Math notebook

Huy Bui

March 5, 2018

Contents

Part I

Matrices

Chapter 1

Abstract Data Types

1.1 Elements and tuples

In Computer Science, *datum* (data in plural) is a reference with a representation (syntax) which is stored in a memory. Datum a may be considered (semantically) as an element or object in some universe of discourse, i.e., universal set U, we write $a \in U$. When $A \subseteq U$ and $a \in U$, then a carries some *information* determined by A. The set A is understood as a *concept*.

Example 1.1. Consider a datum a which semantically is an element in a set of digital cars (which is a universal set). One might say that a is an *instance of an object class "digital car"*. However, the digital car a might be considered as an object whose properties are stored in computer memory in very many memory blocks. Naturally, each memory block should be referred somehow (by variables!) and in this sense there are datum semantically forming this digital car a.

Example 1.2. An element 5 may be interpreted in many different ways: $5 \in r$ or $5 \in \mathbb{N}$.

Example 1.3. When data is processed, it is usually obligatory to know in which universal set each datum belongs. For example, the term (expression) $5 + 4$ is not reasonable if $4, 5 \in \mathbf{S}$ ($\mathbf{S}$ is the set of arabic digits) and "+" is an operator on natural numbers. The operator "+" and $4, 5$ are not compatible.

Sometimes a datum may be a *tuple*. A tuple (or n-tuple) is an element of a product set, thus, $(a_1, \ldots, a_n) \in U_1 \times \cdots \times U_n$.

Example 1.4. If V is a set of color names, and M is a set of men's names, then $a = (75, \text{gray}, \text{Jari}) \in \mathbb{N} \times V \times W$. One might say that tuple is a multidimensional datum.

1.2 Memory and variables

In program languages, data is stored in computer memory and data is processed using symbols called *variables*. From a programmer point of view, a variable is a container, label, or frame. A variable (only a symbol) refers to a datum.

Program language compiler should transmit variable (symbol) to the operation system which then reserves on or more memory blocks to this symbol. Variable then refers to contents in computer memory block(s) having certain address(s), in which a datum is stored.

The use of variables in some context then serves as *access* to corresponding data. As we have seen, datum is always of some type in the sense that the applicable processing methods variate.

Example 1.5. We have two variables x and y, which refers to computer memory and there are datum. Remember that in computer memory blocks ONLY BITS ARE STORED. How the compiler of the current language "knows" that a program clause $z := x + y$ (or cannot) be processed in a reasonable (in the sense of current program) way?

In program languages, the variables must be *typed*. There are weakly and strongly typed program languages. *Note!* Operators demand data types and they know themselves the type.

1.3 Variables and data types

The first clauses in a program are often declaring of variables and their types, like `x::integer` or `y::float`. After that, variable assignment can be processed; in this presentation, values of variables (variable assignment) are given by `:=` symbol, for example `x:=0`.

A *data type* s is a symbolic name, which has a set A with operations O on that set as its semantics. The operations usually satisfy some laws, so, semantics of s is some specified algebra $\mathcal{A} = (A, O)$. A variable x of type s, written `x::s`, will be assigned to an element $a \in A$ interpreted as *a point in the algebra* $\mathcal{A}$, so we write `x:=` x.

Example 1.6. clauses `y::float` and `y:=5` together stores 5 in some specified memory block, but 5 is understood now as a real number. So, the bits in this memory block are handled as a floating point number.

Example 1.7. The clauses `z::char` and `z:='c'` together could store the ASCII-code of the letter c.

Example 1.8. If an operation ω takes values of data types $s_1, \ldots, s_n$ and produces values of data type t then this is often written as $\omega : s_1 \times \ldots \times s_2 rightarrow t$. In this case, data types may have a many-sorted algebra as their semantics.

Note! Some program languages (e.g. Python/Sage) process data as OBJECTS (object-oriented languages). Then, variables are often allowed to have *aliases* by default.

Example 1.9. In a clause `x:=5` the symbol `x` refers to a memory block containing object 5 (must be created); this object has a "nametag" `x`, i.e. this object is allowed to be referred by other variables also without copying the datum. This referencing method differs from symbol `x` being a "label of box" which contains 5.

1.4 Data structures

1.4.1 What is data structure?

Usually programs handle a large collection of different kinds of data from data type point of view. Some data may form an entity of its own, a datum, which may be called as *data aggregate*. Data aggregate is referred by one variable.

The data itself is naturally stored in different memory blocks, but it is now reasonable to assume that there must be some relation between elements, because the data aggregate is a datum itself.

Data structure is a way of organizing data such that data aggregate may be stored in and accessed from the memory efficiently and in a reasonable way from application point of view. The structure may be expressed as a relation between elements.

1.4.2 Tuple

One of the simplest data structure is *tuple*. Tuples are elements of a product set and in this sense it is a data aggregate. The elements in the tuple are not changeable (tuples are immutable) and the relation between elements is the identity relation.

Note! In a tuple each element may be of different data types. It might be possible to declare $x :: s_1, \ldots, s_n$ if x would create a new type from the existing ones (type constructor).

1.4.3 Array

Array is a data structure, where the elements have position numbers. The size of an array is fixed, but elements are changeable (arrays are mutable). There are no other specific methods to manipulate arrays.

