Applied Healthcare Data Science

Mathematics Basis Review

Semester 2, 2023

Basic mathematics skills for this unit

This unit requires the basics of:

1. Linear algebra.
2. Calculus.
3. Probability.
4. Statistical inference.
5. Computer programming.

We will therefore start with this background knowledge so that you can maximise your learning.

Basic mathematics skills for this unit

- We assume that you are already familiar with most of this material from previous studies, possibly with a few things to catch up on.
- Therefore, the following sections that are just a brief discussion of selected concepts.
- Refer to the notes for a complete review.

Linear Algebra

Scalars

A **scalar** is a simple numerical value such as 2 or -1.5. We use lower case letters like x or a to denote a variable or constant that takes a scalar value.

Vectors

A **vector** is an ordered finite list of scalars. We write vectors as vertical arrays such as

$$\begin{bmatrix} -1 \\ 0 \\ 2.5 \\ -7.2 \end{bmatrix} \quad \text{or} \quad \begin{pmatrix} -1 \\ 0 \\ 2.5 \\ -7.2 \end{pmatrix},$$

or a list of numbers separated by commas, for example $(-1, 0, 2.5, -7.2)$.

The number of elements is the **size** or **dimension** of the vector.

Vectors

We denote a vector as a lower case bold letter such as $\boldsymbol{a}$ and $\boldsymbol{y}$.

We represent a general $n-$dimensional vector $\boldsymbol{a}$ as

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}.$$

We denote the $i-$th element of $\boldsymbol{a}$ as a_i.

Response vector

In supervised learning, our objective is to build a model to predict a **response variable**. The response data is the vector

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

where y_i refers to the observed value of the response variable for observational unit (individual) i.

Matrices

A matrix is a rectangular two-dimensional array of numbers such as

$$\begin{bmatrix} 0 & 1 & -2.3 & 0.1 \\ 1.3 & 4 & -0.1 & 7 \\ 4.1 & -1 & 0 & 1.7 \end{bmatrix}$$

The number of rows and columns are the **dimensions** of a matrix. The matrix above has 3 rows and 4 columns, so the size is 3×4 (it reads 3-by-4).

Matrices

We denote a matrix as an upper case bold letter such as $\boldsymbol{A}$ and $\boldsymbol{X}$. We represent a general $m \times n$ matrix as

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

where a_{ij} denotes the element in the $i-$th row and $j-$th column of $\boldsymbol{A}$ (we also sometimes write $\boldsymbol{A}_{ij}$).

Design matrix

In supervised learning, we predict the response variable based on input or predictor variables. We represent the predictor data as the matrix

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix},$$

where n is the number of observations, p is the number of predictor variables, and x_{ij} denotes the value of predictor j for observation i.

We refer to this matrix as the **design matrix**.

Scalar multiplication

Let α denote a scalar. The vector $\alpha\,\mathbf{a}$ is the vector with elements $\alpha\,a_i$.

Example. Let $\mathbf{a} = (5, -2, -3)$. Then

$$0.5\,\mathbf{a} = (0.5 \times 5, 0.5 \times -2, 0.5 \times -3) = (2.5, -1, -1.5)$$

Vector addition

Let $\mathbf{a}$ and $\mathbf{b}$ be two vectors with the same size n. The sum $\mathbf{c} = \mathbf{a} + \mathbf{b}$ is the vector with elements $c_i = a_i + b_i$.

Example. Let $\mathbf{a} = (5, -2, -3)$ and $\mathbf{b} = (-1, 2, 4)$. Then,

$$\mathbf{a}+\mathbf{b} = (5, -2, -3) + (-1, 2, 4) = (5-1, -2+2, -3+4) = (4, 0, 1).$$

Inner product

We define the **inner product** or **dot product** between two n-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$ as

$$\boldsymbol{a}^T\boldsymbol{b} = a_1b_1 + a_2b_2 + \ldots + a_nb_n = \sum_{i=1}^{n} a_ib_i$$

Example. Let $\mathbf{a} = (2, -1, 3)$ and $\mathbf{b} = (5, -2, -3)$. Then,

$$\boldsymbol{a}^T\boldsymbol{b} = 2 \times 5 + (-1) \times (-2) + 3 \times (-3) = 3$$

Norm

A **norm** is a measure of the **length** or size of a vector (see the notes for a formal definition).

