第1章 算法设计与分析基础

《人工智能算法》

清华大学出版社
2022年7月

提纲

- 概述
- 算法的基本概念
- 算法效率分析
- 算法的最优、最坏和平均效率
- 算法运行时间估计
- 总结

概述 (1)

- **人工智能的三大基石：**数据，算法，算力；人工智能的本质是算法；算法的优劣决定了智能系统水平高低
- **算法对工程教育毕业要求的支撑：**

- **工程知识：**能够将数学、自然科学、工程基础和专业知识用于解决计算机领域的复杂工程问题。
- **设计/开发解决方案：**能够设计针对复杂工程问题的解决方案，设计满足特定需求的软件系统、模块/组件，并能够在设计环节中体现创新意识，考虑社会、健康、安全、法律、文化以及环境等因素。
- **研究：**能够基于计算机科学与工程的技术和方法对复杂工程问题进行分析与研究，包括设计实验、分析与解释数据、并通过信息综合得到合理有效的结论。

概述 (2)

- 学界与业界为实现同样的目标而努力
- 人们越来越客观地看待学界与业界研究工作的价值，学界与业界的对立逐渐消除、逐渐认可对方的价值
- 计算机科学的特点需要业界做科研、学界解决实际问题，算法助力克服技术瓶颈
- 学界与业界的合作成为常态，算法的价值得到双方认可

当代计算机专业人才工程能力 ← 算法“驾驶员”+“算法造车人”

算法设计与分析助力程序设计能力的提升、程序设计水平的提高

程序设计能力提升算法设计与分析水平

提纲

- 概述
- 算法的基本概念
- 算法效率分析
- 算法的最优、最坏和平均效率
- 算法运行时间估计
- 总结

算法的基本概念 (1)

- **算法：**解决问题的一步一步的方法
- **数据结构+算法=程序**

  - 有了好的算法和数据结构，以某种程序设计语言予以实现
  - 算法不依赖于特定程序语言，描述求解问题的通用的一般步骤

- **算法的定义与特点**

  算法是一系列解决问题的步骤；对于符合一定规范或约束的输入，能在有限时间内得到所要求的输出；用伪代码（Pseudocode）描述；特点：

  （1）**有穷性：**算法在有限时间内完成。

  （2）**确定性：**算法的每一步必须是确定的，不能有二义性的解释。

  （3）**可行性：**算法中的每一步必须是有意义的，且能达到预期目的。

  （4）**输入：**输入的值域必须仔细定义。

  （5）**输出：**得到问题的解。

  （6）同一问题可能存在几种不同的算法，执行效率也会有所差异。

算法的基本概念 (2)

算法的伪代码描述示例

算法 InsertionSort //插入排序

输入：

A[1: *n*]：待排序列表

输出：

按非降序排序后的列表 *A*

步骤：

```
1. For i=2 To n Do
2,      x←A[i]
3.      j←i−1
4,      While j>0 And A[j]<x Do
5.          A[j+1]←A[j]
6.          j←j−1
7.      End While
8.      A[j+1]←x
9. End For
10. Return A
```

