ChatGPT

Adaptive Understanding Score Model

We introduce a mathematical framework that dynamically tracks a learner's mastery of a topic (e.g. fluid dynamics) based on quiz performance. Let **θ_t** denote the student's *understanding score* at time *t* (for a given domain). Each quiz question *i* has: - a **difficulty** level *d_i* (on the same scale as θ), - a **relevance** factor *r_i*∈[0,1] (how closely it matches the user's notes/syllabus), - and the student's **correctness** *C_i* (1 if correct, 0 if incorrect, or a fractional score if partial credit).

We model the probability of a correct answer with a logistic (Rasch/IRT) function of (ability − difficulty):

$$P_i = P(\text{correct} \mid \theta_t, d_i) = \frac{1}{1 + \exp[-(\theta_t - d_i)]},$$

so that easier questions (θ≫d) give high *P* and harder questions (θ<d) give low *P*. This follows the standard IRT/Rasch formulation [1].

After observing the student's answer, we update the understanding score. A convenient rule (analogous to Elo-rating updates [2]) is:

$$\theta_{t+1} \;=\; \theta_t \;+\; \alpha_t \;\times\; r_i \;\times\; \big(C_i - P_i\big),$$

where **α_t** is a *learning rate* (possibly changing with t). Here **(C_i – P_i)** is the prediction error: if the student outperforms expectation (C=1 but P<1) it is positive, boosting θ; if they underperform (C=0 but P>0) it is negative, reducing θ. Multiplying by *r_i* means that less relevant questions cause smaller adjustments. For example, if *r_i=0.5* (only half-relevant), then only half of the update is applied to the fluid-dynamics score, reflecting that the question partly tested unrelated material.

We can implement a *dynamic learning rate* α_t to reflect user level. For instance, beginners (low θ) might have a larger α (rapid learning), while advanced users get smaller α (fine-tuning). One practical choice is to decay α with the number of questions *N_t* attempted so far, e.g.

$$\alpha_t \;=\; \frac{\alpha_0}{\sqrt{1 + N_t}},$$

or use an exponential decay α_0 exp(–γ N_t). In Elo-style systems, this is akin to a "K-factor" that decreases for higher-rated players [3]. Alternatively, α_t could depend directly on θ_t (e.g. α_t∝1–σ(θ_t)).

Putting it all together, our **update equation** is:

$$\theta_{t+1} = \theta_t \;+\; \alpha_t\, r_i \big(C_i - \tfrac{1}{1+e^{-(\theta_t - d_i)}}\big).$$

In words, the model predicts the chance of a correct answer via the difference (θ–d) (a higher θ relative to d makes success more likely [1]). The actual result (correct or incorrect) then nudges θ up or down by an

amount proportional to the error (C–P). Difficulty *d_i* naturally scales how surprising a correct answer is (hard questions give larger updates when answered correctly), and relevance *r_i* scales the update to focus on the target domain. The factor α_t ensures early questions have big impact (rapid learning) while later questions yield smaller, stabilizing updates.

- **Predicted correctness (logistic model):** $P(\text{correct}) = \frac{1}{1 + e^{-(\theta_t - d_i)}}$ (Rasch/1PL IRT model) [1].
- **Score update:** $\theta_{t+1} = \theta_t + \alpha_t \, r_i \, (C_i - P_i)$, analogous to an Elo update $R' = R + K\,(S - E)$ [2].
- **Difficulty:** Appears in *P_i*; harder questions (higher *d*) lower *P_i*, so a correct answer yields a larger increment (learning more from tougher items).
- **Relevance:** Factor *r_i*∈[0,1] multiplies the update; if a question is only partly related, it only partially updates the domain score.
- **Dynamic learning rate:** α_t can decrease over time or with θ, e.g. $\alpha_t = \alpha_0 / \sqrt{1 + N_t}$. This ensures diminishing updates as the student learns more, similar in spirit to lowering the Elo K-factor for experienced users [3].

Illustration with a Fluid Dynamics Example

Consider a student learning fluid dynamics. Initially suppose their understanding score is θ_0=0.2. The tutor gives a question on Bernoulli's principle with difficulty d=0.7 and relevance r=1. The predicted probability of a correct answer is:

$$P = \frac{1}{1 + e^{-(0.20 - 0.70)}} = \frac{1}{1 + e^{0.50}} \approx 0.38.$$

- **If the student answers correctly (C=1)**, the error term is 1−0.38=0.62. With a high initial learning rate, say α=0.5, the score update is Δθ=0.5·1·0.62=0.31. So θ increases to ≈0.51, reflecting a big jump after mastering a hard question.
- **If instead they answer incorrectly (C=0)**, the error term is 0−0.38=−0.38, and the update is −0.5·1·0.38=−0.19, dropping θ to ≈0.01 (the student clearly needs more study).

Next, the student reviews an explanation of a concept from their notes, which also raises understanding (we could model this as a partial "correct answer" of a virtual question, or simply adjust θ upward by a small amount). Later another question is asked, e.g. on fluid continuity with d=0.4. Now θ≈0.51, so P≈1/(1+exp(–0.11))≈0.53. If the student answers correctly, the update is Δθ=0.5·1·(1−0.53)≈0.235, giving θ≈0.745. Notice the increments are smaller as θ grows (expected: going from 0.51 to 0.745, not doubling again).