The **Euclidean norm** or ℓ_2-norm of a vector is

$$\|\mathbf{a}\|_2 = \sqrt{\mathbf{a}^T\mathbf{a}} = \left(\sum_{i=1}^{n} a_i^2\right)^{1/2}.$$

Distance

The **Euclidean distance** between two vectors $\mathbf{x}$ and $\mathbf{y}$ is the Euclidean norm of the difference vector $\mathbf{x} - \mathbf{y}$:

$$\text{dist}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2 = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y})} = \left(\sum_{i=1}^{n}(x_i - y_i)^2\right)^{1/2}$$

Every norm $\|\cdot\|$ induces a distance metric $\|\boldsymbol{x} - \boldsymbol{y}\|$.

Example. Let $\mathbf{a} = (2, -1, 3)$ and $\mathbf{b} = (5, -2, -3)$. Then,

$$\begin{aligned}\text{dist}(\boldsymbol{a}, \boldsymbol{b}) &= \sqrt{(2-5)^2 + (-1-(-2))^2 + (3-(-3))} \\ &= \sqrt{-3^2 + 1^2 + 6^2}\end{aligned}$$

Matrix transpose

The **transpose** of an $m \times n$ matrix $\boldsymbol{A}$, denoted as $\boldsymbol{A}^T$, yields an $n \times m$ matrix that interchanges the rows and columns of $\boldsymbol{A}$.

Example.

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 2 & -1 \\ 3 & -2 & 5 \\ -2 & 4 & 1 \end{bmatrix}, \qquad \mathbf{A}^T = \begin{bmatrix} 2 & 1 & 3 & -2 \\ 3 & 2 & -2 & 4 \\ 4 & -1 & 5 & 1 \end{bmatrix}$$

Matrix-vector multiplication

The product of an $m \times n$ matrix $\boldsymbol{A}$ with an n-vector $\boldsymbol{b}$ is an m-vector $\boldsymbol{c}$ with element i equal to the inner product of the row i of $\boldsymbol{A}$ with $\boldsymbol{b}$.

$$c_i = \boldsymbol{a}_i^T \boldsymbol{b} = \sum_{j=1}^{n} a_{ij} b_j,$$

where $\boldsymbol{a}_i^T$ denotes i-th row of $\boldsymbol{A}$.

Example.

$$\begin{bmatrix} 1 & 4 \\ 7 & -3 \\ 2 & -5 \end{bmatrix} \times \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \times 2 + 4 \times 1 \\ 7 \times 2 - 3 \times 1 \\ 2 \times 2 - 5 \times 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 11 \\ -1 \end{bmatrix}$$

Matrix multiplication

The product of an $m \times p$ matrix $\boldsymbol{A}$ with an $p \times n$ matrix $\boldsymbol{B}$ is an $m \times n$ matrix $\boldsymbol{C} = \boldsymbol{AB}$ with element ij equal to the inner product of the row i of $\boldsymbol{A}$ with column j of $\boldsymbol{B}$

$$c_{ij} = \boldsymbol{a}_i^T \boldsymbol{b}_j = \sum_{k=1}^{p} a_{ik} b_{kj}$$

where $\boldsymbol{a}_i^T$ denotes i-th row of $\boldsymbol{A}$ and $\boldsymbol{b}_j$ denotes j-th column of $\boldsymbol{B}$.

Example.

$$\begin{bmatrix} 2 & 3 \\ 3 & -2 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 \times 1 + 3 \times 3 & 2 \times 2 + 3 \times 4 \\ 3 \times 1 - 2 \times 3 & 3 \times 2 - 2 \times 4 \end{bmatrix} = \begin{bmatrix} 11 & 16 \\ -3 & -2 \end{bmatrix}$$

Mathematics

Set

A **set** is an *unordered* collection of unique elements. We denote a set using a calligraphic capital letter such $\mathcal{N}$ or $\mathcal{S}$. We write $x \in \mathcal{S}$ to say that an element x belongs to set $\mathcal{S}$.