In a more detail, an array f with the size of n position is a function, such that $f(i)$ is an element at the position number i, thus, $f : \{0, 1, \ldots, n - 1rbracerightarrowS$, when the stored elements are members in S.

The structure is now given as a function producing a linear order between the elements. Quite often arrays are declared like `x::array[0..n]` of `s`, where

`s` is some data type. For example, `x:=[3,4,5]` would assign a 3-placed array to `x` declared by `x::array[0..2]` of `s`, where `s` represents integer data type.

For *many-dimensional arrays*, one uses many different indices to position the elements. Notice that some modern program language do not demand array declarations, but the form of writing declares the array themselves.

If the array is declared, however, to be the size of n positions then the operating system reserves n memory blocks from the memory, even if the program would not need them all.

1.5 Abstract data type

Primitive data types, like integer or float, are sets with operations such that they are defined by hardware, operating system or compilers. These kinds of data types are called "built-in" such that the corresponding algebras are implemented in hardware, operating system, or compiler.

An *abstract data type s* is a symbolic name, which has a collection of *mathematically isomorphic algebras* as its semantics, i.e. the operations defined for these algebras behave in a similar manner and the underlying sets are bijective.

Example 1.10. An abstract data type *bool* has a collection of all 2-element Boolean algebras (mathematically isomorphic algebras) as its semantics. For example, *bookl* might have algebras like $B_1 = \{0, 1rbrace$ and $B_2 = \{\text{True}, \text{False}erbrace$ with the operations on these sets behaving in the same way (like connectives in Propositional Logic).

Example 1.11. Consider a program language with an abstract data type *bool*. The collection of all 2-element Boolean algebras is isomorphic to $\mathcal{B} = (B_2, *, +, ', F, T)$ as its semantics. In this program language, the syntactic operations for bool-type are $*, +, ', F, T$ where F is the zero element and T is the unit element. We have:

```
x,y,z::bool
z = x * y
WHILE z == T
...
ENDWHILE
```

These program clauses are independent from the actual implementation: it does not matter which 2-element Boolean algebra there is implemented.

Chapter 2

Matrices

2.1 Vector space

A vector space is a collection of objects called vectors, which may be added together and multiplied ("scaled") by numbers, called scalars.

Let F be the field[1] and X be abelian group[2]. The pair $(X, \odot)$ is a vector space if $\odot : F \times X \to X$ satisfies $\forall s, t \in F$ and $\forall u, v \in X$:

$$\begin{cases} s \odot (t \odot u) & = (s \cdot t) \odot u \\ s \odot (u + v) & = (s \odot u) + (s \odot v) \odot u \\ (s + t) \odot u & = (s + u) \odot (s + v) \\ 1 \odot u & = u \end{cases}$$

The elements of F (meaning s, t) are called "scalars". The elements of X (meaning u, v) are called "vectors". The habit to denote vectors in this notebook, say $u \in X$, by $\mathbf{u}$.

In this notebook, $F = \mathbb{R}$, thus we only discuss vector spaces. Additionally, we omit the symbol $\odot$, thus $s \odot \mathbf{u}$ is writing $s\mathbf{u}$.

Example 2.1. $\mathbb{R}$ is a vector space.

Example 2.2. $\mathbb{R}^2$ is also a vector space, where for $\mathbf{u} = (u_1, u_2)$ and $\mathbf{v} = (v_1, v_2)$ and $s \in \mathbb{R}$:

$$\begin{cases} s\mathbf{u} & = (su_1, su_2) \\ \mathbf{u} + \mathbf{v} & = (u_1 + u_2, u_2 + v_2) \\ \mathbf{0} & = (0, 0) \end{cases}$$

[1] a field is a set on which addition, subtraction, multiplication, and division are defined, and behave as when they are applied to rational and real numbers.

[2] an abelian group, also called a commutative group, is a group in which the result of applying the group operation to two group elements does not depend on the order in which they are written.

Example 2.3. Any $\mathbb{R}^n$ is a vector space, where for $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ and $s \in \mathbb{R}$ satisfies:

$$\begin{cases} s\mathbf{u} & = (su_1, su_2, \ldots, su_n) \\ \mathbf{u} + \mathbf{v} & = (u_1 + u_2, u_2 + v_2, \ldots, u_n + v_n) \\ \mathbf{0} & = (\underbrace{0, 0, \ldots, 0}_{\text{n times}}) \end{cases}$$

Note! If $\mathbf{u} \in \mathbb{R}^2$ and $\mathbf{v} \in \mathbb{R}^3$ then $\mathbf{u} + \mathbf{v}$ is not defined because $\mathbf{u}$ and $\mathbf{v}$ do not belong to the same vector space.

2.2 Definition

Let $\mathbb{R}^{m \times n}$ be the set of all tables with m rows and n columns such that they consist of real numbers. For example, $\begin{bmatrix} 2 & 3 & -1 \\ 2 & 1 & 4 \end{bmatrix}$ is a table of 2×3 Let define a table $\mathbf{A}$ such that:

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

Note that the element a_{ij}, called the ij-entry, appears in the ith row and the jth column. We simply denote $\mathbf{A} = [a_{ij}]$, where $i \in 1, 2, \ldots, m$ and $j \in 1, 2, \ldots, n$.

