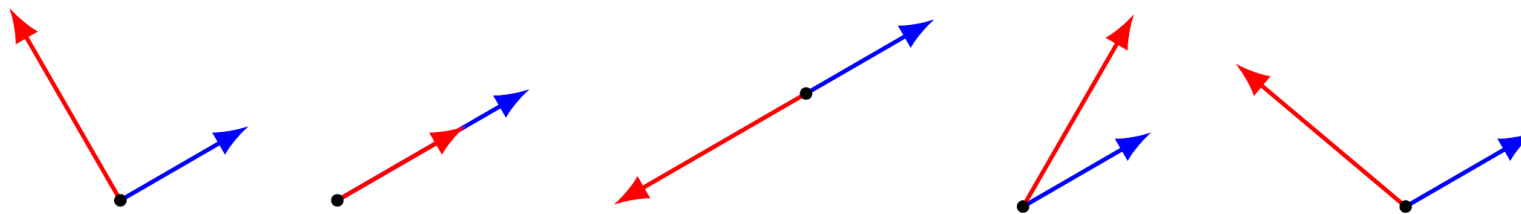
$$a^T b = \|a\| \|b\| \cos(\angle(a, b))$$

- ▶ coincides with ordinary angle between vectors in 2-D and 3-D

Inner Product and the Angle Between Two Vectors

$$\theta = \angle(a, b)$$

- ▶ $\theta = \pi/2 = 90^\circ$: a and b are *orthogonal*, written $a \perp b$ ($a^T b = 0$)
- ▶ $\theta = 0$: a and b are *aligned* ($a^T b = \|a\| \|b\|$)
- ▶ $\theta = \pi = 180^\circ$: a and b are *anti-aligned* ($a^T b = -\|a\| \|b\|$)
- ▶ $\theta \leq \pi/2 = 90^\circ$: a and b make an *acute angle* ($a^T b \geq 0$)
- ▶ $\theta \geq \pi/2 = 90^\circ$: a and b make an *obtuse angle* ($a^T b \leq 0$)



Document Distance

- Possible distance definitions:
 - The angle $\arccos(\frac{a^T b}{\|a\| \|b\|})$ on the word count vectors
 - The Euclidean distance of the word count vectors
- 5 Wikipedia articles: “Veterans Day”, “Memorial Day”, “Academy Awards”, “Golden Globe Awards”, “Super Bowl”.
- Word count vectors from 4423 selected words.

	Veterans Day	Memorial Day	Academy Awards	Golden Globe Awards	Super Bowl
Veterans Day	0	0.095	0.130	0.153	0.170
Memorial Day	0.095	0	0.122	0.147	0.164
Academy A.	0.130	0.122	0	0.108	0.164
Golden Globe A.	0.153	0.147	0.108	0	0.181
Super Bowl	0.170	0.164	0.164	0.181	0

Euclidean distance

	Veterans Day	Memorial Day	Academy Awards	Golden Globe Awards	Super Bowl
Veterans Day	0	60.6	85.7	87.0	87.7
Memorial Day	60.6	0	85.6	87.5	87.5
Academy A.	85.7	85.6	0	58.7	85.7
Golden Globe A.	87.0	87.5	58.7	0	86.0
Super Bowl	87.7	87.5	86.1	86.0	0

Cosine distance

2. Matrices

Matrix

- ▶ a *matrix* is a rectangular array of numbers, e.g.,

$$\begin{bmatrix} 0 & 1 & -2.3 & 0.1 \\ 1.3 & 4 & -0.1 & 0 \\ 4.1 & -1 & 0 & 1.7 \end{bmatrix}$$

- ▶ its *size* is given by (row dimension) \times (column dimension)
e.g., matrix above is 3×4
- ▶ *elements* also called *entries* or *coefficients*
- ▶ B_{ij} is i, j element of matrix B
- ▶ i is the *row index*, j is the *column index*; indexes start at 1
- ▶ two matrices are *equal* (denoted with $=$) if they are the same size and corresponding entries are equal

In programming terms, a matrix is a 2d array.

3d, 4d, ..., n-d arrays are called tensors in math.