We write finite sets using curly brackets, for example $\{a_1, a_2, \ldots, a_n\}$ for a set of scalars. A set can also be infinite, for example all values in some interval.

In this unit may refer to sets not only of scalars but also data points, vectors, functions, and models.

Function

A **function** is a relation that associates each element x of a set $\mathcal{X}$, called the **domain** of the function, to a single element y of another set $\mathcal{Y}$, the codomain of the function.

For example, if the function is called f, we may write $y = f(x)$ (read f of x). The element x is the **argument** or **input**, and y is the value or **output**.

Logarithms and exponentials

The (natural) exponential and logarithm functions appear many times in this unit. Important properties (x and y are scalars):

$$\exp(0) = 1$$

$$\exp(x + y) = \exp(x) \exp(y)$$

$$\exp(x - y) = \exp(x) / \exp(y)$$

$$\exp(\log(x)) = x$$

$$\log(1) = 0$$

$$\log(xy) = \log(x) + \log(y)$$

$$\log(x/y) = \log(x) - \log(y)$$

$$\log(x^y) = y \log(x)$$

$$\log(1/x) = -\log(x)$$

Derivative

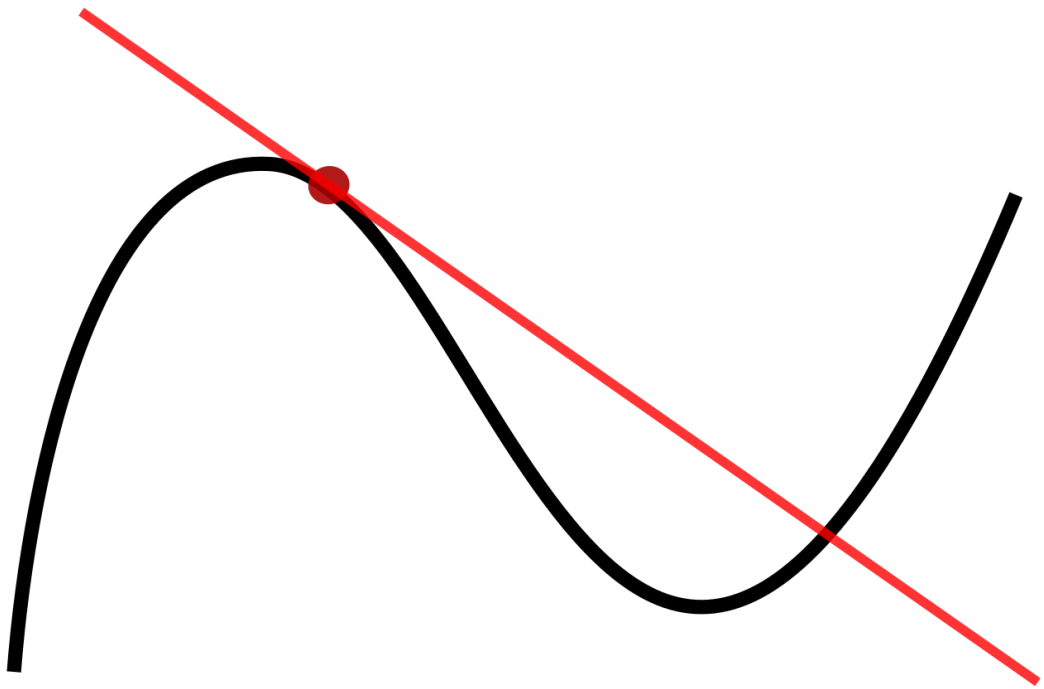
Informally, the **derivative** f' of a function $f(x)$ is a function that measures the how fast the value of f (the output) increases (or decreases) as we change the argument (input) x.

We denote the derivative as $f'(x)$ or $\frac{df(x)}{dx}$.

The process of finding a derivative is called **differentiation**.

Derivative

The slope of the tangent line corresponds to the derivative of the function at the indicated point.



Differentiation rules

We need to know the basic rules for differentiation. Here are some examples (see the notes for a complete list):

- If $f(x) = c$ for a constant c, then $f'(x) = 0$.
- If $f(x) = cx$ for a constant c, then $f'(x) = c$.
- If $f(x) = x^2$, then $f'(x) = 2x$.
- If $f(x) = \log(x)$, then $f'(x) = 1/x$.
- If $f(x) = \exp(x)$, then $f'(x) = \exp(x)$.
- If $h(x) = f(x) + g(x)$, then $h'(x) = f'(x) + g'(x)$.