Now, $\mathbb{R}^{m \times n}$ is a vector space if $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$ and $s \in \mathbb{R}$ satisfies:

$$\begin{cases} \mathbf{A} + \mathbf{B} & = [a_{ij} + a_{ij}] \\ s\mathbf{A} & = [sa_{ij}] \\ [0] & \text{is the zero-element} \end{cases}$$

In this case, $[0]$ is an $m \times n$ table whose entries are all zero. $\mathbf{A}, \mathbf{B}, [0]$ are now called matrices. Notice that a vector can be regarded as a special type of matrix. A row vector is a $1 \times n$ matrix, and a column vector is a $m \times 1$ matrix.

Example 2.4. Vector $\mathbf{s} = (1, 2)$ can be written as a matrix $\begin{bmatrix} 1 & 2 \end{bmatrix}$ (row vector) or $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ (column vector).

2.3 Basic operations

2.3.1 Matrix addition

Let $\mathbf{A}$ and $\mathbf{B}$ are two matrices of the same vector space. The sum of $\mathbf{A}$ and $\mathbf{B}$, written $\mathbf{A}+\mathbf{B}$, is the matrix obtained by adding corresponding elements from $\mathbf{A}$ and $\mathbf{B}$:

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

2.3.2 Scalar multiplication

The product of a scalar k and a matrix $\mathbf{A}$, written $k\mathbf{A}$ or $\mathbf{A}k$, is the matrix obtained by muliplying each element of $\mathbf{A}$ by k:

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

We also define $-\mathbf{A} = (-1)\mathbf{A}$ and $\mathbf{A}-\mathbf{B} = \mathbf{A}+(-\mathbf{B})$. The matrix $-\mathbf{A}$ is called the *negative* of the matrix $\mathbf{A}$.

2.3.3 Matrix multiplication

Now suppose $\mathbf{A}$ and $\mathbf{B}$ are two matrices such that the number of columns of $\mathbf{A}$ is equal to the number of rows of $\mathbf{B}$, say $\mathbf{A}$ is an $m \times p$ matrix and $\mathbf{B}$ is an $p \times n$. Then the product of $\mathbf{A}$ and $\mathbf{B}$, written $\mathbf{AB}$, is the $m \times n$ matrix whose ij-entry is obtained by muliplying the elements of ith row of $\mathbf{A}$ by the corresponding elements of the jth column of $\mathbf{B}$ and then adding:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ . & . & \dots & . \\ a_{i1} & a_{i2} & \dots & a_{ip} \\ . & . & \dots & . \\ a_{m1} & a_{m2} & \dots & a_{mp} \end{bmatrix} \begin{bmatrix} b_{11} & \dots & b_{1j} & b_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix} = \begin{bmatrix} c_{11} & \dots & c_{1n} \\ \vdots & \vdots & \vdots \\ \vdots & c_{ij} & \vdots \\ \vdots & \vdots & \vdots \\ c_{m1} & \dots & c_{mn} \end{bmatrix}$$

where

$$c_{ij} = a_{i1}b_{1j} + a_{i1}b_{1j} + \ldots + a_{ip}b_{pj} = \sum_{k=1}^{p} a_{ik}b_{kj}$$

If the number of columns of $\mathbf{A}$ is not equal to the number of rows of $\mathbf{B}$, then the product $\mathbf{AB}$ is not defined.

Example 2.5. Find $\mathbf{AB}$, where

$$A = \begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 0 & -4 \\ 3 & -2 & 6 \end{bmatrix}$$

Since $\mathbf{A}$ is a 2×2 matrix, $\mathbf{B}$ is a 2×3, the product matrix $\mathbf{AB}$ is defined and is a 2×3 matrix. To obtain the elements in the first row of the product matrix $\mathbf{AB}$, multiply the first row $\begin{bmatrix} 1 & 3 \end{bmatrix}$ of $\mathbf{A}$ by the columns:

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ -2 \end{bmatrix} \qquad \text{and} \begin{bmatrix} -4 \\ 6 \end{bmatrix}$$

of $\mathbf{B}$ respectively:

$$\begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} 2 & 0 & -4 \\ 3 & -2 & 6 \end{bmatrix} = \begin{bmatrix} 1 \cdot 2 + 3 \cdot 3 & 1 \cdot 0 + 3(-2) & 1(-4) + 3 \cdot 6 \\ & & \end{bmatrix}$$
$$= \begin{bmatrix} 11 & -6 & 14 \\ & & \end{bmatrix}$$

To obtain the elements in the second row of the product matrix $\mathbf{AB}$, multiply the second row $\begin{bmatrix} 2 & -1 \end{bmatrix}$ of $\mathbf{A}$ by the columns of $\mathbf{B}$ respectively:

$$\begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} 2 & 0 & -4 \\ 3 & -2 & 6 \end{bmatrix} = \begin{bmatrix} 11 & -6 & 14 \\ 2 \cdot 2 + (-1) \cdot 3 & 2 \cdot 0 + (-1) \cdot (-2) & 2 \cdot (-4) + (-1) \cdot 6 \end{bmatrix}$$
$$= \begin{bmatrix} 11 & -6 & 14 \\ 1 & 2 & -14 \end{bmatrix}$$

2.3.4 Transpose

The transpose of a matrix is an operator which flips a matrix over its diagonal, that is, it switches the row and column indices of the matrix by producing another matrix.