提纲

- 概述
- 算法的基本概念
- 算法效率分析
- 算法的最优、最坏和平均效率
- 算法运行时间估计
- 总结

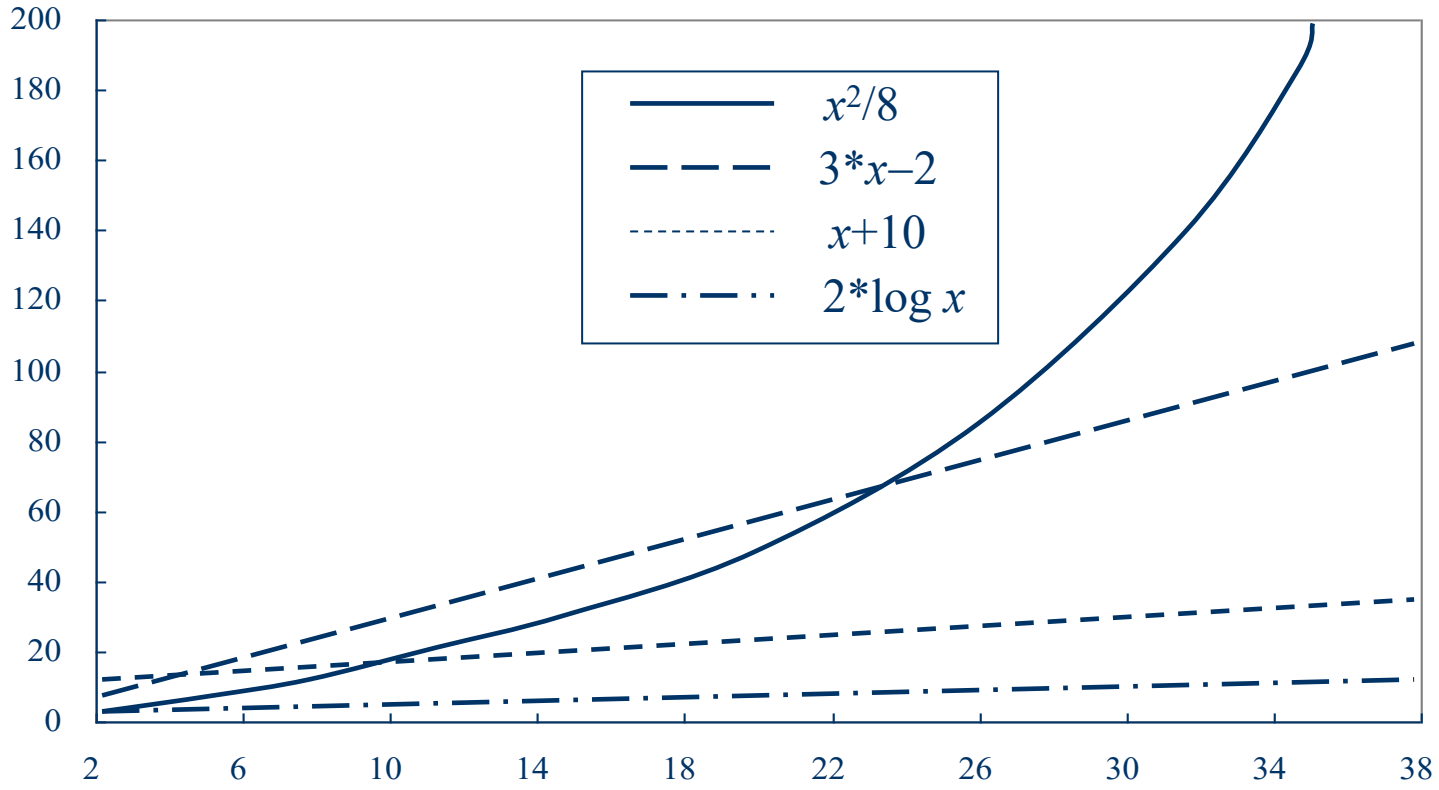
算法效率分析 (1)

- **效率：运行时间，存储空间**
- **计算时间**
  - 将操作的执行次数作为计算复杂度
  - 不依赖于程序运行软硬件环境和编程语言等因素且具有一般性的算法效率分析结果
  - 并不是实际执行的分和秒之类的时间（对于相同的运行环境有意义，但在不同处理器和内存等环境下并无意义）
- **增长率**
  - **基本操作：**算法中最重要（对算法运行时间的贡献最大）的操作
  - **关注：**随着输入规模的增加，算法执行时间变化的趋势
  - **讨论：**针对较大规模的输入，运行时间的增长率或增长的阶（Order）

基于渐进时间（增长率）对算法进行比较和分组

算法效率分析 (2)

小规模输入会掩盖算法效率的显著差异，因此需要考虑大规模输入



算法效率分析 (3)

- **2类重要操作**

- 比较操作（Comparison）

计算从数值计算发展到数据处理，比较是数据处理中最重要的操作之一

(1) 所有元素比较操作等价

(2) 搜索和排序算法中的基本操作

- 算术操作（Arithmetic）

(1) 加法操作（additive）：+, 一, 递增（increment）, 递减（decrement）

(2) 乘法操作（multiplication）：×, ÷, 取模（modulus）

(3) 算法分析中，加法操作和乘法操作分别考虑

算法效率分析 (4)

- **如何计算增长率？**

算法运行的渐进时间：去除了低阶项和首项系数后的算法运行时间函数，用渐进时间来表示算法的时间复杂度

✓ 对规模为n的输入，若算法运行时间为cn^2，随着n的增大，正常量c的作用逐渐降低；当与其他运行时间为dn^3的算法相比，常量c并没有多大作用

✓ 若算法运行时间为$n^2\log n+3n^2+5n$，n越大，低阶项$3n^2+5n$对算法效率影响越小

✓ 以上算法的运行时间是n^2阶、n^3阶和$n^2\log n$阶的

- **哪几类常见的增长率？**

✓ **多项式函数**（运行时间随着问题规模n的增加呈多项式增长）

- $\log n$、n、n^2和n^3，分别称为对数函数、线性函数、平方函数和立方函数

- n^c和$n^c\log n$（$0<c<1$）称为次线性函数，$n\log n$和$n^{1.5}$称为次平方函数

✓ **指数函数**（运行时间随着问题规模n的增加而爆炸性增长，例如2^n）

算法效率分析 (5)