Chain rule

The **chain rule** helps us to differentiate more complicated functions. Let $h(x) = f(g(x))$ for some functions f and g, Then $h'(x) = f'(g(x))g'(x)$.

Example. Let $h(x) = (5x + 1)^2$. Then $g(x) = 5x + 1$ and $f(g(x)) = g(x)^2$. Using the chain rule and the rules from the previous slide,

$$h'(x) = 2g(x)g'(x) = 2(5x + 1)5 = 50x + 10.$$

Gradient

The **gradient** is a generalisation of the derivative for functions of many variables (inputs). The gradient is a vector of **partial derivatives**.

Example.

Let $f(x_1, x_2) = ax_1 + x_2^2 + c$. The partial derivatives are

$$\frac{\partial f}{\partial x_1} = a, \quad \frac{\partial f}{\partial x_2} = 2x_2,$$

and the gradient is

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right) = (a, 2x_2).$$

Gradient

Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. We write

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{pmatrix}$$

or

$$\frac{d\, f(\boldsymbol{x})}{d\boldsymbol{x}} = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_n} \right)$$

Probability

Random variables

A **random variable** is a variable that takes a numerical value that depends on the outcome of a random phenomenon.

A **random vector** is a vector of random variables.

We generally use a capital letter such as Y and X to denote both random variables and random vectors.

Expected value

We define the **expected value** or **mean** of a random variable X as

$$\mathbb{E}(X) = \begin{cases} \sum_x xp(x) & \text{if } X \text{ is discrete} \\ \int xp(x)dx & \text{if } X \text{ is continuous,} \end{cases}$$

where $p(x)$ is the probability mass function or density function for discrete and continuous random variables respectively.

We can think of $\mathbb{E}(X)$ as the average $\sum_{i=1}^n X_i/n$ for a very large number of IID draws $X_1, \ldots, X_n$.

Properties of expectations

If X and Y are random variables and a and b are constants,

$$\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y).$$

More generally, if $X_1, \ldots, X_n$ are random variables and $a_1, \ldots, a_n$ are constants,

$$\mathbb{E}\left(\sum_{i=1}^{n} a_i X_i\right) = \sum_{i=1}^{n} a_i \mathbb{E}(X_i).$$

We call this property the linearity of expectations. It is extremely important that you remember this property.

Variance and covariance

Let X be a random variable with mean μ. We define the **variance** of X, denoted by $\mathbb{V}(X)$, $\mathrm{Var}(X)$, or σ_X^2 as

$$\mathbb{V}(x) = \mathbb{E}(X - \mu)^2.$$

Properties of variances

Let X and Y be random variables and a and b constants. We frequently use the following properties. Memorise them all.

$$\mathbb{V}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$$

$$\mathbb{V}(aX + b) = a^2\mathbb{V}(X)$$

$$\mathbb{V}(X + Y) = \mathbb{V}(X) + \mathbb{V}(Y) + 2\mathsf{Cov}(X, Y)$$

$$\mathbb{V}(X - Y) = \mathbb{V}(X) + \mathbb{V}(Y) - 2\mathsf{Cov}(X, Y)$$

$$\mathbb{V}(aX + bY) = a^2\mathbb{V}(X) + b^2\mathbb{V}(Y) + 2\,ab\mathsf{Cov}(X, Y)$$

If $X_1, \ldots, X_n$ are independent random variables and $a_1, \ldots, a_n$ are constants,

$$\mathbb{V}\left(\sum_{i=1}^{n} a_i X_i\right) = \sum_{i=1}^{n} a_i^2 \mathbb{V}(X_i).$$

Statistical inference

Estimation

Point estimation is the processing of computing a single "best guess" of some quantity of interest, called the **estimand**.

Let $X_1, \ldots, X_n$ be n data points from some distribution F. A **point estimator** $\widehat{\theta}$ is a function

$$\widehat{\theta} = g(X_1, \ldots, X_n)$$

of $X_1, \ldots, X_n$.