Let $\mathbf{A} \in \mathbb{R}^{m \times n}$. The transpose of $\mathbf{A}$, denote $\mathbf{A}^T$, is a matrix $\mathbf{A}^T \in \mathbb{R}^{n \times m}$ such that for $\mathbf{A} = \begin{bmatrix} a_{ij} \end{bmatrix}$, we have $\mathbf{A}^T = \begin{bmatrix} a_{ji} \end{bmatrix}$.

Example 2.6. The transpose of $\mathbf{A} = \begin{bmatrix} 2 & 3 & 1 \\ 0 & 7 & 8 \end{bmatrix}$ is $\mathbf{A}^T = \begin{bmatrix} 2 & 0 \\ 3 & 7 \\ 1 & 8 \end{bmatrix}$.

For matrices $\mathbf{AB}$ and scalar k, we have the following properties of transpose:

- The operation of taking the transpose is an involution (self-inverse), meaning that $\left(\mathbf{A^T}\right)^T = \mathbf{A}$.
- The transpose respects addition, $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$.
- The order of the factors reverses when taking tranpose of a product matrix, $\mathbf{AB}^T = \mathbf{B}^T\mathbf{A}^T$. From this one can deduce that a square matrix $\mathbf{A}$ is invertible if and only if its transpose $\mathbf{A}^T$ is invertible, and in this case, we have $\left(\mathbf{A^{-1}}\right)^T = \left(\mathbf{A^T}\right)^{-1}$.
- $\mathbf{cA}^T = c\mathbf{A}^T$.

2.3.5 Inner product, norm, and unit vector

Let X be a vector space. If one can define a binary operation $\cdot : X \times X \to \mathbb{R}$ (take 2 matrices and produce a scalar) such that $\forall \mathbf{w}, \mathbf{u}, \mathbf{v} \in X$ and $\forall s \in \mathbb{R}$, we have:

$$\begin{cases} \mathbf{u} \cdot \mathbf{v} & = \mathbf{v} \cdot \mathbf{u} \\ \mathbf{u} \cdot (\mathbf{v} + \mathbf{w}) & = \mathbf{u} \cdot \mathbf{v} + \mathbf{u} \cdot \mathbf{w} \\ s\,(\mathbf{u} \cdot \mathbf{v}) & = (s\mathbf{u}) \cdot \mathbf{v} = \mathbf{u} \cdot (s\mathbf{v}) \\ \mathbf{u} \cdot \mathbf{u} & \geqslant 0 \\ \mathbf{u} \cdot \mathbf{u} = 0 \Leftrightarrow \mathbf{u} = 0 & \end{cases}$$

then X is called inner product space. The binary operation $\cdot$ is called inner product. Now, for any matrix $\mathbf{u} \in X$, one can define inner product norm $\|\mathbf{u}\|$ by

$$\|\mathbf{u}\| = \sqrt{\mathbf{u} \cdot \mathbf{u}}$$

For any two matrices $\mathbf{u}, \mathbf{v} \in X$, we have

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\| \cos\alpha$$

A vector whose norm is 1, is called a unit vector. One can define a unit vector $\widehat{\mathbf{x}}$ by the following formula:

$$\widehat{\mathbf{x}} = \frac{1}{\|\mathbf{x}\|}\mathbf{x}$$

Example 2.7. Vector $\mathbf{x} = (-2, 5, 3, 6)$ has inner product norm of $\|\mathbf{x}\| = \sqrt{(-2)^2 + 5^2 + 3^2 + 6^2} = \sqrt{74}$. Thus,

$$\widehat{\mathbf{x}} = \frac{1}{\|\mathbf{x}\|}\mathbf{x} = \frac{1}{\sqrt{74}}(-2, 5, 3, 6)$$

Example 2.8. Consider an object living in the 3D real place.

2.4 Square matrices

2.4.1 Definition

A matrix with the same number of rows as columns is called a *square matrix*. A square matrix with n rows and n columns is said to be of *order* n, and is called *n-square matrix*. The diagonal of an n-square matrix $\mathbf{A} = \left[a_{ij}\right]$ consists of elements $a_{11}, a_{22}, \ldots, a_{mn}$.

Example 2.9. The matrix

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

is a square matrix of order 3. The numbers along the diagonal are 1, -4, and 2.

2.4.2 Unit matrix

The n-square matrix with 1s along the diagonal and 0s elsewhere, e.g.,

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

is called *unit matrix* and will be denoted by $\mathbf{I}$. The unit matrix $\mathbf{I}$ plays the same role in matrix multplication as the number 1 does in the usual multiplication of number. Specifically,

$$\mathbf{AI} = \mathbf{IA} = \mathbf{A}$$

for any square matrix $\mathbf{A}$.

