杭州应用工程技术学院学报,第 13 卷第 1 期,2001 年 3 月
Journal of Hangzhou Institute of Applied Engineering
Vol.13 No.1, Mar. 2001

试题难度系数确定数学模型的建立与实现

林雪明
(宁波大学科技学院　宁波 315211)

摘　要　试题难度系数是试题库中的一个重要参数.作者以潜在倾向理论为基础,结合区分度算法,提出了一种试题难度系数数学模型的建立和实现方法.该方法能使难度系数在试题库应用过程中不断得到修正,使确定的难度系数具有客观性和科学性.

关键词　数学模型　难度系数　潜在倾向理论　区分度
中图分类号　G642.474

建立试题库管理系统是计算机辅助教学的一个重要环节,是实现考试规范化、科学化的有力工具,同时也是实现教考分离的重要手段.一份试卷的组成涉及题型、题量、试题知识点分布、考试时间和试题难度等因素.其中试题难度系数的确定直接影响到试题库管理系统的命题质量.因此,难度系数的确定是试题库管理系统建立的核心,是合理、科学地组卷的基础.传统的难度系数确定方法是:组卷教师根据教学大纲和教学过程中教师自身对教学质量的感觉确定一份试题的难度系数.尽管这种方法存在着一定的合理性,但忽略了一个根本的因素——难度系数是被测对象对试题难度的客观反映.作者以潜在倾向理论为基础,结合区分度算法,提出了一种试题难度系数数学模型的建立和实现方法.

1　数学模型的建立

将测试结果经过一定的数学模型处理后可以确定试题的难度系数,根据考试学理论,某一测试题的难度系数与该题的得分率成正比.设一套试卷有 M 道测试题,第 i 题的满分值为 C_i,有 N 个人参加测试,第 j 个人的第 i 题得分为 g_{ij},根据潜在倾向理论,则这套试卷各题的难度系数 b_i 可从下式得到:

$$b_i = \ln\left(\frac{C_i \times N}{\sum_{j=1}^{N} g_{ij}} - 1\right) \tag{1}$$

在测试过程中,还应考虑试题全部答对和全部答错这两种特殊情况.当试题全部答对时,(1)式中的 b_i 为 $-\infty$;当试题全部答错时,(1)式中的 b_i 为 $+\infty$.因此,在处理过程中,当

收稿日期:2000-10-20

$\left|\left\{\frac{C_i \times N}{\sum_{j=1}^{N} g_{ij}} - 1\right\}\right| < \varepsilon$(即 b_i 大于一定的值时),将这些试题从试题库中抽出,进行必要的修改后再加入到试题库中.

当然,(1) 式计算得到的难度系数的值还不能直接用于试题库,应将它们作归整处理,转换为 1 至 5 之间的值,转换公式如下:

$$d_i = 2 \times \frac{\max | b_j | + b_i}{\max | b_j |} + 1, \qquad j = 1,2,\cdots,m \tag{2}$$

从(1) 式可以看出,每道试题的难度系数取决于全体测试者的得分率,这道试题的得分率越高,难度系数就越低;这道试题的得分率越低,难度系数就越高.这样就存在一个问题:所得的试题难度与测试对象有关.这就是说,测试者能力高低不同,测试出来的难度系数就不同.当测试者群体的能力普遍较高时,则难度系数就较低,当测试者群体的能力普遍较低时,则难度系数就高.因此,这种难度系数的确定方法缺乏一定的客观性.

为了增强试题的难度系数客观性,还需反复测试确定.设试题经过 K 次测试后,其难度系数为 d_i^k,第 $K+1$ 次测试计算得到的难度系数为 d_i,d_i 对 d_i^k 的影响程度和本次测试的区分度有关.区分度采用"两端分组法" 进行计算,即将某一试题得分前 25% 的测试者划分为高分组,将某一试题得分的后 25% 划分为低分组,计算公式如下:

$$Q_i = \frac{H_i - L_i}{C_i} \tag{3}$$

(3) 式中 Q_i 是第 i 题的区分度,H_i 是高分组测试者第 i 题的得分平均值,L_i 是低分组测试者第 i 题的得分平均值.Q_i 的值越大说明该试题能区分出不同能力的测试者知识掌握的程度;反之则说明该试题不能区分不同能力的测试者的知识掌握程度.根据经验值,一道理想的试题的区分度应大于 0.4.将 0.4 作为典型值,则难度系数修正公式如下:

$$d_i^{k+1} = \frac{d_i \times \frac{Q_i}{0.4} + k \times d_i^k}{k + \frac{Q_i}{0.4}} \tag{4}$$

下面分析(4) 的合理性.

将(4) 式两边各减去 d_i^k,化简得:

$$d_i^{k+1} - d_i^k = \frac{d_i - d_i^k}{\frac{0.4}{Q_i} \times k + 1} \tag{5}$$

记

$$\lambda = \frac{1}{\frac{0.4}{Q_i} \times k + 1} \tag{6}$$

则

$$d_i^{k+1} - d_i^k = \lambda \times (d_i - d_i^k) \tag{7}$$

当新计算得到的难度系数等于原来的难度系数,即 $d_i = d_i^k$ 时,(7) 式两边都为 0,$d_i^{k+1} = d_i^k$,难度系数不变.