The estimator is a random variable, since it depends on a random sample.

Estimation

- We refer to the observed value of an estimator as the **estimate**.
- It is fundamental not to confuse these two concepts: the estimator is a random variable, while the estimate is the numerical value that we obtain by computing the estimator for a dataset.
- We use the hat notation as in $\widehat{\theta}$ or $\widehat{f}(x)$ for both estimators and estimates. You should be able to make the distinction from the context.

Variance of an estimator

We call the distribution of $\widehat{\theta}$ the **sampling distribution**.

We are often interested in the **sampling variance** of the estimator, denoted as $\mathbb{V}(\widehat{\theta})$.

The standard deviation of $\widehat{\theta}$ is called the **standard error**, denoted by se:

$$\text{se}(\widehat{\theta}) = \sqrt{\mathbb{V}(\widehat{\theta})}.$$

Bias

We define the **bias** of an estimator as

$$\text{Bias}(\widehat{\theta}) = \mathbb{E}(\widehat{\theta}) - \theta,$$

where θ is the estimand and the expectation is with respect to the data. We say that $\widehat{\theta}$ is **unbiased** when $\mathbb{E}(\widehat{\theta}) = \theta$.

Mean squared error

We often evaluate the quality of an estimator by the (population) **mean squared error**

$$\text{MSE} = \mathbb{E}(\widehat{\theta} - \theta)^2,$$

where θ is the estimand and the expectation is with respect to the data. We can write the MSE as

$$\text{MSE} = \text{bias}^2(\widehat{\theta}) + \mathbb{V}(\widehat{\theta}),$$

which is called the **bias-variance decomposition**.

Applied Healthcare Data Science

Programming Basis Review

Semester 2, 2023

Basic Programming skills for this unit

This unit requires the basics of:

1. **Mechine learning related packages**
2. Basic grammar of python
3. Machine learning related programming

Common Packages Used in Machine Learning

- For predefined ML methods
  - pytorch
  - tensorflow
  - scikit-learn (sklearn)
- For better data structure
  - numpy
  - pandas
- For visaulization
  - matplotlib
  - tensorboard
  - wandb

Basic Programming skills for this unit

This unit requires the basics of:

1. Mechine learning related packages (numpy, pytorch, scikit-learn, etc.)
2. **Basic grammar of python**
3. Machine learning related programming

Grammar

1. Value & Variable
2. Sentence
    - 2.1. Simple Ssentence
    - 2.2. Branch
    - 2.3. Loop
3. Function
4. Class (object)

Value

Types of Value:	**Examples**
1. Numeric	
• Integer (int):	**1, 2, 3, 5, 8, 13**
• Decimal (float):	**2.718, 3.14**
• No fraction	
2. Boolean(bool)	**True, False**
3. String(str)	**':(', "123", "wasd", ",./123qwe", ""**
4. NoneType	**None**

Value

Types of Value:

5. List: sequence of elements

   e.g. **[0, 1, 2], [‘a’, ‘c’, ‘b’], [True, False, ‘l’, [1, 2]], []**

   5.1. Label of an element: the sequence ot that element

   - Always start from 0

6. Dictionary (dict): A set of elements

   e.g. **{False:1, 2:‘b’, ‘c’: True}, {}**;

   6.1. Composition of an elements: key-value pair, divided by a colon

   6.2. Label of an element: the key of it

Variable

1. Composition of Variables: Name and Value

e.g.

```
a = 3
e = 2.718
is_variable = True
str_0 = 'coding is ez :)'
list_0 = [1, 2, 3, 4]
dict_0 = {'intro': 'this is a dictionary', 'e': 2.718}
```

