

# Using Item Response Theory To Rate (Not Only) Programmers

Michal Forišek

`<forisek@dcs.fmph.uniba.sk>`

Katedra informatiky

Fakulta matematiky, fyziky a informatiky

Univerzita Komenského

Bratislava, Slovensko

10<sup>th</sup> of August, 2009

# Objectivity of ratings

## Common questions

... (not only) when organizing a programming contest:

- is the evaluation method objective (enough)?
- is the task difficulty appropriate?  
(and what does “appropriate” mean?)
- will the best ones really win?

## A usual goal of a contest

ranking = arranging the contestant into a linear order  
based on their skill levels

## ... and a common tool

rating = a numeric approximation of the skill level

# State of the art

## Bayesian rating systems

- Contestant's performance:  
a normally distributed random variable
- Rating: its estimated mean
- An incremental approach:  
current estimate + new result  $\rightarrow$  new estimate

## Examples

**Elo** Elo, 1978.

**Glicko** Glickman, 1999.

**TrueSkill** Herbrich, Graepel (Microsoft) 2006.

**TopCoder** TopCoder Inc. 2002–2008.

## Disadvantages of existing rating systems

- They can not take task difficulty into account.
- They can only predict relative performance.
- Models usually ignore that performance may change in time.
- Low attack resistance.

# What is IRT?

## Setting:

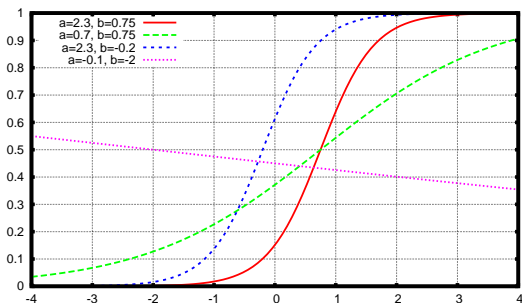
- people have latent abilities (e.g., problem solving)
- latent = we can not measure them directly
- goal of a testing theory: estimate them using tests

Item Response Theory: a modern testing theory  
(basics developed in '50s and '60s, only becomes used in '80s)

# Modelling task difficulty

Item Response Theory: 2-parameter logistic model

$$Pr(\theta, a, b) = \frac{1}{1 + e^{-a(\theta - b)}}$$



Good enough for “binary” programming tasks,  
and we managed to modify it for some types of partial scores.

## Basic principle of use

To estimate the parameters (task difficulty, contestant skill) we may use the maximum likelihood estimate.

A simple example:

We know task parameters  $a_i$ ,  $b_i$ , a response pattern  $s_i$ , we are estimating ability  $\theta$ .

$$L(\theta) = \prod_{i=1}^n Pr(\theta, a_i, b_i)^{s_i} \cdot (1 - Pr(\theta, a_i, b_i))^{1-s_i}$$

# Fisher information

## How to measure information?

Intuition: the reciprocal of the precision of the estimate

Formally: expected variance of the score

(score = partial derivative of the log-likelihood function)

$$\mathcal{I}(\theta) = E \left( \frac{\partial}{\partial \theta} \ln L(\theta, x) \right)^2 \quad (1)$$

## Importance for us

We are able to compute *the amount of information* the results of a given *contest* give us about a *contestant's skill estimate*.



# Overview

Some results of our research:

- The model fits real life data.
- It can be used to argue about task difficulty.
- Solving time in contests tends to have a log-normal distribution.
- Models can be compared via the predictions they make.
- Our model makes good predictions.