In this class, we focus on real matrices.



Matrix Transpose

The transpose of A is denoted as A^T

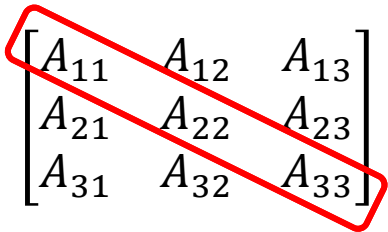
$$A_{ij} = A_{ji}^T$$

$$\text{Example: } A = \begin{bmatrix} 3 & -5 & 17 \\ 0 & 6 & 52 \\ 0 & 2 & 9 \end{bmatrix}, A^T = \begin{bmatrix} 3 & 0 & 0 \\ -5 & 6 & 2 \\ 17 & 52 & 9 \end{bmatrix}$$

We also have $(AB)^T = B^T A^T$

Matrix Notations

- A $m \times n$ matrix is
 - Tall if $m > n$
 - Wide if $m < n$
 - Square if $m = n$
- A (column) vector is a $n \times 1$ matrix
- A row vector is a $1 \times n$ matrix
- A square matrix where all non-zero values are on the diagonal is a diagonal matrix.
- A square matrix A
 - is diagonal if and only if $A_{ij} = 0$ for all $i \neq j$
 - is upper-triangular if and only if $A_{ij} = 0$ for all $i > j$
 - lower-triangular if and only if $A_{ij} = 0$ for all $i < j$


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

Diagonal

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 9 \end{bmatrix} \text{ A diagonal matrix}$$

$$\begin{bmatrix} 3 & -5 & 17 \\ 0 & 6 & 52 \\ 0 & 0 & 9 \end{bmatrix} \text{ An upper-triangular matrix}$$

$$\begin{bmatrix} 3 & 0 & 0 \\ 4 & 6 & 0 \\ 28 & 0 & 9 \end{bmatrix} \text{ A lower-triangular matrix}$$

Block Matrices

- ▶ we can form *block matrices*, whose entries are matrices, such as

$$A = \begin{bmatrix} B & C \\ D & E \end{bmatrix}$$

where B , C , D , and E are matrices (called *submatrices* or *blocks* of A)

- ▶ matrices in each block row must have same height (row dimension)
- ▶ matrices in each block column must have same width (column dimension)
- ▶ example: if

$$B = \begin{bmatrix} 0 & 2 & 3 \end{bmatrix}, \quad C = \begin{bmatrix} -1 \end{bmatrix}, \quad D = \begin{bmatrix} 2 & 2 & 1 \\ 1 & 3 & 5 \end{bmatrix}, \quad E = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

then

$$\begin{bmatrix} B & C \\ D & E \end{bmatrix} = \begin{bmatrix} 0 & 2 & 3 & -1 \\ 2 & 2 & 1 & 4 \\ 1 & 3 & 5 & 4 \end{bmatrix}$$

- ▶ A is an $m \times n$ matrix
- ▶ can express as block matrix with its (m -vector) columns a_1, \dots, a_n

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}$$

- ▶ or as block matrix with its (n -row-vector) rows b_1, \dots, b_m

$$A = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Examples

- ▶ *image*: X_{ij} is i, j pixel value in a monochrome image
- ▶ *rainfall data*: A_{ij} is rainfall at location i on day j
- ▶ *multiple asset returns*: R_{ij} is return of asset j in period i
- ▶ *contingency table*: A_{ij} is number of objects with first attribute i and second attribute j
- ▶ *feature matrix*: X_{ij} is value of feature i for entity j

Confusion Matrices

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Ground truths

airplane	828	13	12	11	18	0	2	4	85	27
automobile	10	910	0	5	1	1	0	1	11	61
bird	47	1	708	64	88	14	63	4	8	3
cat	3	4	16	768	33	93	50	19	4	10
deer	10	0	39	43	788	12	57	43	6	2
dog	2	0	10	137	29	777	8	33	0	4
frog	7	2	10	54	29	7	888	1	1	1
horse	24	2	14	39	76	17	4	818	2	4
ship	27	13	0	7	3	0	3	0	933	14
truck	19	64	1	7	2	1	1	0	18	887
	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck

Predicted values
(CIFAR-10 Image Classification)

Matrix-vector Multiplication

- $m \times n$ matrix A
- n -vector x
- The product is $y = Ax$

- $$\begin{bmatrix} 5 & -1 & 2 \\ 9 & 2 & -7 \end{bmatrix} \begin{bmatrix} 1 \\ -4 \\ 3 \end{bmatrix} = \begin{bmatrix} 15 \\ -20 \end{bmatrix}$$

- Row interpretation

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} x = \begin{bmatrix} a_1^\top x \\ a_2^\top x \end{bmatrix}$$

- Column interpretation

$$\begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_1 b_1 + x_2 b_2 + x_3 b_3$$

Matrix-vector Multiplication

- Identity matrix is a square diagonal matrix with 1s on the diagonal

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

- Sometimes we write I_n to indicate its dimension

- ▶ $0x = 0$, *i.e.*, multiplying by zero matrix gives zero
- ▶ $Ix = x$, *i.e.*, multiplying by identity matrix does nothing
- ▶ inner product $a^T b$ is matrix-vector product of $1 \times n$ matrix a^T and n -vector b
- ▶ $\tilde{x} = Ax$ is de-meaned version of x , with

$$A = \begin{bmatrix} 1 - 1/n & -1/n & \cdots & -1/n \\ -1/n & 1 - 1/n & \cdots & -1/n \\ \vdots & & \ddots & \vdots \\ -1/n & -1/n & \cdots & 1 - 1/n \end{bmatrix}$$

Feature Matrix – Weight Vector

- We have N data points, each having P features.
- $N \times P$ matrix X
- X_{ij} is the i -th data point's j -th feature
- The linear regression weight is a P -vector β
- The predictions of the linear regression model is $X\beta + b\mathbf{1}$
- $\mathbf{1}$ is an all-one vector and b is a scalar called the bias

Matrix-matrix Multiplication

- $m \times n$ matrix A , $n \times p$ matrix B

- $A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_m \end{bmatrix}$ (row vector form)

- $B = [b_1 \ b_2 \ b_3 \ \cdots \ b_p]$ (column vector form)

- AB is a $m \times p$ matrix

- $$AB = \begin{bmatrix} a_1 b_1 & a_1 b_2 & \cdots & a_1 b_p \\ a_2 b_1 & a_2 b_2 & \cdots & a_2 b_p \\ \vdots & \vdots & \cdots & \vdots \\ a_m b_1 & a_m b_2 & \cdots & a_m b_p \end{bmatrix}$$
$$= \left[\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_m \end{bmatrix} b_1 \quad \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_m \end{bmatrix} b_2 \quad \cdots \quad \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_m \end{bmatrix} b_p \right]$$
$$= \begin{bmatrix} a_1 [b_1 \ b_2 \ b_3 \ \cdots \ b_p] \\ a_2 [b_1 \ b_2 \ b_3 \ \cdots \ b_p] \\ \vdots \\ a_m [b_1 \ b_2 \ b_3 \ \cdots \ b_p] \end{bmatrix}$$

Linear Equations and Matrices

- A set of equations

$$\begin{cases} 3x + y - z = 5 \\ x + 8z = 12 \\ -5x - 4y + z = -20 \end{cases}$$

- Can be written in matrix form

$$\begin{bmatrix} 3 & 1 & -1 \\ 1 & 0 & 8 \\ -5 & -4 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 12 \\ -20 \end{bmatrix}$$

- In general, for known $A \in R^{m \times n}$, $b \in R^n$, $b \neq 0$, we seek $x \in R^m$ such that
$$Ax = b$$
- When does the set of equation have a unique solution?
 - Condition 1: We need n equations for n unknowns. That is, $m = n$
 - Condition 2: The n equations must be linearly independent
- When those hold, we have the unique solution

$$x = A^{-1}b$$

- That is, matrix A has an inverse.

