目录

湖南工商大学 算法导论 课程教案

授课题目（教学章、节或主题） 课时安排: 2学时

第四讲：概率分析与随机算法 授课时间 :第四周

教学内容（包括基本内容、重点、难点）：

基本内容： （1）雇佣问题：问题的提出与算法分析；

（2）指示器随机变量；

（3）随机算法：算法时间复杂度分析;

（4）概率分析和指示器随机变量的进一步使用：生日悖论；球与箱子；特征序列；在线雇佣问题.

教学重点、难点：重点为概率分析与随机算法原理、算法设计与分析

教学媒体的选择：本章使用大数据分析软件Jupyter教学，Jupyter集课件、Python程序运行、HTML网页制作、Pdf文档生成、Latex文档编译于一身，是算法导论课程教学的最佳选择。

板书设计：黑板分为上下两块，第一块基本定义，推导证明以及例子放在第二块。第一块 整个课堂不擦洗，以便学生随时看到算法流程图以及基本算法理论等内容。

课程过程设计： （1）讲解基本算法理论； （2）举例说明； （3）程序设计与编译； （4）对本课堂进行总 结、讨论； （5）布置作业与实验报告

第四讲 概率分析与随机算法

一. 雇佣问题

考虑一个雇佣问题：你是一个老板，向猎头公司委托寻找一个秘书职位，猎头每天为你推荐一个应聘者，而你对他进行面试。你的目标是，任 用所有应聘者中资质最好的。但由于秘书职位不能空缺，在每次面试完后，都要立即给面试者结果，所以只要当天的面试者资质比现任秘书好，你就解雇现任的秘 书，而重新雇佣当天的应聘者。

1. 程序实现

面试n个人的伪代码：

```
HIRE_ASSISTANT(n) {
    best = 0; // candidate 0 is a least-qualified dummy candidate
    for i = 1 to n {
        interview candidate i;
        if candidate i is better than candidate best {
            best = i;
            hire candidate i;
        }
    }
}
```

Python代码实现：

In [1]:

```python
def hire_assistant(assList):
    n = len(assList)
    best = 0
    index = 0
    for i in range(n):
        value = assList[i].score
        if value > best:
            best = value
            index = i
    return assList[index]

class Assistant:
    def __init__(self,a_name="anonymous",value=0):
        self.name = a_name
        self.score = value

assList=[Assistant("xiaoming",12),Assistant("zhonghou",13),Assistant("yuanlian
         Assistant("dapeng",10),Assistant("guomin",22),Assistant("lase",21)]
print("Assistant "+hire_assistant(assList).name +" is the best assitant")
```

```
Assistant guomin is the best assitant
```

2. 算法分析：

现在来分析一下面试过程中的花费。这里我们不是分析运行时间，而是花费，但本质是一样的——分析代码执行的代价。设每次进行面试的花费为 c_i，而雇佣一个新秘书的的花费为c_h。c_i 的花费比较少，而c_h的花费很高，因为雇佣新的秘书要给猎头一笔佣金，同时解雇现任秘书也需要花费。 假设期间我们雇佣过m个人，则上面算法的总花费为 $O(c_i * n + c_h * m)$。进行面试的花费是固定的，为 $c_i * n$，所以我们关注于雇佣的花费，而雇佣的花费取决于雇佣的次数。在最坏情况下，每天到来的面试者资质都比前一天的好，则每天都要雇佣新的秘书，总花费为 $O((c_i + c_h) * n)$。我们的算法依赖于面试者到来的顺序，但我们不能预期也不能改变这个顺序，所以我们预期一个一般或平均情况，这就需要对面试者的到来顺序进行概率分析。

3. 概率分析：

事实上，我们既不能得知应聘者出现的顺序，也不能控制这个顺序，因此我们使用概率分析。概率分析就是在问题的分析中使用概率技术。为了使用概率分析，必须使用关于输入分布的知识或者对其做假设，然后分析算法，计算出一个期望的运行时间。这个期望值通过对所有可能的输入分布算出。

有些问题，我们对所有可能的输入集合做某种假设。对于其他问题，可能无法描述一个合理的输入分布，此时就不能使用概率分析方法。

在雇佣问题中，可以假设应聘者是以随机顺序出现的。假设可以对任何两个应聘者进行比较并确定哪个更优；换言之，在所有的应聘者之间存在这一个全序关系。因此可以使用从1到n的唯一号码来标志应聘者的优秀程度。用rank(i)来表示应聘者i的名次。这个有序序列<rank(1),rank(2)，...,rank(n)>是序列<1,2,...,n>的一个排列。说应聘者以随机的顺序出现，就等于说这个排名列表是 1 到 n 的 n！中排列中的任何一个，每种都以相等的概率出现。

4. 随机算法

在许多情况下，我们对输入分布知识知之甚少；即使知道关于输入分布的某些信息，也无法对这种分布建立模型。然而通过使一个算法中的某些部分的行为随机化，就常常可以利用概率和随机性作为算法设计和分析的工具。

