目录

Part I

文本类

Chapter 1

文本类

“我现在年纪大了，搞了这么多年软件，错误不知犯了多少，现在觉悟了。我想，假设我早年要是在数理逻辑上好好的下点功夫的话，我就不会犯这么多的错误。不少东西逻辑学家早就说了，可我不知道。要是我能年轻 *20* 岁的话就要回去学逻辑。’我看这是经验之谈，说明搞技术没有理论指导是不行的，而我们研究思维科学要从哲学的逻辑学吸收营养。” (Dijkstra)

Part II

数学类

Chapter 2

数学字体

$mathrm\{ABCxyz1234\}$ →ABCxyz1234

$mathit\{ABCxyz1234\}$ →*ABCxyz1234*

$mathtt\{ABCxyz1234\}$ →ABCxyz1234

$mathsf\{ABCxyz1234\}$ →ABCxyz1234

$mathbf\{ABCxyz1234\}$ →**ABCxyz1234**

$mathcal\{ABCxyz1234\}$ →$\mathcal{ABC}xyz1234$

Chapter 3

数理逻辑

对于两个公式 A, B，当“$A \vDash B$ 且 $B \vDash A$”时，写作：

$$A \models\models B$$

此时称公式 A 和 B“逻辑等值（**tautologically equivalent**）的（简称为等值公式 **(equivalent)**）”。两个公式为等值公式，当且仅当对这两个公式的任何真假赋值都得到相同的真假值。

Chapter 4

数学公式

4.1 积分的定义

我们将通过三个步骤定义可测函数的积分。首先定义非负简单函数的积分。以下设 E 是 $\mathscr{R}^n$ 中的可测集。

定义 4.1.1 (可积性)

设 $f(x)=\sum_{i=1}^{k} a_i\chi_{A_i}(x)$ 是 E 上的非负简单函数，中文其中 $\{A_1,A_2,\dots,A_k\}$ 是 E 上的一个可测分割，$a_1,a_2,\dots,a_k$ 是非负实数。定义 f 在 E 上的积分为 $\int_a^b f(x)$

$$\int_E f\,dx=\sum_{i=1}^{k} a_i m(A_i)\pi\alpha\beta\sigma\gamma\nu\xi\epsilon\epsilon.\oint_a^b\oint\prod_{i=1}^{n} \tag{4.1}$$

一般情况下 $0\le\int_E f\,dx\le\infty$。若 $\int_E f\,dx<\infty$，则称 f 在 E 上可积。

一个自然的问题是，Lebesgue 积分与我们所熟悉的 Riemann 积分有什么联系和区别？在 4.4 在我们将详细讨论 Riemann 积分与 Lebesgue 积分的关系。这里只看一个简单的例子。设 $D(x)$ 是区间 $[0,1]$ 上的 Dirichlet 函数。即 $D(x)=\chi_{Q_0}(x)$，其中 Q_0 表示 $[0,1]$ 中的有理数的全体。根据非负简单函数积分的定义，$D(x)$ 在 $[0,1]$ 上的 Lebesgue 积分为

$$\int_0^1 D(x)dx=\int_0^1 \chi_{Q_0}(x)dx=m(Q_0)=0 \tag{4.2}$$

即 $D(x)$ 在 $[0,1]$ 上是 Lebesgue 可积的并且积分值为零。但 $D(x)$ 在 $[0,1]$ 上不是 Riemann 可积的。

有界变差函数是与单调函数有密切联系的一类函数。有界变差函数可以表示为两个单调递增函数之差。与单调函数一样，有界变差函数几乎处处可导。与单调函数不同，有界变差函数类对线性运算是封闭的，它们构成一线空间。练习题 4.1 是一个性质的证明。

练习 **4.1** 设 $f\notin\in L(\mathscr{R}^1)$，$g$ 是 $\mathscr{R}^1$ 上的有界可测函数。证明函数

$$I(t)=\int_{\mathscr{R}^1} f(x+t)g(x)dx\quad t\in\mathscr{R}^1 \tag{4.3}$$

是 $\mathscr{R}^1$ 上的连续函数。

解 即 $D(x)$ 在 $[0,1]$ 上是 *Lebesgue* 可积的并且积分值为零。但 $D(x)$ 在 $[0,1]$ 上不是 *Riemann* 可积的。 □

证明 即 $D(x)$ 在 $[0,1]$ 上是 *Lebesgue* 可积的并且积分值为零。但 $D(x)$ 在 $[0,1]$ 上不是 *Riemann* 可积的。 □

定理 4.1.2 (Fubini 定理)

(1) 若 $f(x,y)$ 是 $\mathscr{R}^p \times \mathscr{R}^q$ 上的非负可测函数，则对几乎处处的 $x \in \mathscr{R}^p$，$f(x,y)$ 作为 y 的函数是 $\mathscr{R}^q$ 上的非负可测函数，$g(x) = \int_{\mathscr{R}^q} f(x,y)dy$ 是 $\mathscr{R}^p$ 上的非负可测函数。并且

$$\int_{\mathscr{R}^p \times \mathscr{R}^q} f(x,y)dxdy = \int_{\mathscr{R}^p} \left(\int_{\mathscr{R}^q} f(x,y)dy \right) dx. \tag{4.4}$$

(2) 若 $f(x,y)$ 是 $\mathscr{R}^p \times \mathscr{R}^q$ 上的可积函数，则对几乎处处的 $x \in \mathscr{R}^p$，$f(x,y)$ 作为 y 的函数是 $\mathscr{R}^q$ 上的可积函数，并且 $g(x) = \int_{\mathscr{R}^q} f(x,y)dy$ 是 $\mathscr{R}^p$ 上的可积函数。而且 (4.4) 成立。