1.1. When using a variable in the code, use its name.

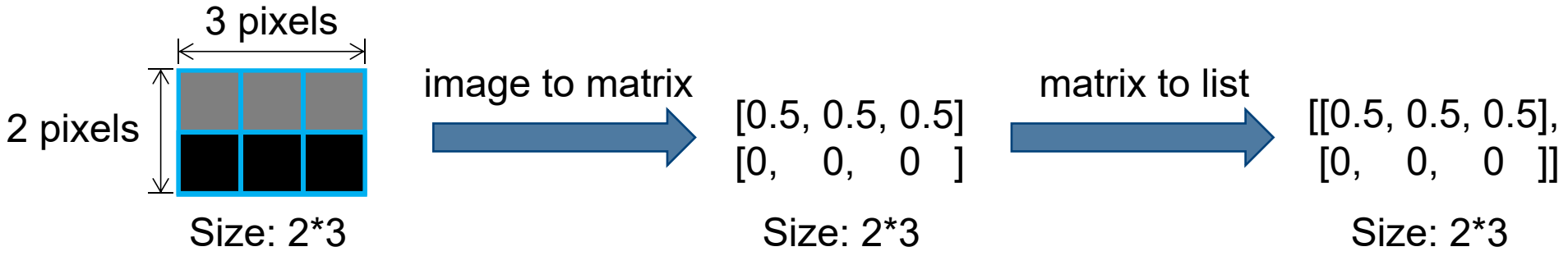
2. Types of variable: determined by its value

Use a variable to represent an image/matrix

Types of image representation :

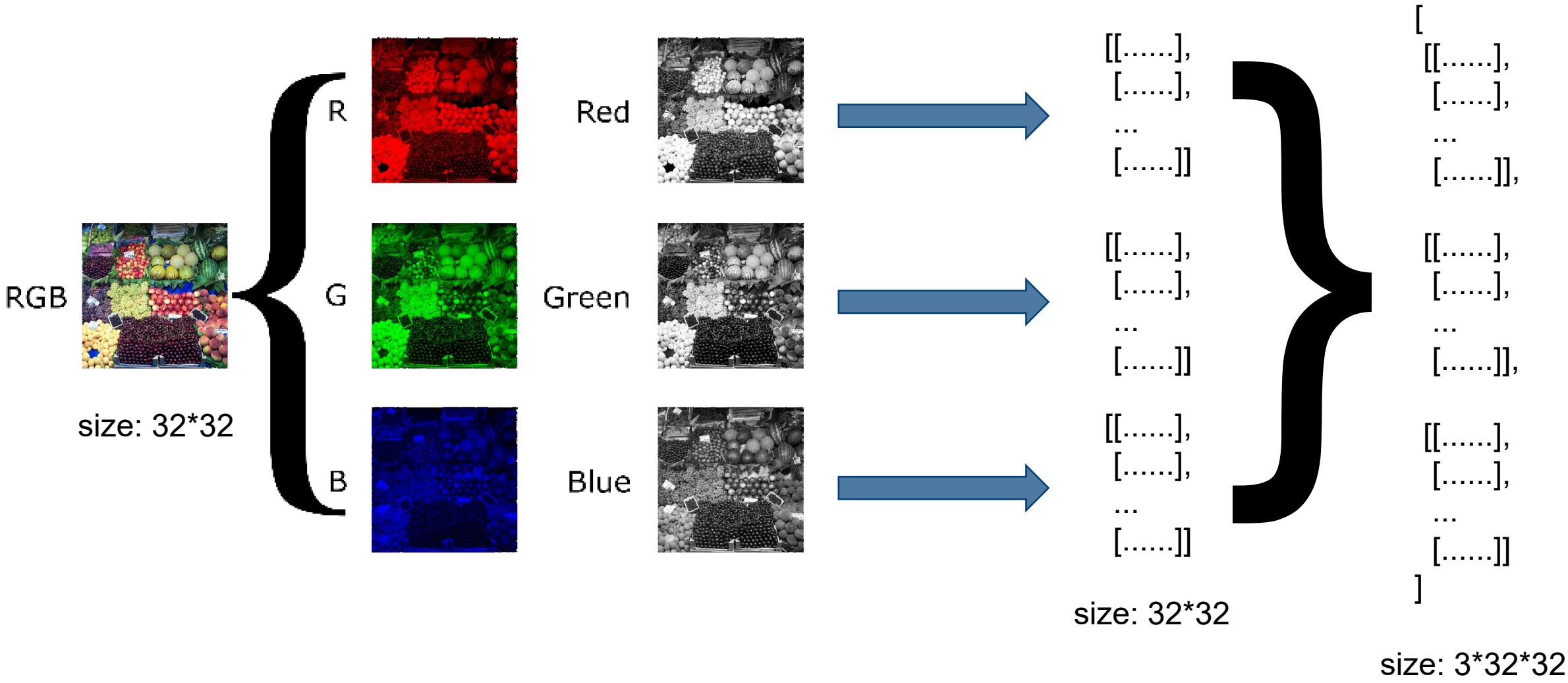
1. Binary image: 0, 1 (boolean)
2. Gray scale image: 0~1
3. RGB image:
   3 channels, 0~255 / 0~1