比如在雇佣问题中，如果雇佣代理给我们一份应聘者的名单，每天我们随机地挑选一个应聘者进行面试，从而确保了应聘序列的随机性。

更一般地，如果一个算法的行为不只有输入决定，同时也由随机数生成器所产生的数值决定，则称这个算法是随机的。

二. 指示器随机变量

给定一个样本空间S和事件A，那么事件A对应的指示器变量 $I\{A\}$ 的定义为：

$$I\{A\} = \begin{cases} 1 & \text{如果}A\text{发生的话} \\ 0 & \text{如果}A\text{不发生的话} \end{cases}$$

比如抛一枚均匀硬币，样本空间为 $S = H, T$ （H为正面朝上，T为背面朝上），正反面朝上的概率都分别为1/2，即 $Pr\{H\} = Pr\{T\} = 1/2$。我们用指示器随机变量 X_H 来对应正面朝上的情况，则：

$$X_H = I\{H\} \begin{cases} = 1 & \text{如果}H\text{发生，即正面朝上} \\ = 0 & \text{如果}T\text{发生，即背面朝上} \end{cases}$$

我们可以计算抛一次硬币时指示器随机变量 X_H 的期望值：

$$E[X_H] = E[IH] = 1 * Pr\{H\} + 0 * Pr\{T\} = 1 * 1/2 + 0 * 1/2 = 1/2$$

不难发现，指示器随机变量的期望值等于对应事件发生的概率。现在连续抛硬币 n 次，假设随机变量 X_i 对应第 i 次抛硬币时正面朝上的事件：

$$X_i = I\{\text{第}i\text{次抛硬币的正面朝上}\}$$

我们用随机变量X来对应 n 次抛硬币中正面朝上的总次数:

$$X = \sum_{i=1}^{n} X_i$$

则正面朝上的期望次数为:

$$E[X] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} 1/2 = n/2$$

现在回到刚才的雇佣问题，我们用指示器随机变量来分析花费：假设vX_i 对应事件A“第i个应聘者被雇佣”:

$$X_i = IA \begin{cases} = 1 & \text{如果应聘者}i\text{被雇佣} \\ = 0 & \text{如果应聘者}i\text{没有被雇佣} \end{cases}$$

事件A的概率为：应聘者是1 到 i 中最好的概率是 $1/i$，所以 X_i 值的期望值也为 $1/i$。用随机变量X表示雇佣的总次数:

$$X = X_1 + X_2 + \cdots + X_n$$

则X的期望值为:

$$E[X] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \frac{1}{i} = \ln(n) + O(1)$$

综上所述，在应聘者以随机的次序出现时，面试n个人后平均雇佣的人数为 $\ln(n) + O(1)$，而HIRE_ASSISTANCE总的雇佣费用为 $O(c_h * \ln(n))$。期望的雇佣费用比最坏情况下的雇佣费用 $O(c_h * n)$有了显著改善。

三. 随机算法

前面我们的概率分析是基于前提：应聘者到来的顺序是随机分布的。可以看到输入的随机化可以保证我们算法的一个期望值，所以随机算法（先将输入序列随机排列再进行计算）有比较好的平均效率。比如用随机算法重新考虑雇佣问题:

伪代码:

```
RANDOMIZED_HIRE_ASSISTANT(n) {
    randomly permute the list of candidates
    HIRE_ASSISTANT(n)
}
```

随机算法Python程序

In [ ]:

```python
# _*_coding:utf-8_*_
import pylab as pl
import random
assist = 0
cost = 0
cost_list = []
for tr in range(50):
    peoples = random.sample(range(50), 50)
    for people in peoples:
        cost += 20
        if assist < people:
            assist = people
            cost += 40
            print(assist,end=" ")
    print("")
    cost_list.append(cost)
    cost = 0
    assist = 0
min_cost = cost_list[0]
max_cost = cost_list[0]
list_y = range(1, 51)
print(min(cost_list), max(cost_list))
pl.plot(list_y, cost_list, 'r')
pl.xlim(0.0, 50)
pl.show()
```

In [ ]:

```
import random
list1=[1,2,3,4,5,6,7,8,9,10]
random.shuffle(list1)#对列表进行随机排序
print(list1)


def permute_by_sorting(a):
    permute_list=[]
    n=len(a)
    for i in range(0,n):
        permute_list.append(random.randint(1,n**3))
    print(permute_list)
    for i in range(0,n):
        for j in range(i+1,n):
            if permute_list[j]<permute_list[i]:
                permute_list[i],permute_list[j]=permute_list[j],permute_li
                a[i],a[j]=a[j],a[i]

def randomize_in_place(a):
    n = len(a)
    for i in range(n):
        j = random.randint(i,n-1)  #j=random.randint(1,n-1)是错误的
        a[i],a[j]=a[j],a[i]
    return a
```