# Sanity check

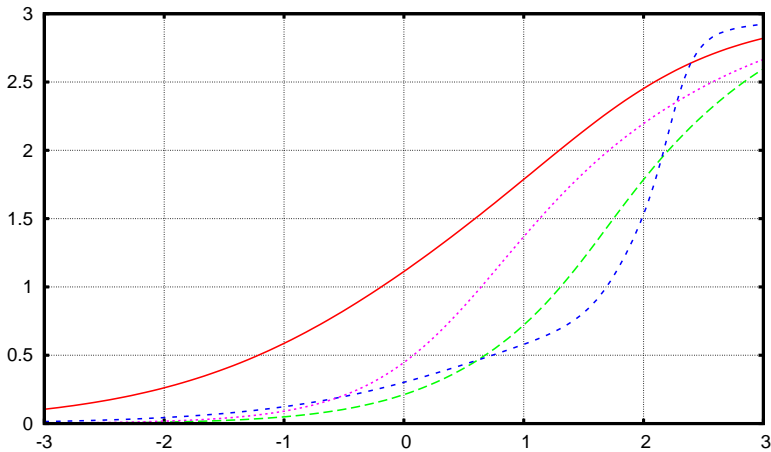
Corellation coefficients between the actual data  
and the best fitting model

(Slovak programming competitions: 300 contestants, 88 tasks)

$\rho$ range	year 2006/07	year 2007/08
[0.99, 1.00]	9	12
[0.98, 0.99)	6	5
[0.95, 0.98)	9	9
[0.90, 0.95)	7	7
[0.75, 0.90)	12	8
[0.00, 0.75)	1	2
[-1.00, 0.00)	0	1

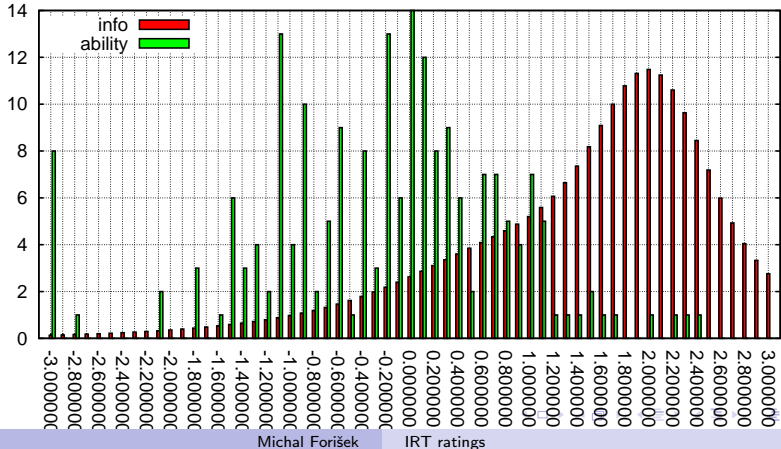
## Regional round tasks

Four tasks of different difficulty: skill vs. expected score



# Arguing about task difficulty

- skill estimate distribution (same data set)
- test information function for the regional round



## Comparing predictions

### Main observation

- You can not compare rating systems directly.
- Compare pairs (rating system, prediction algorithm)!

### What can we predict?

Tournament advancement:

We have  $N$  contestants and a tournament round,  $K$  best advance.  
For each contestant, predict the advancement probability.

### How to evaluate it?

Compare their likelihood given the actual result:

$$Q(p_1, \dots, p_N) = \sum_{i=1}^N s_i \log p_i + \sum_{i=1}^n (1 - s_i) \log(1 - p_i)$$

## Advantages of IRT based ratings

### Bayesian rating systems

... can only predict relative performance – we can estimate the probability that contestant  $A$  beats contestant  $B$ .

### IRT-based rating system

... can predict absolute performance – we can compute the expected number of tasks  $A$  will solve.

(or generalize: probability that  $A$  solves  $x$  tasks, expected score  $A$  gets, etc.)

## Example: TopCoder Open 2008

We tweaked our model to approximately match the TopCoder competition. Many difficulties:

- We completely ignore challenges.
- We have to take a weird scoring function into account.

Still, we got quite lovely predictions:

- for round 1 a slightly better prediction
- for round 2 a slightly worse prediction
- for round 1: we predicted 872.944 advancers, reality: 864/900
- for round 2: we predicted 300/300 advancers, happened

Thanks for your attention!

Questions, comments?