If a question is only partially about fluid dynamics, say about general calculus with r=0.5, then only half of Δθ would apply to the fluid dynamics score. For example, a math question (difficult, say d=0.5, r=0.3) answered correctly (θ=0.745⇒P≈1/(1+exp(0.255))≈0.44) would yield Δθ≈0.5·0.3·(1−0.44)≈0.083.

Over time, as *N* (questions seen) increases, α_t might drop (e.g. to α≈0.3 after many questions). Thus later updates become even smaller – by design the student is reaching mastery. The app can display θ on a [0,1] or [0,100%] scale if desired, updating it after each quiz to give real-time feedback on the learning curve.

Integration into an AI Tutoring App

This model fits naturally into an app workflow. For each **topic/domain** (e.g. fluid dynamics), the app maintains a running θ score. When a user studies notes or gets an explanation from the AI tutor, this builds implicit understanding (the app might optionally give a small boost or set an initial θ). Then each time the user takes a quiz question:

1. The system estimates *difficulty d_i* (e.g. by pre-tagging the question) and *relevance r_i* (e.g. by matching question keywords to the user's notes/topics).
2. It computes the probability *P_i* using the current θ and updates θ using the formula above.
3. The learning rate α_t is adjusted as needed (for example decayed after each question or when θ crosses proficiency thresholds).

Thus the **understanding score is dynamically updated** after every answer. The app can display this score as a percentage or proficiency bar in the fluid dynamics module. Because our update rule requires only the last score, difficulty, relevance and correctness, it can run in real time on-device. The student can also review their note-taking topics: if future questions drift outside those topics (low *r*), the app ensures the fluid-dynamics score isn't artificially inflated by unrelated items.

For example, if a student repeatedly answers questions on fluid statics correctly, their θ will rise and plateau. If later they review a tangent topic (e.g. general physics), the limited relevance factor means their fluid-dynamics θ changes little. The dynamic α ensures large early gains (when the user is at a "novice" level) and smaller increments as they become more advanced – mirroring human learning curves in adaptive systems.

Comparison to Other Models

- **Elo-like models:** The classic Elo update $R' = R + K\,(S - E)$ [2] is very similar in form. In Elo, *K* is fixed or varies by player strength [3]; here α_t plays that role. Elo has been used for knowledge estimation (treating student vs question as a match) [4]. Our model is effectively an Elo-style rating where *expected score* is given by the logistic Rasch formula, and we add a *relevance* multiplier to focus updates. Like Elo, it is interpretable (updates are simple errors) and adaptive in real time [4], but our extension explicitly weights question content and allows partial credit. In practice, Elo and Rasch have been shown to give similar ability estimates given enough data [5] [6].

- **Item Response Theory (Rasch):** IRT models (especially the 1PL/Rasch model) give $P(\text{correct}) = 1/(1 + e^{-(\theta - d)})$ [1]. These are statistically principled and interpretable (θ is latent ability, d is item difficulty). However, traditional IRT is static and assumes pre-calibrated items. Our model uses the same logistic idea for P, but updates θ online per interaction (no batch re-calibration needed). Like Rasch, it is interpretable and provides meaningful probabilities, but it is more adaptive – we update after each response, whereas pure IRT would require a full test to re-estimate ability. Moreover, we add topic relevance and dynamic α, which standard IRT lacks.

- **Bayesian Knowledge Tracing (BKT):** BKT treats each skill as a binary mastered/unmastered variable in a hidden Markov model [7]. Observations (correct/incorrect) update the probability of mastery via parameters: initial knowledge p(L_0), learning rate p(T), slip p(S), and guess p(G) [8]. BKT is intuitive

for modeling mastery learning in ITS, but it is limited (each skill is simply known or not, and update rules are fixed). Our model, in contrast, gives a continuous θ (more nuanced than binary mastery) and uses explicit item difficulty and correctness errors for updates. This yields greater flexibility (it can model partial learning) while remaining interpretable. Like BKT, it can update online after each question, but it does not require estimating slip/guess parameters from data.

Overall, our **Adaptive Understanding Score** model blends strengths of these approaches: it uses an interpretable logistic/Elo-style update rule, tracks a continuous mastery score for a topic, adapts after every answer, and incorporates content relevance. It is more adaptive than static IRT (real-time updates) and more granular than binary BKT, yet remains transparent. This makes it well-suited for an AI-driven learning app: it can incorporate new questions on the fly without retraining, and students can readily understand how their score changes with performance.

References: Foundational ideas are drawn from the Rasch (1PL IRT) model [1], the Elo rating update [2] [4], and the principles of Bayesian Knowledge Tracing [7] [8].

---

[1] Rasch Modeling | Columbia University Mailman School of Public Health
https://www.publichealth.columbia.edu/research/population-health-methods/rasch-modeling

[2] [3] Elo rating system - Wikipedia
https://en.wikipedia.org/wiki/Elo_rating_system

[4] [5] [6] Adaptation of the Multi-Concept Multivariate Elo Rating System to Medical Students' Training Data
https://arxiv.org/html/2403.07908v1

[7] [8] Bayesian knowledge tracing - Wikipedia
https://en.wikipedia.org/wiki/Bayesian_knowledge_tracing