We can form powers of a square matrix $\mathbf{X}$ by defining:

$$\mathbf{X}^2 = \mathbf{XX}, \quad \mathbf{X}^3 = \mathbf{X}^2\mathbf{X} \quad \ldots \mathbf{X}^0 = \mathbf{I}$$

2.4.3 Invertible matrix

An n-square matrix $\mathbf{A}$ is said to be invertible if there exists an n-square matrix $\mathbf{B}$ with the property

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}_n$$

Such a matrix $\mathbf{B}$ is unique, and is called the *inverse* of $\mathbf{A}$, and is denoted by $\mathbf{A}^{-1}$. Observe that $\mathbf{B}$ is the inverse of $\mathbf{A}$ if and only if $\mathbf{A}$ is the inverse of $\mathbf{B}$.

Example 2.10. Suppose

$$\mathbf{A} = \begin{bmatrix} 2 & 5 \\ 1 & 3 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 3 & -5 \\ -1 & 2 \end{bmatrix}$$

Then

$$\mathbf{AB} = \begin{bmatrix} 6-5 & -10+10 \\ 3-3 & -5+6 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}_2$$

$$\mathbf{BA} = \begin{bmatrix} 6-5 & 15-15 \\ -2+2 & -5+6 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}_2$$

Thus $\mathbf{A}$ and $\mathbf{B}$ are inverses.

2.4.4 Determinants

To each n-square matrix $\mathbf{A} = \begin{bmatrix} a_{ij} \end{bmatrix}$ we assign a specific number called the *determinant* of $\mathbf{A}$, denoted $\det(\mathbf{A})$ or

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$$

The determinants of matrices of order one, two, three are defined as follows:

$$\begin{vmatrix} a_{11} \end{vmatrix} = a_{11}$$

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

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

In general, one can calculate determinant of an n-square matrix $\mathbf{A}$ using the following formula:

$$\det(\mathbf{A}) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(\mathbf{A_{ij}})$$

Using determinant, we can calculate the inverse of an invertible matrix. For example, the inverse of a 2×2 matrix with nonzero determinant can be calculated as follow:

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

For n-square matrix $\mathbf{A} = \left[a_{ij}\right]$ with nonzero determinant, the inverse would be:

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{vmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{vmatrix}$$

where $c_{ij} = (-1)^{i+j} \det(\mathbf{A_{ij}})$.

For matrices $\mathbf{AB}$ and scalar $k \neq 0$, we have the following properties of determinants:

- $\det(\mathbf{AB}) = \det(\mathbf{A}) \cdot \det(\mathbf{B})$
- A *square* matrix is invertible if and only if it has a nonzero determinant.
- Swapping two rows of a matrix produces the negative matrix. If $\mathbf{B}$ is a matrix $\mathbf{A}$ with any two rows interchanged, then $\det(\mathbf{B}) = -\det(\mathbf{A})$.
- If $\mathbf{B}$ is a matrix $\mathbf{A}$ but one row multiplied by k then $\det(\mathbf{B}) = k \det(\mathbf{A})$.
- If $\mathbf{B}$ is a matrix $\mathbf{A}$ but row ith, say R_i, is replaced by $R_i + kR_t, \forall t \neq i$, then $\det(\mathbf{B}) = \det(\mathbf{A})$.

Example 2.11. Given matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} -2 & -1 & 4 \\ 6 & -3 & -2 \\ 4 & 1 & 2 \end{bmatrix} \quad \det(\mathbf{A}) = 100$$

Interchange any two rows of $\mathbf{A}$, we have matrix $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} 6 & -3 & -2 \\ -2 & -1 & 4 \\ 4 & 1 & 2 \end{bmatrix} \quad \det(\mathbf{B}) = -\det(\mathbf{A}) = -100$$

Multiply first row of $\mathbf{A}$ with 4 , we have matrix $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} -8 & -4 & 16 \\ 6 & -3 & -2 \\ 4 & 1 & 2 \end{bmatrix} \quad \det(\mathbf{B}) = 4 \det(\mathbf{A}) = 400$$

Replace the first row R_1 by $R_1 + \frac{1}{2}R_3$

$$\mathbf{B} = \begin{bmatrix} 1 & -5/2 & 3 \\ 6 & -3 & -2 \\ 4 & 1 & 2 \end{bmatrix} \quad \det(\mathbf{B}) = \det(\mathbf{A}) = 100$$

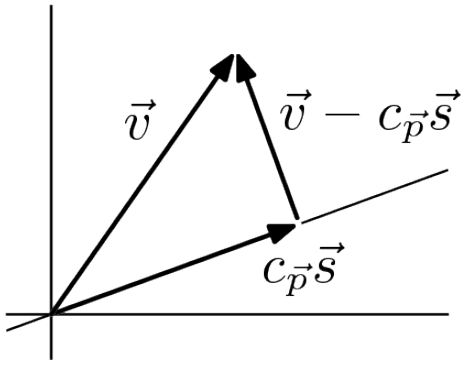
2.5 Matrix applications

2.5.1 Orthogonal projection

The orthogonal projection of $\mathbf{v}$ onto the line spanned by a nonzero $\mathbf{s}$ (see figure 2.1) is:

$$\mathbf{v_s} = \frac{\mathbf{v} \cdot \mathbf{s}}{\|\mathbf{s}\|^2} \mathbf{s}$$

Figure 2.1: Orthogonal projection



2.5.2 Linear least squares

Let $\mathbf{A}$ be an $m \times n$ matrix, $\mathbf{x}$ be an $n \times 1$ matrix and b be an $m \times 1$ matrix. Solve the matrix equation $\mathbf{Ax} = \mathbf{b}$.