- **渐进时间的符号**

(1) Ω 符号（Big Omega）

- Ω (*f*)：增长至少与*f*一样快的函数（**增长不比*f*慢，效率不比*f*对应算法高**）

令$f(n)$和$g(n)$是从自然数集到非负实数集的两个函数，若存在一个自然数n_0和一个正常数c，使得对所有的$n \geq n_0$，$f(n) \geq cg(n)$，则称$f(n)$为$\Omega(g(n))$，记为$f(n) \in \Omega(g(n))$或$f(n)=\Omega(g(n))$。

- 描述了一个运行时间的下界
- 若$\lim_{n\to\infty} \frac{f(n)}{g(n)}$存在，则$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$，蕴含着$f(n)=\Omega(g(n))$
- $f(n)$的增长至少与$g(n)$的某个常数倍一样快

算法效率分析 (6)

(2) *O* 符号（Big Oh）

- $O(f)$：增长不比f快的函数（**增长不比f快，效率不比f对应算法低**）

令$f(n)$和$g(n)$是从自然数集到非负实数集的两个函数，若存在一个自然数n_0和一个正常数c，使得对所有的$n \geq n_0$，$f(n) \leq cg(n)$，则称$f(n)$为$O(g(n))$，记为$f(n) \in O(g(n))$或$f(n)=O(g(n))$。

- 描述了一个运行时间的上界
- 若$\lim_{n\to\infty} \frac{f(n)}{g(n)}$存在，则$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq \infty$蕴含着$f(n)=O(g(n))$
- $f(n)$没有$g(n)$的某个常数倍增长得快

算法效率分析 (7)

(3) Θ符号

- $\Theta(f)$：增长与f一样快的函数

令$f(n)$和$g(n)$是从自然数集到非负实数集的两个函数，若存在一个自然数n_0和两个正常数c_1和c_2，使得对所有的$n \geq n_0$，$c_1 g(n) \leq f(n) \leq c_2 g(n)$，则称$f(n)$为$\Theta(g(n))$，记为$f(n) \in \Theta(g(n))$或$f(n)=\Theta(g(n))$。

- 若$\lim\limits_{n\to\infty} \frac{f(n)}{g(n)}$存在，则$\lim\limits_{n\to\infty} \frac{f(n)}{g(n)} = c$蕴含着$f(n)=\Theta(g(n))$，其中$c$为正常数

- $f(n)=\Theta(g(n))$当且仅当$f(n)=O(g(n))$且$f(n)=\Omega(g(n))$

例如：

若$f(n) = n + 2\sqrt{n}$，$g(n) = n^2$，则$f(n) = O(g(n))$，$g(n) = \Omega(f(n))$；

若$f(n) = n + \log n$，$g(n) = \sqrt{n}$，则$f(n) = \Omega(g(n))$，$g(n) = O(f(n))$；

若$f(n) = 10n^2 + 2n$，$g(n) = 30n^2$，则$f(n) = \Theta(g(n))$。

算法效率分析 (8)

利用极限比较增长次数

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\begin{cases}0: & f(n)=O(g(n)) \quad f(n)\text{比 } g(n)\text{ 增长慢}\\ c: & f(n)=\Theta(g(n)) \quad f(n)\text{ 与 } g(n)\text{ 增长相同}\\ \infty: & f(n)=\Omega(g(n)) \quad f(n)\text{比 } g(n)\text{ 增长快}\\ \text{不存在：该方法不适用}\end{cases}$$

(1) $f(n)=(n^2-n)/2$, $g(n)=6n$. $f(n)=O(g(n))$? $g(n)=O(f(n))$?

$$\lim\frac{g(n)}{f(n)}=\lim\frac{6n}{(n^2-n)/2}=0$$

$g(n)=O(f(n))$

(2) $f(n)=n^4+3$, $g(n)=n^5$. $f(n)=O(f(n))$? $g(n)=O(f(n))$?

$f(n)=O(g(n))$

算法效率分析 (9)

一些常用的结论：

- $\sum_{i=0}^{k} a_i n^i = \Theta(n^k)$
- $\log n^k = \Theta(\log n)$
- $\log n! = \sum_{i=1}^{n} \log i = \Theta(n \log n)$
- $\sum_{i=1}^{n} \frac{n}{i} = (n \log n)$

算法效率分析 (10)

渐进符号的有用特性

- $O(f)+O(g)=O(f+g)$

特性：某些算法是由两个（以上）执行部分组成，该算法的整体效率由具有较大增长率的部分决定，即它效率最差的部分

证明（根据O的定义证明）：

假设$F(n)=O(f)$, $G(n)=O(g)$

那么，存在c_1和n_1，使得$n \geq n_1$时，有$F(n) \leq c_1 f(n)$；

同理，存在c_2和n_2，使得$n \geq n_2$时，有$G(n) \leq c_2 g(n)$。

假设$c_3=\max\{c_1, c_2\}$，$n_3=\max\{n_1, n_2\}$，当$n \geq n_3$时有

$F(n)+ G(n) = O(f)+O(g) \leq c_1 f(n) + c_2 g(n) \leq c_3(f(n) + g(n))$，

即$O(f)+O(g) \leq c_3 (f(n) + g(n))$。

因此，$O(f)+O(g) = O(f+g)$。

- $O(f) \cdot O(g)=O(f \cdot g)$

提纲

- 概述
- 算法的基本概念
- 算法效率分析
- 算法的最优、最坏和平均效率
- 算法运行时间估计
- 总结

算法的最优、最坏和平均效率 (1)