四. 概率分析和指示器随机变量的进一步使用

1. 生日悖论

一个房间里面的人数要达到多少，才能使有至少两个人生日相同的概率达到1/2。

- 分析1：假设一年的天数为Y天，对房间里的所有人进行编号 $m_1, m_2, \cdots, m_n$。假设事件

$$A_i = \{m_i\text{与}m1, \cdots, mi - 1\text{的生日不同}\}, \qquad B_i = \{m_1 \cdots m_i\text{生日互不相同}$$

于是 $B_k = \cap_{i=1}^{k} A_i$ $

$$\begin{aligned} PrB_k &= Pr\{B_{k-1} \cap A_k\} = Pr\{B_{k-1}\} * Pr\{A_k | B_{k-1}\} \\ &= Pr\{B_{k-1}\} * (Y - k + 1)/Y \end{aligned}$$

依据递推式

$$Pr\{B_n\} = \prod_{i=1}^{n} (1 - (i-1)/Y)$$

求使 $Pr\{B_n\} < 1/2$ 的最小 n 值。

- 分析2：利用指示器随机变量,令指示器变量 $X_{ij} 对应于事件 {人i和人j生日相同}

$$E[X_{ij}] = 1/Y \;;\; \sum E[X_{ij}] = 1/Y * k(k-1)/2;$$

随机变量 $X = \sum X_{ij}$ 表示生日相同的两人对的对数； 如果取Y=356天，则只要k>=28,就可以期望至少有两个人生日。 注意：依据期望来反推概率只是简单而近似的分析，并不如分析1准确。不过虽然两种分析方法的结果不一致，但是都是$\Theta(n1/2)$。

生日悖论问题Python代码：

In [6]:

```
from random import randint

def list_birth():
    list_birth=[]
    for i in range(23):
        x=randint(1,12)
        if x in [1,3,5,7,8,10,12]:
            y=randint(1,31)
        elif x==2:
            y=randint(1,28)
        else:
            y=randint(1,30)
        list_birth.append((x,y))
    return list_birth
#建立生日日期对的列表，这里也可以用365天来算

def set_birth():
    set_birth=set(list_birth())
    return set_birth
#转换成集合是为了去重，若集合长度小于23，则说明，有生日一样的日期被去掉了

def main(count):
    n=0
    for i in range(count):
        list_birth()
        set_birth()
        if len(set_birth())<23:n=n+1
    chance=n/count #计算去重的次数占总循环次数的比例，即概率
    print("{:.2%}".format(chance))

main(10000)
```

```
50.66%
```

2. 球与箱子

投球符合几何分布，第i-1次命中到第i次命中之间的投球次数的期望为$1/((b-i+1)/b)=b/(b-i+1)$，则每个箱子中至少有一个球的期望为1到b的和为$b(ln(b)+O(1))$。

3. 特征序列

共n次实验，连续出现k次正面向上的期望值为$X(k)$,每种可能的概率为从i到i+k < n出现正面向上的概率，i的值从1到n-k+1，连续出现k次的概率为$1/(2^k)$,则期望x为这些概率的和为$(n-k+1)/(2^k)$。

特征序列即：使得期望达到最大时k的值，转化为求最大值问题，当 $k = \lg(n)$ 时最大。

4. 在线雇佣问题

面试n个人，抛弃前k个人，雇佣k+1到n中第一个比之前都优秀的人其位置为i。在位置i雇佣到最佳人选的期望值 $X(k)$，每一个可能的i为k+1到n，最佳人选在位置i被雇佣的概率由最佳人选排在位置i概率为1/n，且k+1到i-1之间不会有人被聘用即其中的最大值只在1到k之间概率为k/(i-1)。两个概率之积，并由k+1到n的和的期望达到最大值时k的值即为问题的解

引用及参考：

[1]《Python数据结构与算法分析》布拉德利.米勒等著，人民邮电出版社，2019年9月.
[2] https://blog.csdn.net/zhang_xiaomeng/article/details/71425008 (https://blog.csdn.net/zhang_xiaomeng/article/details/71425008)
[3] http://blog.chinaunix.net/uid-26947004-id-3198016.html (http://blog.chinaunix.net/uid-26947004-id-3198016.html)
[4] https://blog.csdn.net/longhuihu/article/details/5864442 (https://blog.csdn.net/longhuihu/article/details/5864442)

课后练习

1. 写出雇佣问题的完整Python代码或 C语言代码。

2. 写出随机排列的完整Python代码或 C语言代码，并举例说明。

3. 写出生日悖论问题、球与箱子问题、在线雇佣问题的python代码或C语言代码。

讨论、思考题、作业：

参考资料（含参考书、文献等）：算法导论. Thomas H. Cormen等，机械工业出版社，2017.

授课类型（请打√）：理论课□ 讨论课□ 实验课□ 练习课□ 其他□

教学过程设计（请打√）：复习□ 授新课□ 安排讨论□ 布置作业□

教学方式（请打√）：讲授□ 讨论□ 示教□ 指导□ 其他□

教学资源（请打√）：多媒体□ 模型□ 实物□ 挂图□ 音像□ 其他□

填表说明：1、每项页面大小可自行添减；2、教学内容与讨论、思考题、作业部分可合二为一。