笔记

在本模板中，引理（lemma），推论（corollary）的样式和定理 4.1.2 的样式一致，包括颜色，仅仅只有计数器的设置不一样。

我们说一个实变或者复变量的实值或者复值函数是在区间上平方可积的，如果其绝对值的平方在该区间上的积分是有限的。所有在勒贝格积分意义下平方可积的可测函数构成一个希尔伯特空间，也就是所谓的 L^2 空间，几乎处处相等的函数归为同一等价类。形式上，L^2 是平方可积函数的空间和几乎处处为 0 的函数空间的商空间。

命题 4.1.3 (最优性原理)

如果 u^* 在 $[s,T]$ 上为最优解，则 u^* 在 $[s,T]$ 任意子区间都是最优解，假设区间为 $[t_0,t_1]$ 的最优解为 u^*，则 $u(t_0) = u^*(t_0)$，即初始条件必须还是在 u^* 上。

我们知道最小二乘法可以用来处理一组数据，可以从一组测定的数据中寻求变量之间的依赖关系，这种函数关系称为经验公式。本课题将介绍最小二乘法的精确定义及如何寻求点与点之间近似成线性关系时的经验公式。假定实验测得变量之间的 n 个数据，则在平面上，可以得到 n 个点，这种图形称为“散点图”，从图中可以粗略看出这些点大致散落在某直线近旁，我们认为其近似为一线性函数，下面介绍求解步骤。

以最简单的一元线性模型来解释最小二乘法。什么是一元线性模型呢？监督学习中，如果预测的变量是离散的，我们称其为分类（如决策树，支持向量机等），如果预测的变量是连续的，我们称其为回归。回归分析中，如果只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，则称为多元线性回归分析。对于二维空间线性是一条直线；对于三维空间线性是一个平面，对于多维空间线性是一个超平面。

性质 4.1.4

柯西列的性质

1. $\{x_k\}$ 是柯西列，则其子列 $\{x_k^i\}$ 也是柯西列。
2. $x_k \in \mathscr{R}^n$，$\rho(x,y)$ 是欧几里得空间，则柯西列收敛，$(\mathscr{R}^n, \rho)$ 空间是完备的。

结论 回归分析（*regression analysis*）是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。运用十分广泛，回归分析按照涉及的变量的多少，分为一元回归和多元回归分析；按照因变量的多少，可分为简单回归分析和多重回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。

Chapter 5

关系代数

定义 **5.0.1**

$$A \bowtie BA \bowtie B \tag{5.1}$$

Part III

代码类

Chapter 6

代码

Listing 6.1: 整数乘法的分而治之算法实现

```
function multiply(x, y)
Input: Positive integers x and y, in binary
Output: Their product

n = max(size of x, size of y)
if n = 1: return xy

x_L, x_R = leftmost⌈n/2⌉, rightmost⌈n/2⌉ bits of x
y_L, y_R = leftmost⌈n/2⌉, rightmost⌈n/2⌉ bits of y

P_1 = multiply(x_L, y_L)
P_2 = multiply(x_R, y_R)
P_3 = multiply(x_L + x_R, y_L + y_R)

return P_1 × 2^n + (P_3 − P_1 − P_2) × 2^(n/2) + P_2
```

Listing 6.2: 最长回文子序列解法一

```
class Solution:
    def longestPalindromeSubseq(self, s):
        n = len(s)
        if n == 1:
            return 1
        elif n == 2:
            if s[0] == s[1]:
                return 2
            return 1
        # 初始化dp数组
        dp = [[0] * n for _ in range(n)]
        dp[0][0] = 1
        dp[1][1] = 1
        if s[0] == s[1]:
            dp[0][1] = 2
        else:
            dp[0][1] = 1

        for i in range(2, n):
            # 每个单个字符都是长度为1的回文子序列
            dp[i][i] = 1
            #for j in range(0, i):
            for j in range(i - 1, -1, -1):
                if s[i] == s[j]:
                    if j == i - 1:
                        dp[j][i] = 2
                    else:
                        dp[j][i] = 2 + dp[j + 1][i - 1]
                else:
                    dp[j][i] = max(dp[j+1][i], dp[j][i - 1])

        return dp[0][n - 1]
```

Chapter 7

算法

7.1 算法效果

Algorithm 1 An algorithm with caption

Require: $n \geq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  $X \leftarrow x$
  $N \leftarrow n$
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow \frac{N}{2}$ ▷ This is a comment
    else if N is odd **then**
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$

Algorithm 2 带行号和 input、output 的算法示意

Input
Output
1: **procedure** MyProcedure
2: *stringlen* ← *length of string*
3: $i \leftarrow patlen$
4: **top**:
5: **if** $i >$ *stringlen* **then return** false
6: $j \leftarrow patlen$
7: **loop**:
8: **if** $string(i) = path(j)$ **then**
9: $j \leftarrow j - 1$.
10: $i \leftarrow i - 1$.
11: **goto loop**.
12: **close**;
13: $i \leftarrow i + \max(delta_1(string(i)), delta_2(j))$.
14: **goto top**.

Algorithm 3 Iterative Policy Evaluation Algorithm

Require: π, the policy to be evaluated

Initilize $V(s) = 0$, for all $s \in \mathcal{S}^+$
repeat
  for each $s \in \mathcal{S}$
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V(s')$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Ensure: $V \approx V^\pi$