AI 岗位基础面试问题

作者：孙峥
专业：计算机技术
邮箱：sunzheng2019@ia.ac.cn
学校：中国科学院大学 (中国科学院)
学院：人工智能学院 (自动化研究所)

2019 年 10 月 30 日

Part I
基础数学问题

Question 1

定义矩阵的范数：$\| A \|_2 = max_{x\neq 0}\frac{\|Ax\|_2}{\|x\|_2}$**，**$A$ **是对称正定阵，证明** $\| A \|_2 = \lambda(\lambda$ **是** A **的最大特征值)。**

证：{先说明一些相关的知识点：矩阵范数定义的时候，有非负性，绝对齐性，三角不等式，还比向量范数多一个相容性。然后引入矩阵的 F 范数，$\| A \|_F^2 = \sum_{i,j=1} a_{ij}^2 = tr(A^T A)$，可以验证矩阵的 F 范数是矩阵范数。再引入矩阵的 p 范数，$\| A \|_p = max_{x\neq 0}\frac{\|Ax\|_p}{\|x\|_p} = max_{\|x\|_p=1} \| Ax \|_p$，容易证明这样定义的也是矩阵范数。由于是向量的 p 范数导出的矩阵的 p 范数，所以此矩阵范数又称为算子范数 (《泛函分析》中有定义)。

上述说明的矩阵范数有以下两个重要性质：(1) 矩阵的 F 范数和 $2-$ 范数都与向量的 $2-$ 范数相容；(2) 所定义的算子范数，即 $p-$ 范数都与向量的 $p-$ 范数相容；(3) 任一矩阵范数，一定存在与之相容的向量范数。下面开始证明这道题，网上可以查找到的证明过程都非常复杂，需要 $A \geq B, A \leq B$, 然后导出 $A = B$ 的过程，此处提供一种相对简单的方法，是我在本科时候的《数值分析》课上由林丹老师讲授。}

假设 A 是一般矩阵，$A^T A$ 是对称半正定矩阵，则 $\exists$ 正交矩阵 $Q, s.t.$

$$A^T A = Q^T \Lambda Q, \Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n), \lambda_i \geq 0$$

且有：

$$\| A \|_2^2 = (Ax)^T (Ax) = x^T A^T A x = x^T Q^T \Lambda Q x = (Qx)^T \Lambda (Qx)$$

由于 Q 正交，且 $\| x \|_2 = 1$，有 $\| Qx \|_2 = 1$, 则：

$$\begin{aligned}
\| A \|_2^2 &= max_{\|x\|_2=1} \| Ax \|_2^2 \\
&= max_{\|x\|_2=1} (Qx)^T \Lambda (Qx) \\
&= max_{\|y\|_2=1} (y)^T \Lambda (y) \\
&= max_{\|y\|_2=1} \sum_{i=1}^{n} y_i^2 \lambda_i \\
&= \lambda_1
\end{aligned}$$

当 A 是对称正定阵时，特征值均大于 0。$A^T A$ 可以视为 $f(A)g(A)$，其特征值的最大值为 λ_1^2，λ_1 是 A 特征值的最大值, 证毕。

(1) 证明过程中用到了正交矩阵不改变向量或矩阵的 $2-$ 范数的性质。假设 P, Q 均为正交矩阵，则 $\| A \|_2 = \| PA \|_2 = \| AQ \|_2 = \| PAQ \|_2$;

(2) 除了矩阵的 $2-$ 范数，还有 $1-$ 范数和 ∞ 范数，计算结果可以用' 一列无穷行' 记忆。

Question 2

设 $X=\{x_1,x_2,...,x_n\},iid$ 服从 $U(0,k)$ 的均匀分布，求 k 的极大似然估计。
解：{求解极大似然估计，应该先写出极大似然函数 $ln(L(\theta))$，再对参数 θ 求导即可，必要时需要验证二阶导。}