Linear Independence

- A set vectors v_1, v_2, \dots, v_n are linearly independent if we cannot write one vector as a weighted sum of the others
- $v_i \neq \sum_{j \neq i} \alpha_j v_j$ no matter what values we assign to $\alpha_1, \dots, \alpha_n$

$$\begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix} \begin{bmatrix} 3 & 1 & -1 \\ 1 & 0 & 0 \\ -5 & -4 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \\ -6 \end{bmatrix}$$

These are not independent!

$$v_3 = 4v_1 - 17v_2$$

Matrix Inverse

- The **rank** of a matrix = the number of linearly independent rows
- A $n \times n$ matrix is full rank when it has n linearly independent rows.
- A matrix has an inverse (i.e., is invertible) if and only if it is full rank.

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

Matrix Inverse

- Recall that, in general, $AB \neq BA$
- However, the left inverse is the right inverse
- $AA^{-1} = I$ implies that $A^{-1}A = I$
- Not showing the proof here but it is available online

Matrix Inverse

- If A has an inverse, A^T also has an inverse.

$$(AA^{-1})^T = I^T = I$$

$$(A^{-1})^T A^T = I$$

- That is, if A has linearly independent rows, it must also have linearly independent columns.

Linear Regression: An example

► y is selling price of house in \$1000 (in some location, over some period)

► regressor is

$x = (\text{house area, \# bedrooms})$

(house area in 1000 sq.ft.)

► regression model weight vector and offset are

$\beta = (148.73, -18.85), \quad v = 54.40$

► we'll see later how to guess β and v from sales data

$$\begin{bmatrix} 0.846 & 1 \\ 1.324 & 2 \\ 1.150 & 3 \\ 3.037 & 4 \\ 3.984 & 5 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 155.00 \\ 243.50 \\ 198.00 \\ 528.00 \\ 527.50 \end{bmatrix}$$

House	x_1 (area)	x_2 (beds)	y (price)	\hat{y} (prediction)
1	0.846	1	115.00	161.37
2	1.324	2	234.50	213.61
3	1.150	3	198.00	168.88
4	3.037	4	528.00	430.67
5	3.984	5	572.50	552.66

No inverse for the 5×2 matrix. No exact solutions.

We can find the so-called “pseudo-inverse”



$$\begin{bmatrix} 0.846 & 1 \\ 1.324 & 2 \\ 1.150 & 3 \\ 3.037 & 4 \\ 3.984 & 5 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 155.00 \\ 243.50 \\ 198.00 \\ 528.00 \\ 527.50 \end{bmatrix}$$

Linear Regression

- In general, we have data matrix $X \in R^{n \times p}$ and label vector $Y \in R^{n \times 1}$
- We assume the relationship between X and Y is

$$X\beta \approx Y$$

- We want to find the best $\beta \in R^{p \times 1}$ that minimizes the error

$$\|X\beta - Y\|^2$$

- We can show (but will not) that the best β is

$$\beta^* = \underbrace{(X^T X)^{-1} X^T}_{\text{Pseudo-inverse}} Y$$

Python Programming

- Python is a dynamic-typed language.
 - No static type-checking
 - Flexible, succinct, though less efficient and error-prone
- Python has many numerical libraries
 - They call highly efficient C/C++ libraries behind the curtain
 - LAPACK, BLAS, etc.
- Great support for deep learning
 - PyTorch, Tensorflow, Jax, Peddle, etc.

Basics of a Programming Language

- Basic syntax
- Data types
- Control structure
- Exceptions
- Threads
- Performance

Basics of a Programming Language

- Basic syntax
- Data types
- ~~• Control structure~~
- ~~• Exceptions~~
- Threads
- Performance

Basic Syntax

- Indentation and colon mark code blocks
- Space-based indentations and tab-based are different!
- Can be super hard to debug!

```
sum = 0
for i in range(10):
    sum += i
print(sum)
```

sum is printed only once!

```
sum = 0
for i in range(10):
    sum += i + i**2 + \
        i**3
print(sum)
```

Basic Syntax

- The return character marks end of line
- To continue the same line of code, use the line continuation character '\'
- # is the beginning of a comment

```
sum = 0
for i in range(10):
    sum += i + i**2 #no continuation
    +i**3
print(sum) # 330
```

```
-----
sum = 0
for i in range(10):
    sum += i + i**2 \
    +i**3
print(sum) # 2355
# good practice to end line on the
operator '+'
```