1. Representation of a gray scale image



Use a variable to represent an image/matrix

2. Representation of an RGB image



Simple Sentence

1. Comments: From # to the end of line. Computer never read them
2. Import lines: form: **from XXX import XXX as XXX**

   e.g.

   ```
   import torch
   import numpy as np
   from torchvision import datasets as ds
   ```

3. Assignment: form: **Variable = Value / Variable**

   e.g.

   ```
   a = 1
   b = ':)'
   ```

Simple Sentence

4. Boolean Calculation (a boolean value)

4.1. Calculation signs

meaning	sign
equal	==
inequality	!=
less	<
less than or equal to	<=
greater	>
greater than or equal to	>=

4.2. Descriptive signs: or, and, not, is

5. Function Calls

Simple Sentence

6. Formula (a Value)

6.1. Numeric calculation

operator	operation
+	addition
-	subtraction
*	multiplication
/	division
**	to the power of
//	rounded division
()	specifying the priority

6.2. String calculation

- string + string: merge two strings

  e.g.

  a = “ab” + “ba” # a == “abba”

- string * integer: repeat the string

  e.g.

  a = “ab” * 3 # a == “ababab”

  a = 4 * “ab” # a == “abababab”

6.3. Special situation: Iterative operations

e.g. a = 2

a *= 3 # a == 6

a **= 2 # a == 36

a += 1 # a == 37

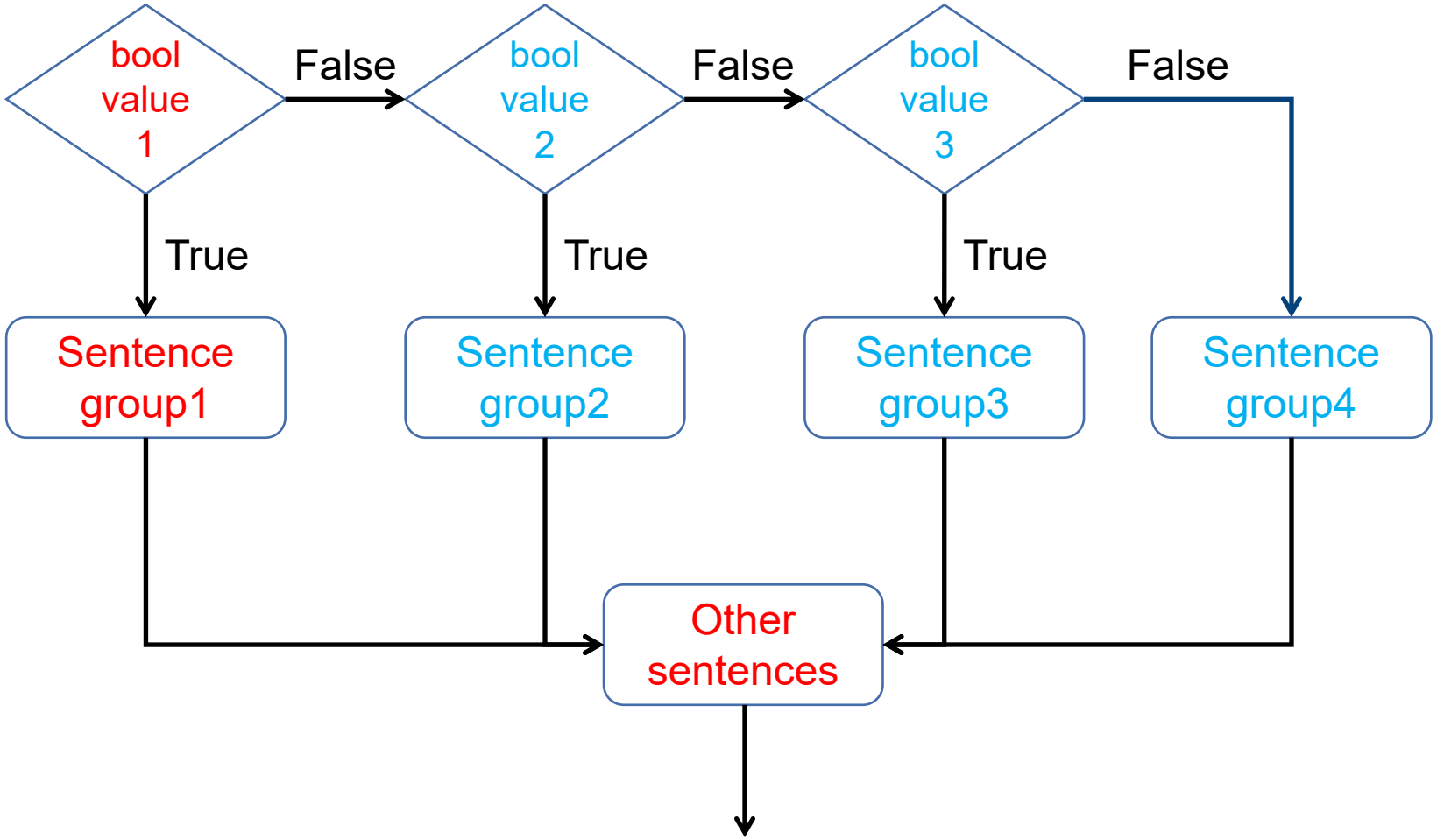
a //= 3 # a == 12

Branch

1. form

```
If <bool value1>:
    <Sentence group1>
elif <bool value2>:
    <Sentence group2>
elif <bool value3>:
    <Sentence group3>
...
else:
    <Sentence group4>
```

2. logic



Loop

1. while

   form:

   while <bool value>:

   <Sentence group>

2. for

   form:

   for <variable> in <list>:

   <Sentence group>

- Skip the loop
  - break: Quit the loop
  - contionue: Skip current round

Function

- A Sentence group
- Return into a value (can be NoneType)

  form:

  def <function name>(<parameters>):