If $m = n$, we can use three methods mentioned in section 3.2. However, if $m > n$, the the solution usually does not exist. But if the columns of $\mathbf{A}$ form a linearly independent set, then

$$\det\left(\mathbf{A^T A}\right) \neq 0 \quad \text{and} \quad \mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b} \tag{2.1}$$

$$\mathbf{x} = \mathbf{A^T A}^{-1}\mathbf{A}^T\mathbf{b} \tag{2.2}$$

The matrix equation now can be solved with the minimum error, meaning $\|\mathbf{Ax} - \mathbf{b}\|$ is minimum. Using formula (2.2) to solve a system of equations is called The method of linear least squares.

2.5.3 Rotation

In linear algebra, a rotation matrix is a matrix that is used to perform a rotation in Euclidean space.

In two dimensions

In two dimensions, every rotation matrix has the following form,

$$\mathbf{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

rotates points in the xy-plane counterclockwise through an angle α about the origin of the Cartesian coordinate system. The column vectors are calculated by means of the following matrix multiplication,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

So the new coordinates (x', y') of the point (x, y) after rotation are are

$$x' = x\cos\alpha - y\sin\alpha \quad y' = x\sin\alpha + y\cos\alpha\,.$$

In three dimensions

A basic rotation (also called elemental rotation) is a rotation about one of the axes of a Coordinate system. The following three basic rotation matrices rotate vectors by an angle θ about the x-, y-, or z-axis, in three dimensions, using the right-hand rule:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.6 Exercises

Exercise 2.1. Consider an object living in 3D real place. Its current position is $P(2,5,3)$. The object is heading towards the point $Q(4,10,5)$, and moves to that direction exactly 3 length units. Calculate the position of this object after movement.

Proof. Let $\mathbf{y}$ be the vector of the movement and $\mathbf{x}$ be the distance between P and Q. We have:

$$\begin{aligned}\mathbf{x} = \mathbf{Q} - \mathbf{P} &= (4,10,5) - (2,5,3) = (2,5,2)\\ \|\mathbf{x}\| = \sqrt{2^2+5^2+2^2} = \sqrt{33}, \quad \widehat{\mathbf{x}} &= \frac{1}{\sqrt{33}}(2,5,2)\\ \mathbf{y} = \mathbf{P} + 3\widehat{\mathbf{x}} &= (2,5,3) + \frac{3}{\sqrt{33}}(2,5,2)\end{aligned}$$

□

Exercise 2.2. Let $\mathbf{a} = (2,4)$ and $\mathbf{b} = (1,-3)$ be vectors in $\mathbb{R}^2$. If these vectors are considered as edges of a parallelogram then calculate its area A.

Proof. Let $\mathbf{A}$ be the matrix formed by $\mathbf{a}$ and $\mathbf{b}$

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

The area of the parallelogram is the absolute value of the determinant of $\mathbf{A}$

$$S = |\det(\mathbf{A})| = |2(-3) - 4 \cdot 1| = 10$$

□

Exercise 2.3. Suppose we want to draw a best fitting line through three points (4,-2), (5,0), (5,-1), (6,4), (7,5) and (8,5) so a linear function $y = t + kx$ should be given. Determine this function by means of the Method of Least Squares (LSM). Perform calculations by SAGE then plot points and the line in the same coordinate system.

Proof. The linear function $y = t + kx$ should satisfies the following system of equations with the minimum error:

$$\begin{cases} 4k + t &= -2 \\ 5k + t &= 0 \\ 5k + t &= -1 \\ 6k + t &= 4 \\ 7k + t &= 5 \\ 8k + t &= 5 \end{cases} \tag{2.3}$$

The equivalent matrix equation of (2.3) is $\mathbf{Ax} = \mathbf{b}$, where

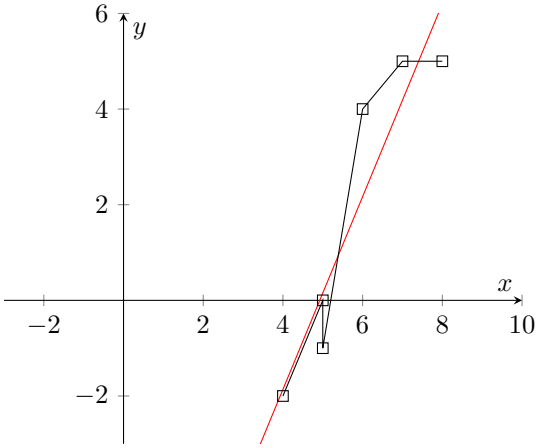
$$\mathbf{A} = \begin{bmatrix} 4 & 1 \\ 5 & 1 \\ 5 & 1 \\ 6 & 1 \\ 7 & 1 \\ 8 & 1 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} k \\ t \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -2 \\ 0 \\ -1 \\ 4 \\ 5 \\ 5 \end{bmatrix}$$

Using formula (2.2), we now can perform calculations by SAGE as follow:

```
A=matrix([[4,1], [5,1], [5,1], [6,1], [7,1], [8,1]])
b=matrix([ [-2,0,-1,4,5,5] ]).T
x=(A.T*A).inverse() * (A.T*b)

k=x[0][0]
t=x[1][0]
z=var("z")

S=plot(k*z+t, (z,-1,9))
P = scatter_plot([[4,-2], [5,0], [5,-1], [6,4], [7,5], [8,5]])
show(S+P)
```



□

Exercise 2.4. Suppose points (0,4),(1,8),(1,10),(2,5),(2,6), (8,10) and (6,10) are given. Approximate the points by a real trigonometric polynomial of degree 2. Plot the points and the graph of this function in the same coordinate system.