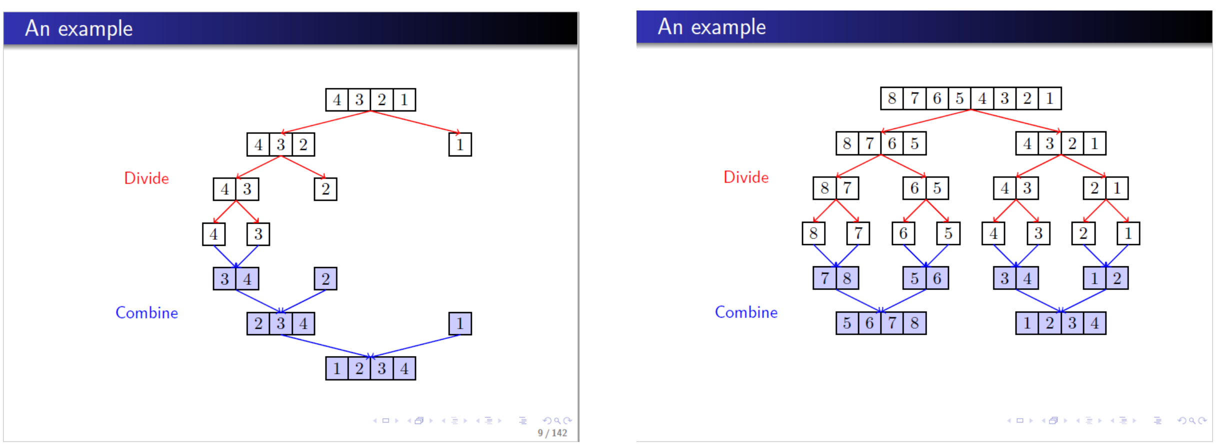
$$f(X)=\frac{1}{k^n},0\le x_i\le k.$$
$$lnL(k)=-nlnk,lnL(k)'=-\frac{n}{k}<0.$$

不存在 k 的极大似然估计。

Part II
计算机算法设计与分析

Question 1

用时间复杂度尽可能少的算法来排序一个 n 个整数的数组。
解：(1) 首先想到的是利用冒泡排序，利用两个 for 循环来排序数组，这种方法的时间复杂度是 $O(n^2)$，代码较简单, 没有递归调用，略去；
(2) 采用 DC(divide and conquer) 思想，每次递归调用数组 $[0,n]$ 的前 $n-1$ 个元素，再回溯合并，大致过程如下图所示 (选自卜东波老师上课的 slides)。



(a) 从倒数第二个元素位置一分为二　　(b) 从中间的位置一分为二

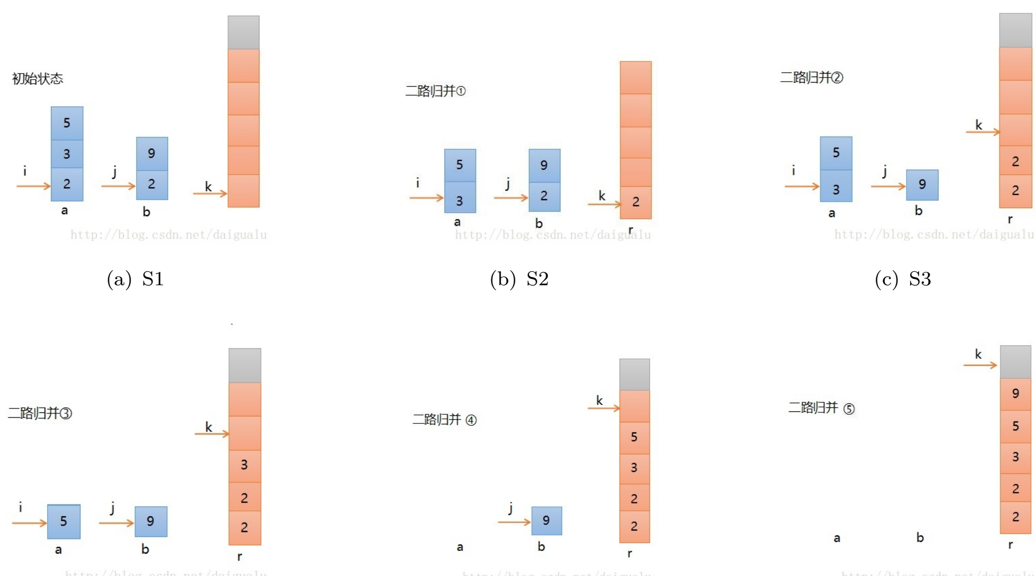
图 1:　采用分治思想排序

合并的时候将末尾的第 n 个元素插入前 $n-1$ 个元素当中，时间复杂度为 $O(n)$，所以有迭代式：$T(n)=T(n-1)+O(n)$，简单推导：

$$
\begin{aligned}
T(n) &\le T(n-1) + cn \\
&\le T(n-2) + c(n-1) + cn \\
&\le ... \\
&\le c(1+2+3+...+n) \\
&= O(n^2)
\end{aligned}
$$

