数据科学基础

Foundations of Data Science

2.3 随机测试示例

陈振宇
南京大学智能软件工程实验室
www.iselab.cn

随机算法测试

验证多项式恒等式的随机算法

- 问题背景
- 随机测试初步
- 随机测试改进

问题背景

假定有一个计算多项式乘法的程序。

程序可能采用左右两边的某一方式实现：

$$[(x+1)(x-2)(x+3)(x-4)(x+5)(x-6) = x^6 - 7x^3 + 25$$

- 如何验证左右两边的多项式相等？
- 我们假设一个超大规模的多项式。

问题背景

给定两个多项式，$F(x)$和$G(x)$，可以通过将它们变换成规范形式($\sum_{i=0}^{d} c_i x^i$)来验证它们是否恒等：

$$F(x) ? \equiv G(x)$$

两个多项式相等当且仅当他们的规范式中所有的对应系数相等。

- 基于此，我们假定$F(x)$为乘积$F(x) = \prod_{i=1}^{d}(x - a_i)$，$G(x)$为规范式，连续地将$F(x)$的第$i$个单项式与前面$i-1$个单项式相乘，如此把$F(x)$变换成规范式，需要做$O(d^2)$次系数相乘。

随机测试初步

设$F(x)$和$G(x)$的最高阶为d，随机算法首先是从$\{1，\cdots，100d\}$中均匀随机(等可能)地选择一个整数$r*$，然后计算$F(x)$和$G(x)$的值。

将出现以下两种情况：

- $F(r) \neq G(r)$：判定两个多项式不等；
- $F(r) = G(r)$：判定两个多项式相等。

**为了讨论简单，我们假设是整数多项式。*

随机测试初步

计算$F(r)$和$G(r)$需要$O(d)$时间，要快于计算$F(r)$的规范形式，但是这种算法可能给出错误的答案。

（1） 为什么?

（2） 给出错误答案的概率是多少呢?

随机测试初步

为什么会给出错误答案?

- 如果 $F(x) = G(x)$，那么对于任意的 r，$F(r) = G(r)$，算法检测结果正确。
- 如果 $F(x) \neq G(x)$，$F(r) \neq G(r)$，那么算法判定不等，算法检测结果正确。
- 如果$F(x) \neq G(x)$，$F(r) = G(r)$，那么算法检测结果错误。

检测的单边错误：

- 我们判定两个多项式不等，那么判定一定正确。
- 我们判定两个多项式相等，则判定不一定正确。

随机测试初步

给出错误答案的概率多少?

- 当r是方程$F(x)-G(x)=0$的根时，必然会出现错误结果。
- $F(x)-G(x)$的次数不高于d，由代数基本定理可知，$F(x)-G(x)=0$不可能多于d个根。
- 当$F(x)\neq G(x)$时，在$\{1,\ \cdots,\ 100d\}$范围内，不可能有多于d个值，使得$F(r)=G(r)$。
- 因为$\{1,\ \cdots,\ 100d\}=100d$，所以算法选取一个值并给出错误答案的概率机会不会大于$\frac{1}{100}$。

随机测试改进

如何改进算法正确率?

在更大的整数范围进行取值。比如在$\{1, \cdots, 1000d\}$中随机选择r进行算法检测，则错误答案的概率至多为$\frac{1}{1000}$。

随机测试改进

如何改进算法正确率?

重复多次地进行随机检测恒等式。

- 当有一次$F(r) \neq G(r)$时，算法停止，给出结果不等。
- 当所有都是$F(r) = G(r)$时，算法给出多项式相等。

◆有放回抽样

◆无放回抽样

随机测试改进

有放回抽样

事件E_1，…，E_k独立当且进当对于任意$I \subseteq [1, k]$，有

$$P(\cap_{i\in I} E_i) = \prod_{i\in I} P(E_i)$$

每次检测错误率不超过$\frac{1}{100}$，则k次有放回抽样的错误率为

$$P(E_1 \cap \cdots \cap E_k) = \prod_{i=1}^{k} P(E_i) \leq (\frac{1}{100})^k$$

算法错误率指数级降低。

随机测试改进

无放回抽样

在已知事件F发生的条件下，事件E也发生的条件概率为

$$P(E|F)=\frac{P(E\cap F)}{P(F)}$$

仅当$P(F)>0$时，条件概率$P(E|F)$有意义。

对于无放回抽样，我们有$P(E_1\cap\cdots\cap E_k)=P(E_k|E_1\cap\cdots\cap E_{k-1})P(E_1\cap\cdots\cap E_{k-1})$

重复推导可得$P(E_1\cap\cdots\cap E_k)=P(E_1)P(E_2|E_1)P(E_3|E_1\cap E_2)\cdots P(E_k|E_1\cap\cdots\cap E_{k-1})$

因为有

$$P(E_j|E_1\cap\cdots\cap E_{j-1})\leq\frac{d-(j-1)}{100d-(j-1)}$$

因此在$k\leq d$次迭代后，算法给出错误答案的概率为

$$P(E_j|E_1\cap\cdots\cap E_{j-1})\leq\prod_{j=1}^{k}\frac{d-(j-1)}{100d-(j-1)}\leq(\frac{1}{100})^k$$

当$j>1$时，$\frac{d-(j-1)}{100d-(j-1)}<\frac{1}{100}$，算法进一步改善。

更多详情，请咨询课程助教。

总结

- $F(x)$无放回抽样的准确率比有放回抽样高。
- 有放回抽样的算法实现通常比无放回抽样简单。
- 当$d+1$次无放回抽样后，能够确保算法的准确性。
  但算法复杂度提升到$O(d^2)$。

内容回顾

- 概率的定义

  三个条件

- 条件概率

$$P(A|B)=\frac{P(AB)}{P(B)}$$

- 乘法公式

$$P(A_1\cdots A_n)=P(A_1)P(A_2|A_1)\cdots P(A_n|A_1A_2\cdots A_{n-1})$$

- 全概率公式

$$P(A)=P(A|B_1)P(B_1)+\cdots+P(A|B_n)P(B_n)$$

- 贝叶斯公式

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$