Data Type

- Integers have no limits on their value.

```
num = 17**320  
#55418509492959182024050815403659657  
929728529771725585293012842056168997  
165580793547643430275803715662642272  
842832243289432165449023699657184114  
245868466923045382966077651495468000  
637704311015847106327664574761871274  
329808467403219150531075527293256040  
818605608055541618014533595136336836  
482778399338370133645413199500895371  
373463812637754232271926836143081204  
90424451247427541517608408025625601
```

Data Type

- Integers have no limits on their value.
- Floating point numbers, however, do have limits.



```
num = 1.8e308  
# inf
```

- 64-bit max $\approx 1.8 \times 10^{308}$
- 32-bit max $\approx 3.4 \times 10^{38}$
- 16-bit max = 65,504

Danger of overflow and underflow during mixed precision training

Data Type

- Linked lists are first-class citizens

```
n1 = list()
n1.append(3)
n1.append("hello")
n1.append(8.1)
#[1, 'hello', 0.33]
```

```
n1 = []
n1 = [3, 0.2]
```

Data Type

- Dictionaries are first-class citizens

```
dd = {1:0, 'n':0.3, "big": "better"}
```

```
print(dd)
```

```
#{1: 0, 'n': 0.3, 'big': 'good'}
```

```
print(dd["big"])
```

```
#good
```

Data Type

- Objects are not encapsulated
- Python does not have the private keyword.
- Based on convention, anything that starts with two underscores are private.



```
class Robot(object):  
    def __init__(self):  
        self.a = 123  
        self._b = 223  
        self.__c = 323
```

```
obj = Robot()  
print(obj.a)  
print(obj._b)  
print(obj.__c) # will not work  
print(obj.__dict__['_Robot__c'])  
# this works!  
# does allow for hacks
```

Functions

- Python has some aspects of a functional language
 - You can assign a function to a variable

```
def addone(x):  
    return x+1
```

```
def addtwo(x):  
    return x+2
```

```
f = addone  
print(f(1)) # result is 2  
f = addtwo  
print(f(1)) # result is 3
```

Functions

- Python has some aspects of a functional language
 - You can assign a function to a variable
 - Python provides higher-order functions like Lisp

```
import functools
def add_one_more(x, y):
    return x+y+1
```

```
f = functools.partial(add_one_more, 1)
# this creates a partial function
# whose first argument is known but the
# second is not.
```

```
print(f(2)) # result is 1+2+1=4
```

Functions

- Python has some aspects of a functional language
 - You can assign a function to a variable
 - Python provides higher-order functions like Lisp
 - `map()`, `filter()`, `reduce()`, etc.

```
def add_one(x):  
    return x+1
```

```
l1 = [1, 2, 3, 4]  
l2 = list(map(add_one, l1))  
print(l2) # [2, 3, 4, 5]
```

```
l3 = [add_one(x) for x in l1]  
print(l3) # [2, 3, 4, 5]
```

the two results are the same but
many prefer the second style

Multi-threading

- You can create multiple threads in Python but they don't work as you expect.
- The Global Interpreter Lock (GIL) ensures that only one thread is executing at a time.
- It is created as a simple fix for thread safety.
- Multi-threading leads to well-known problems of race conditions and dead locks.
- Python chose a simple solution: get rid of multi-threading
- Today, there is just too much path dependency to do anything about it.

Multi-processing

- We can circumvent the GIL by using the multiprocessing package of Python.
- A process has more overhead than a thread.
- All multi-threading problems, like race conditions and deadlocks, still need to be handled by the programmer.
- Those problems are inherent to any memory-sharing programs running in parallel.
- **Don't over-simplify problems.**

Performance

- Python is an interpreted, dynamically typed language.
- You can do a lot of unexpected things in Python
- Automatic optimization is hard because the program cannot predict what you will do.
- The result in slow execution
- To improve performance, avoid things like for loops and lists.
- Use packages like numpy, which calls fast libraries written in C.