Proof. Suppose points (0,4),(1,8),(1,10),(2,5),(2,6), (8,10) and (6,10) are given. Approximate the points by a real trigonometric polynomial of degree 2. Plot the points and the graph of this function in the same coordinate system. □

Chapter 3

Linear equations

3.1 Linear independent set

Let X be any vector space. If vectors $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n} \in X$, then for any scalars $s_1, s_2, \ldots, s_n \in \mathbb{R}$, the vector

$$\mathbf{x} = \sum_{i=1}^{n} s_i \mathbf{a_i}$$

is called *a linear combination* of $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}$. Let S be a set of vectors $S = \{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$, where all vectors in S belong to the same vector space X. Denote the zero-element of vector space X as $\overline{0}$. If the set S is *linearly independent* then a linear combination of vectors in S equals zero-element if and only if all scalars in the linear combination are zero.

$$\sum_{i=1}^{n} s_i \mathbf{a_i} = \overline{0} \Leftrightarrow s_i = 0, \forall i \in \{1, 2, \ldots, n\}$$

Example 3.1. Let $X = \mathbb{R}^2$ and $S = \{\mathbf{a_1}, \mathbf{a_2}\}$, where $\mathbf{a_1} = (2, 4)$ and $\mathbf{a_2} = (1, 2)$. We have

$$s_1 \mathbf{a_1} + s_2 \mathbf{a_2} = \overline{0} \Leftrightarrow s_1 \begin{bmatrix} 2 \\ 4 \end{bmatrix} + s_2 \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Leftrightarrow \begin{cases} 2s_1 + s_2 & = 0 \\ 4s_1 + 2s_2 & = 0 \end{cases} \Leftrightarrow s_2 = -2s_1$$

Thus, the set S is not linearly independent. Notice that the equation $s_2 = -2s_1$ indicates $\mathbf{a_1}, \mathbf{a_2}$ are on the same line in the two-dimensional coordinate system. We can use $\mathbf{a_1}$ to define $\mathbf{a_2}$ based on the equation $s_2 = -2s_1$, and vice versa $\mathbf{a_1}$ can be defined by $\mathbf{a_2}$.

Example 3.2. Let $X = \mathbb{R}^2$ be a vector space and $S = \{\mathbf{a_1}, \mathbf{a_2}\}$, where $\mathbf{a_1} = (2, 4)$ and $\mathbf{a_2} = (-1, 2)$. We have

$$s_1\mathbf{a_1} + s_2\mathbf{a_2} = \overline{0} \Leftrightarrow s_1 \begin{bmatrix} 2 \\ 4 \end{bmatrix} + s_2 \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Leftrightarrow \begin{cases} 2s_1 - s_2 & = 0 \\ 4s_1 + 2s_2 & = 0 \end{cases} \Leftrightarrow \begin{cases} s_2 & = 2s_1 \\ s_2 & = -2s_1 \end{cases} \Leftrightarrow s_2 = s_1 = 0$$

Thus, the set S is linearly independent. Notice that the equation $s_2 = s_1 = 0$ indicates there is no way to define $\mathbf{a_1}$ using $\mathbf{a_2}$, or vice versa.

Example 3.3. Show that vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^5$ does not form a linearly independent set if $\mathbf{u} = (-1, 0, 8, 5, 2)$ and $\mathbf{v} = (-6, 0, 48, 30, 12)$.

The set $\{\mathbf{u}, \mathbf{v}\}$ is linearly independent if and only if

$$s_1\mathbf{u} + s_2\mathbf{v} = \overline{0} \Leftrightarrow s_1 = s_2 = 0 \tag{3.1}$$

We have $s_1\mathbf{u} + s_2\mathbf{v} = \overline{0}$ is equivalent to

$$\begin{cases} -s_1 - 6s_2 & = 0 \\ 8s_1 + 48s_2 & = 0 \\ 5s_1 + 30s_2 & = 0 \\ 2s_1 + 12s_2 & = 0 \end{cases}$$

All of these equations are equivalent to $s_1 + 6s_2 = 0$. There are an infinite number of solutions to this equations, which does not satisfies (3.1).

Let X be a vector space and the set $A \subseteq X$ is linearly independent. If any vectors $\mathbf{u} \in X$ is a linear combination of vectors in A, then A is a *base of* X.

3.2 System of linear equations

Given $\mathbf{A}$ an n-square matrix and two column vectors $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$ and $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}^T$. The system of n linear equations in n unknowns is equivalent to the matrix equation:

$$\mathbf{Ax} = \mathbf{b}$$

The matrix $\mathbf{A}$ is called the *coefficient matrix*. The properties of this matrix, especially determinant, decide the solution of the equation.