代码相对简单，略去；

(3) 和 (2) 中方法的分治一样，按照**下标**来分治，此时分治从该数组的中心位置一分为二，分别对两个子问题排序，分别排好序之后再回溯合并，大致过程如图所示 (选自卜东波老师上课的 slides)，这实际上就是归并排序 (二路归并)。归并的过程可以简单描述为：先准备一个数组，数组容量是两个子问题的规模之和，比较 $a[i]$ 和 $b[j]$ 的大小，若 $a[i] \le b[j]$，则将第一个有序表中的元素 $a[i]$ 复制到 $r[k]$ 中，并令 i 和 k 分别加上 1；否则将第二个有序表中的元素 $b[j]$ 复制到 $r[k]$ 中，并令 j 和 k 分别加上 1；如此循环下去，直到其中一个有序表取完；然后再将另一个有序表中剩余的元素复制到 r 中从下标 k 到最后的单元，大致过程如下 (参考 $https://blog.csdn.net/daigualu/article/details/78399168$)。介绍完方法，下面给出实际的可运行代码 (C++)，利用分治和归并排序的思想来排序某一数组，其中的数组规模和元素是自行输入，更加灵活。



(a) S1 (b) S2 (c) S3

(d) S4 (e) S5 (f) S6

图 2: 二路归并过程

```cpp

#include <iostream>
#include <stdio.h>

using namespace std;

long int merge(int a[], int left, int mid, int right,int b[])
{
  int i = mid;
```

```
    int j = right;
    int k = 0;
    while (i >= left && j >= mid+1)
    {
        if(a[i] > a[j])
        {
            b[k++] = a[i--];
        }
        else
        {
            b[k++] = a[j--];
        }
    }
    while (i >= left)
    {
        b[k++] = a[i--];
    }
    while (j >= mid+1)
    {
        b[k++] = a[j--];
    }
    for (i = 0; i < k; i++)
    {
        a[right - i] = b[i];
    }
}

long int solve(int a[],int left, int right,int b[])
{
    if(right > left)
    {
    int mid = (right+left) / 2;
    solve(a,left, mid,b);
    solve(a,mid + 1, right,b);
    merge(a,left, mid, right,b);
    }
}


int main()
{
    long int n;//数组维度
    scanf("%d", &n);
    int *a = new int[n];
    int *b = new int[n];
    for(long int i=0;i<n;i++)
    {
        scanf("%d", &a[i]);//scanf的速度要比cin的速度快
    }

    solve(a,0,n-1,b);//归并排序

    for(int i = 0;i<n;i++)
        cout<<a[i]<<' ';
    return 0;
}
```

Listing 1: 归并排序,C++

上述的代码过程中，两个子问题的归并实际上是从后向前的归并，下面给出从前向后的归并过程，二者本质一样。(但是不知道为什么下面这个代码无法完成排序？)

```
#include <iostream>
#include <stdio.h>
```

```cpp
using namespace std;

long int merge(int a[], int left, int mid, int right,int b[])
{
  int i = left;
  int j = mid+1;
  int k = 0;
  while (i <= mid && j <= right)
  {
  if(a[i] > a[j])
  {
    b[k++] = a[i++];
  }
  else
  {
    b[k++] = a[j++];
  }
  }
  while (i <= mid)
  {
    b[k++] = a[i++];
  }
  while (j <= right)
  {
    b[k++] = a[j++];
  }
  for (i = 0; i < k; i++)
  {
    a[right - i] = b[i];
  }
}

long int solve(int a[],int left, int right,int b[])
{
  if(right > left)
  {
    int mid = (right+left) / 2;
    solve(a,left, mid,b);
    solve(a,mid + 1, right,b);
    merge(a,left, mid, right,b);
  }
}


int main()
{
  long int n;//数组维度
  scanf("%d", &n);
  int *a = new int[n];
  int *b = new int[n];
  for(long int i=0;i<n;i++)
  {
    scanf("%d", &a[i]);//scanf的速度要比cin的速度快
  }

  solve(a,0,n-1,b);//归并排序

  for(int i = 0;i<n;i++)
    cout<<a[i]<<' ';
  return 0;
}
```

Question 2

$C++$ 中输入二维 (多维) 数组的方法。(这不是个具体的问题，只是为了面试要求手写代码的时候可参考)。

Question 3

利用问题 1 和问题 2 中的方法，来解决数组逆序数计算问题 (包括数组显著逆序数计算问题)。

Question 4

与第 1 题按照数组的下标来分治不同, 这道题按照数组的值来分治, 即选取 $pivot$ 来排序一个数组。

Question 5

计算分治问题时间复杂度的总结，主定理 (Master theorem))

Part III 机器学习与深度学习基础模型与算法

Question 1

手推 BP(Back Propagation) 算法。