  <Sentence group (including return)>

  return <variable/value>

  e.g.

```
def plus(a, b):
    return a + b
def print_plus(a, b):
    print(a + b)
```

- Function call:

  form:

  <name>(<parameters>)

  e.g.

  plus(2, 3)

Predefined Function

function	operation
print()	output
type(XXX)	get the type of XXX
len(XXX)	get the length of XXX
range(a, b, c = 1)	return a list from **a** to **b** , step length = c

Class

Class is an object, includes a group of functions and variables (attributes)

form:

1. defination

```
class <name>():
    def __init__(self, <other parameters>):
        <Sentence group> # these sentences will be done when initialized

    def <name>(self, <other parameters>): ...
```

2. initialize

```
<varaible name> = <class name>
```

Class

Class is an object, includes a group of functions and variables (attributes)

3. adding attributes: in any function of the class:

self.<attribute name> = <initial value>

4. using attributes:

4.1. in any function of the class

self.<attribute name>

4.2. out of the class

<class name>.<attribute name>

```
class Quadratic_Equation(): # define a class for the solution
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
    def cal_delta(self):
        self.delta = self.b ** 2 - 4 * self.a * self.c
    def cal_root(self):
        self.root1 = None
        self.root2 = None
        self.cal_delta()
        if self.delta > 0:
            self.root1 = (self.delta ** (1/2) - self.b) / (2 * self.a)
            self.root2 = (-self.delta ** (1/2) - self.b) / (2 * self.a)
```

$aX^2 + bX + c = 0$

if a=1, b=4, c=3,

Calculate root of X

```
eq = Quadratic_Equation(1, 4 ,3) # initialize the class


eq.cal_delta() # calculate delta

print(eq.delta)

eq.cal_root() # calculate the roots

print(eq.root1)

print(eq.root2)
```

$aX^2 + bX + c = 0$

if a=1, b=4, c=3,

Calculate root of X