QFort Rating - An Introduction

1 QFort - A Derivation

Every question that a player has the opportunity to answer is quantified by a difficulty level d. One way to quantify a player's ability is the average difficulty $\bar{d}$ of all the questions they have answered successfully. Building on this idea, we define a player's ability (called the QFort rating) as the number $\mathcal{Q}$ such that they have a "high probability" of answering any question with difficulty $d \leq \mathcal{Q}$. Modeling the response as probabilistic allows us to account for the inevitable scenarios where a strong player will miss an easy question and vice versa.

Technically, we model each player's ability as a Gaussian normal variable $\mathcal{Q} \sim \mathcal{N}(\mu, \sigma^2)$ and assume the following data generation model. When a question is asked, the player answers it with a skill level drawn from this distribution. If that draw happens to be larger than the question difficulty d, that player would answer the question successfully. For each player, we treat μ and σ^2 as latent variables that are to be estimated given the difficulties of the N questions $\vec{\boldsymbol{d}} = (d_1, d_2, \ldots, d_N)$ that they had an opportunity to answer and their corresponding answers $\vec{\boldsymbol{a}} = (a_1, a_2, \ldots, a_N)$ where $a_i = 1$ if they answered the ith question correctly and 0 otherwise (including passes and wrong answers).

Given this set-up, the likelihood function can be written as

$$\begin{aligned}
P(\vec{\boldsymbol{a}}|\vec{\boldsymbol{d}}, \mu, \sigma^2) &= \prod_{i=1}^{N} P(a_i|d_i, \mu, \sigma^2) \\
P(a_i|d_i, \mu, \sigma^2) &= \begin{cases} P(\mathcal{N}(\mu, \sigma^2) \leq d_i, & a_i = 0 \\ P(\mathcal{N}(\mu, \sigma^2) \geq d_i, & a_i = 1 \end{cases} \\
&= \begin{cases} P(\mathcal{N}(0, 1) \leq \frac{d_i - \mu}{\sigma}, & a_i = 0 \\ P(\mathcal{N}(0, 1) \geq \frac{d_i - \mu}{\sigma}, & a_i = 1 \end{cases} \\
&= \begin{cases} \Phi(\frac{d_i - \mu}{\sigma}), & a_i = 0 \\ \left(1 - \Phi(\frac{d_i - \mu}{\sigma})\right), & a_i = 1 \end{cases} \\
&= (1 - a_i)\Phi\left(\frac{d_i - \mu}{\sigma}\right) + a_i\left(1 - \Phi\left(\frac{d_i - \mu}{\sigma}\right)\right)
\end{aligned}$$

where $\Phi(\cdot)$ is the CDF of the unit normal distribution.

For any given player, the QFort at the end of N questions is the value of μ (and σ^2) that maximizes this likelihood function. Note that in this expression, all a_i and d_i are known and this is just a simple optimization problem on two variables. While the functional form of this optimization is complicated, it can be easily solved using any numerical optimization library.

2 QFort - Extensions

This basic framework can be extended in many ways both to improve the underlying mathematical model and to better account for the dynamics of the MIMIR format.

2.1 Distinguishing Directs vs. Passes

The most straightforward extension is to give more weight to a player's direct questions (since players might be conservative and pass on a question that they are reasonably confident about if it is not their direct). This can be easily subsumed into the framework above by defining two different sets of parameters to be estimated - one pair of (μ, σ^2) for the directs and another for the passes. This can be made more parsimonious by retaining the same variance parameter and estimating only two different means.

2.2 Understand Variance Better

In a quiz, the unexpected happens quite regularly. For instance, a strong player might miss an easy question that most of the league answers. Conversely, a relatively weak player might answer a question that is right up their wheelhouse but which stumped the rest of the league. These occurrences will increase the variance σ^2 of a quizzer's performance. In the above outlined method of QFort estimation, the estimate of the mean μ doesn't react much to outlier events. Instead, it is the variance estimate σ^2 that sees large movements when a player's answer confounds expectations. In fact, the current QFort ratings are derived using a constrained optimization that upper bounds the σ^2 estimate at 0.09 to encourage more movement in μ.

2.3 Make the setup truly Bayesian

The current approach is not "incremental". With every new question, the entire estimation process is re-run to estimate a new (μ, σ^2) which runs counter to the philosophy of Bayesian estimation. Further, this might also dampen movement in μ since each question only adds a little incremental data to the player's history and the estimation is done across the entire history.

The way to make this approach truly Bayesian is to specify some priors on μ and σ^2 and allow their posterior distributions to evolve as a function of $(\vec{\boldsymbol{d}}, \vec{\boldsymbol{a}})$. The QFort rating after N questions will then be the mean of this posterior distribution after N updates. This method also has the added advantage that estimation is incremental - when the $(N+1)^{\mathbf{th}}$ question comes along, the posterior is updated based only on this question.

Technically, suppose the posterior distribution of (μ, σ^2) after $(N-1)$ questions is $f_{N-1}(\mu, \sigma^2)$. Suppose further that the next question has a difficulty level of d_N and the player answered with a_N. Then, the updated posterior is

$$\begin{aligned} f_N(\mu, \sigma^2) &\propto \int P(a_N | d_N, \mu, \sigma^2) f_{N-1}(\mu, \sigma^2) d\mu d\sigma^2 \\ &\propto \int \left[(1-a_N)\Phi\left(\frac{d_N - \mu}{\sigma}\right) + a_N\left(1 - \Phi\left(\frac{d_N - \mu}{\sigma}\right)\right)\right] f_{N-1}(\mu, \sigma^2) d\mu d\sigma^2 \end{aligned}$$

We need to sample from this posterior distribution using some algorithm such as Gibbs sampling or Metropolis-Hastings. We can compute posterior mean by averaging these samples. The binary nature of the observables (a_i) means that the evolution of posteriors is complicated and there won't be any well-behaved conjugate priors for $f_k(\mu, \sigma^2)$. Given this, it isn't clear to me how to simulate $f_{N-1}(\mu, \sigma^2)$.