The collection of all columns in $\mathbf{A}$ form a lineary independent set (rank of $\mathbf{A}$ is n) if and only if $\det(\mathbf{A}) \neq 0$. This also means that there exists a unique solution for the equation. On the other hand, if $\det(\mathbf{A}) = 0$, then either of two

cases occurs: the system has no solutions, or there are an infinite number of solutions.

There are many ways to solve a system of equations, however, in this notebook, only three solutions related to matrix are mentioned: Inverse matrix, Crammer's rule, Gaussian elimination.

3.2.1 Inverse matrix

Given the matrix equation $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is an n-square matrix, $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$ and $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}^T$. If $\det(\mathbf{A}) \neq 0$, then $\mathbf{A}$ is invertible and there is a unique solution to the equation:

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

Observe that this method only works if the determinant of the coefficient matrix is nonzero.

Example 3.4. Solve the following system of equation:

$$\begin{cases} 2x_1 + 4x_2 &= 8 \\ 7x_1 - 2x_2 &= 4 \end{cases} \tag{3.2}$$

Let define matrix $\mathbf{A}$, $\mathbf{x}$, and $\mathbf{b}$ as follow:

$$\mathbf{A} = \begin{bmatrix} 2 & 4 \\ 7 & -2 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 8 \\ 4 \end{bmatrix}$$

The system of equations is equivalent to

$$\mathbf{Ax} = \mathbf{b}$$

The determinant of coefficient matrix is

$$\det(\mathbf{A}) = 2 \cdot (-2) - 7 \cdot 4 = -32 \neq 0$$

Thus, the equation has a unique solution, which is

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = \frac{1}{-32} \begin{bmatrix} -2 & -4 \\ -7 & 2 \end{bmatrix} \begin{bmatrix} 8 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 3/2 \end{bmatrix}$$

3.2.2 Crammer's rule

Suppose we need to solve the matrix equation $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is an n-square matrix, $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$ and $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}^T$. Let $\mathbf{A}^{(j)}$ be a matrix $\mathbf{A}$ but replace jth column with $\mathbf{b}$. Then we have

$$x_j = \frac{\det\left(\mathbf{A}^{(\mathbf{j})}\right)}{\det(\mathbf{A})}$$

Like Inverse matrix, this method only works if the determinant of the coefficient matrix is nonzero.

Example 3.5. The system of equation (3.2) can solved using Crammer's rule as follow:

$$\mathbf{A} = \begin{bmatrix} 2 & 4 \\ 7 & -2 \end{bmatrix}, \quad \det(\mathbf{A}) = -32$$

$$\det\left(\mathbf{A}^{(\mathbf{1})}\right) = \begin{vmatrix} 8 & 4 \\ 4 & -2 \end{vmatrix} = -32 \Rightarrow x_1 = \frac{-32}{-32} = 1$$

$$\det\left(\mathbf{A}^{(\mathbf{2})}\right) = \begin{vmatrix} 2 & 7 \\ 8 & 4 \end{vmatrix} = -48 \Rightarrow x_2 = \frac{-48}{-32} = \frac{3}{2}$$

3.2.3 Gaussian elimination

Gaussian elimination (also known as row reduction) is an algorithm for solving systems of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients. This method can also be used to find the rank of a matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix.

To perform Gaussian elimination, one uses a sequence of elementary row operations to modify the matrix until the lower left-hand corner of the matrix is filled with zeros, as much as possible. There are three types of elementary row operations:

- Swap the positions of two rows.
- Multiply a row by a nonzero scalar.
- Add to one row a scalar multiple of another.

These operations are already shown in section 2.4.4 as properties of determinants.

Example 3.6. The system of equation (3.2) can solved using Gaussian elimination. Let define matrix $\mathbf{A}$, $\mathbf{x}$, and $\mathbf{b}$ as follow:

$$\mathbf{A} = \begin{bmatrix} 2 & 4 \\ 7 & -2 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 8 \\ 4 \end{bmatrix}$$

First, we create the augmented matrix which is formed by merging matrix $\mathbf{A}$ and $\mathbf{b}$.

$$\left[\begin{array}{cc|c} 2 & 4 & 8 \\ 7 & -2 & 4 \end{array}\right]$$

Swap the positions of two rows, we have:

$$\left[\begin{array}{cc|c} 7 & -2 & 4 \\ 2 & 4 & 8 \end{array}\right]$$

Multiply the first row by $1/7$, we have:

$$\left[\begin{array}{cc|c} 1 & -2/7 & 4/7 \\ 2 & 4 & 8 \end{array}\right]$$

Replace the second row by $R_2 \leftarrow R_2 - 2R_1$

$$\left[\begin{array}{cc|c} 1 & -2/7 & 4/7 \\ 0 & 32/7 & 48/7 \end{array}\right]$$

Multiply the first row by $32/7$, we have:

$$\left[\begin{array}{cc|c} 1 & -2/7 & 4/7 \\ 0 & 1 & 48/32 \end{array}\right]$$

Replace the first row by $R_1 \leftarrow R_2 + \frac{2}{7}R_1$

$$\left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 48/32 \end{array}\right]$$

Thus, $x_1 = 1$ and $x_2 = 48/32$.