算法的执行时间只与问题的规模有关、而与输入值无关

- **最优情况**
  - 当输入规模为n时算法的最短运行时间
  - 无法有效描述算法在一般情况下的时间复杂度，实际中一般不予考虑
- **最坏情况**
  - 当输入规模为n时算法的最长运行时间
  - 算法运行时间的上界
- **平均情况**
  - 所有规模为n的输入的平均运行时间
  - 实际上，考虑以计算时间为依据的不同输入类（计算时间意义上的等价类），计算所有不同输入类的平均运行时间

算法的最优、最坏和平均效率 (2)

顺序搜索算法的效率分析

- **假设**

  - 列表*list*[1· ·*n*]，无重复元素

  - 目标不在列表中，则返回0

- **算法**

```
SequentialSearch (list, target, n)
 for i=1 to n do
   if (target=list[i]) then
     return i
   end if
 end for
 return 0
```

- ***list***: 12, 5, 6, 3, 9, 10, 2, 11

  target: 10

- 搜索过程

12, 5, 6, 3, 9, 10, 2, 11

12, 5, 6, 3, 9, 10, 2, 11

12, 5, 6, 3, 9, 10, 2, 11

12, 5, 6, 3, 9, 10, 2, 11

12, 5, 6, 3, 9, 10, 2, 11

12, 5, 6, 3, 9, 10, 2, 11

算法的最优、最坏和平均效率 (3)

- **最坏情况分析**

- 2种情况：target与list中最后一个元素匹配；target不在list中
- target与list中的每一个元素进行比较
- 最多n次比较，时间复杂度为$O(n)$

- **平均情况分析**

(1) target总能成功找到（n个位置等概率）

第i个位置匹配，执行i次元素比较操作

$$A(n)=\frac{1}{n}\sum_{i=1}^{n}i=\frac{1}{n}*\frac{n*(n+1)}{2}=\frac{n+1}{2}$$

平均情况时间复杂度 $O(n)$

(2) target未必能成功找到

✓ 一共n+1种情况（target在list中：n种；target不在list中：1种）

✓ n+1种情况等概率，平均比较次数：

$$A(n)=\frac{1}{n+1}*\left[\left(\sum_{i=1}^{n}i\right)+n\right]$$

$$=\left(\frac{1}{n+1}\sum_{i=1}^{n}i\right)+\frac{n}{n+1}=\frac{n}{2}+\frac{n}{n+1}\approx\frac{n+2}{2}$$

提纲

- 概述
- 算法的基本概念
- 算法效率分析
- 算法的最优、最坏和平均效率
- 算法运行时间估计
- 总结

算法运行时间估计 (1)

非递归算法的效率分析步骤：

1、确定输入规模；

2、确定基本操作；

3、考虑基本操作的执行次数是否仅仅与输入规模有关，则按需要进行最优、最坏和平均效率分析；

4、建立基本操作执行次数与输入规模 n 的求和表达式，即增长率函数；

5、通过数学运算和公式化简，确定增长率。

算法运行时间估计 (2)

递归算法的效率分析步骤：

1、确定输入规模；

2、确定基本操作；

3、考虑基本操作的执行次数是否仅仅与输入规模有关。若还与其他因素有关，则按需要进行最优、最坏和平均效率分析;

4、建立基本操作数与规模的函数关系，即递推关系/递推式（Recurrence relation）和初始条件：

使用递推式描述递归算法的时间复杂度，通过求解递推式得到以n为自变量的闭合公式、从而估计递归算法的运行时间

5. 解递推式，确定增长率。

例如：

$$M(n) = M(n-1) + 1$$
$$= [M(n-2) + 1] + 1 = M(n-2) + 2$$
$$= [M(n-3) + 1] + 2 = M(n-3) + 3$$
$$= n$$

提纲

- 概述
- 算法的基本概念
- 算法效率分析
- 算法的最优、最坏和平均效率
- 算法运行时间估计
- 总结

总结

- 算法是计算机、人工智能等学科领域的灵魂
- 问题、算法的基本概念，算法与数据结构、程序的区别与联系
- 算法效率分析：渐进时间，增长率，增长率
- 渐进时间的符号和性质
- 算法的最优、最坏和平均效率分析
- 递归和非递归算法的运行时间估计

结语

谢谢！