当新计算得到的难度系数大于原来的难度系数,即 $d_i > d_i^k$ 时,$d_i^{k+1} > d_i^k$ 难度系数的变化量 $d_i^{k+1} - d_i^k$ 是 λ 和 $d_i - d_i^k$ 的函数.由(3) 式可知:$0 < Q_i < 1$.当 $Q_i < 0.4$ 时,由(6) 式得:λ 的值较小,即新的难度系数的变化 $d_i - d_i^k$ 对难度系数的最后结果影响较小.当 $Q_i > 0.4$ 时,λ 的值较大,即新的难度系数的变化 $d_i - d_i^k$ 对难度系数的最后结果影响较大.这是合理的,因为当 Q_i 较小时,区分

度较差，测试结果不可靠，新的难度系数的变化不应对难度系数的结果有较大的影响；当 Q_i 较大时，区分度较好，测试结果可靠，新的难度系数的变化应对难度系数的结果有较大的影响；另外，测试次数较少时，即 k 较小时，λ 较大；反之，λ 较小．这说明随着测试次数的增多，难度系数将越来越稳定．

当新计算得到的难度系数小于原来的难度系数时，难度系数的变化量 $d_i^{k+1} - d_i^k$ 与前述情况方向相反，其它类同，在此不作赘述．最后，对于(4)式的结果，还需作归整处理，处理公式如(2)式．

2　算法实现

考虑系统设计的实用性和合理性，作者将试题的数据结构设置了 8 个属性，它们分别为：课程名称，试题编号，题型，试题描述，试题知识点，难度系数，已测试次数，试题答案．试题的数据结构描述如下：

```
STRUCT    question_DB
          {Char     course_name[30];
          Int       question_ID;
          Int       question_type;
          Struct    question_describe     description;
          Char      question_key[20];
          Int       question_difficulty;
          Int       tested_number;
          Struct    question_answer       answer;
          }test_question[];
```

上述数据结构中，course_name + question_ID 唯一对应一道试题，在一个试题库只有一门课程的情况下，仅 question_ID 一个字段就是一道试题的唯一编码，在后面叙述的算法中，都以该字段为关键字．试题描述 description 及试题答案 answer 两字段的内容较多，结构复杂，因此，应先分别定义 question_describe 和 question_answer 这两个字段的数据结构．

在计算难度系数的算法中，要对每位测试者的每道试题进行难度系数粗算、难度系数归整、试题区分度计算、难度系数修正等处理，因此，在测试结束后，首先生成一个包括难度系数算法必需的各项内容的数组，该数组元素的数据结构描述如下：

```
STRUCT test_result
          {Int      question_ID;
          Int       question_mark;          /*试题满分值*/
          Int       test_mark[N];           /*测试者该题的得分，其中 N 为测试者人数*/
          Int       question_difficulty;    /*试题难度系数，初值由教师设定*/
          Int       tested_number;          /*该试题已测试的次数，初值设定为 1*/
          Int       test_difference;        /*该字段保存试题的区分度*/
          Int       temp_difficulty;        /*该字段保存难度系数粗算结果*/
          }difficulty_calculat[M];
```

在以上定义的两个数据结构的基础上，试题难度系数的实现算法描述如下：

```
calculat_difficulty()
```

```
sort_test_mark(difficulty_calculat);
/*将数组 difficulty_calculat 按 test_mark 从高到低排序,为计算区分度作准备*/
for  (i=1; i<=M; i++)
     {calculat_test_difference(difficulty_calculat[i]);
     store_test_difference(difficulty_calculat[i]);
     }
/*计算各道试题的区分度,将它保存在数组 difficulty_calculat 对应的单元中*/
for  (i=1; i<=M; i++)
     {calculat_temp_difficulty(difficulty_calculat[i]);
     chang_to_one_five(difficulty_calculat[i]);
     }
/*计算各道试题的临时难度系数,然后作归整化处理,转换为1至5之间的值,并将它保
存在数组 difficulty_calculat 对应的单元中*/
     for  (i=1; i<=M; i++)
     {adjust_difficulty(difficulty_calculat[i]);
     chang_to_one_five(difficulty_calculat[i]);
     difficulty_calculat[i].tested_number++;
     }
/*修正各道试题的临时难度系数,然后作归整化处理,转换为1至5之间的值,并将它保
存在数组 difficulty_calculat 对应的单元中,测试次数加1*/
     store_result_to_question_DB(test_question,difficulty_calculat)
/*将计算结果保存到试题库中*/
}
```

3 结 语

本文提出的基于潜在倾向理论、并通过测试结果区分度修正的试题难度系数确定算法,能很好地解决试题的优化命题问题,使试题难度系数具有客观性和科学性.作者在网络考场系统开发过程中,应用这种算法,取得了满意的效果.特别是对于基于计算机网络的局域网考场和远程考场,考生人数多,考试次数频繁,这为难度系数的不断修正创造了非常有利的条件.对于一些能进行自动阅卷的试题库系统,难度系数的修正也可自动进行,使确定的试题难度系数更加可靠、稳定.

参 考 文 献

1 任爱华,武新利.题库建设的目标及数学模型.山东师大学报,1998,(4):

2 王文剑,曹焕光,李跃琴.一个通用试题库管理系统的设计与实现.系统工程理论与实践,1997,(12):

Foundation and realization of mathematics model for difficulty coefficient of test questions

Lin Xueming
(College of Science and Technology, Ningbo University, Ningbo 315211)

Abstract Difficulty coefficient is one of the most important parameter in test questions data-base. According to the latent trait theory and dipartite degree, this paper presents the method of the foundation and realization of the mathematics model for difficulty coefficient. Difficulty coefficient can be modified continously in application and get more objective and scientific.

Key words mathematics model　difficulty coefficient　latent trait theory　dipartite degree