Adaptive Learning for Competitive Physics Problem Solving with Bayesian Knowledge Trace

Stanford CS109 Challenge

Berwyn Berwyn
Stanford University
berwyn@stanford.edu

1 Motivation

Problem solving in physics, particularly in high school competitive physics, requires a thorough understanding of various physical concepts as well as the capability to solve complicated problems in a short period of time. This project allows students to receive a suitable problem selection each time based on their performance history and statistics on prior students' performance, including correctness and speed.

2 Dataset

There are only two datasets used in this project: the time needed for previous students to solve the problem and the difficulty level, which is determined by previous students' performances on the problems. Both of these datasets are artificially generated. The difficulty level does not require careful handling because increasing the number of a (more on this in Section 2, Subsection 5) makes the problem more difficult. For the time dataset, I generated all the times with scipy by sampling 100 normal distribution variables for every problem. Although there is no underlying dataset on what the distribution is, assuming it to be normal, especially as the number of attempts increases, is plausible.

3 Approach

1. **Bayesian Knowledge Trace Formulas** BKT involves four probabilistic parameters [1]: $P(L), P(S), P(G)$, and $P(T)$, which are defined respectively as the probability of knowing how to solve a problem, the probability of getting a problem wrong despite knowing the concept, the probability of getting a problem right despite not knowing the concept, and the probability of improvement after solving a problem, particularly in the context of time. Each student holds a record of prior $P(L)$, which is updated with the following Bayesian formula:

$$P(L|correct)_{new} = \frac{P(L) \times (1 - P(S))}{P(L) \times (1 - P(S)) + P(G) \times (1 - P(L))} \tag{1}$$

$$P(L|wrong)_{new} = \frac{P(L) \times P(S)}{P(L) \times P(S) + (1 - P(G)) \times (1 - P(L))} \tag{2}$$

$$P(L) = P(L) + (1 - P(L)) \times P(T) \tag{3}$$

   To prevent the problem from getting difficult too soon or too long, the above equations are modified as follows:

$$\gamma = \frac{P(L) \times (1 - P(S))}{P(L) \times (1 - P(S)) + P(G) \times (1 - P(L))} \tag{4}$$

$$P(L) = P(L) + \alpha(\gamma - P(L)) \tag{5}$$

$$P(L) = 0.5 \times P(L) + 0.5 \times P(T) \tag{6}$$

Where α is an adaptive learning rate determined in section 3.

For each problem, three conceptual questions regarding the problem are given. Aside from helping students integrate multiple physics concepts, these questions can also estimate $P(G)$ and $P(S)$ since there is no chance a student could answer a physics question correctly if they used the wrong concept (for example, thinking that angular momentum is conserved when it's not).

2. **Beta Distribution and Sampling** $P(G)$ and $P(S)$ are modeled with a beta distribution, $Beta(a, b)$, where a and b are updated based on the number of conceptual questions they got correct and the correctness of the core problem. In particular, if the student got the question correct:

$$a_G = a_G + wrong \tag{7}$$
$$b_G = b_G + correct \tag{8}$$

Conversely if the student got the question right:

$$a_S = a_S + correct \tag{9}$$
$$b_S = b_S + wrong \tag{10}$$

To demonstrate why these equations make sense, consider, for example, equation (7). Since the student got the question correct, the more they made mistakes on the conceptual questions, the larger a_G would be. This ultimately increases the value of the expected value of the Beta distribution, which means that the student is more likely to have guessed the solution.

To get the value of $P(G)$ and $P(S)$ from the Beta distributions to calculate $P(L)$, 20 values are sampled with numpy, and the averages of the values are assumed to be the approximate $P(G)$ and $P(S)$. The number of samples is kept low enough such that the probabilities won't converge to a particular value. This aims to add a little bit of randomness so that the difficulty level won't grow or decay monotonically.

3. **Learning Rate** The learning rate α aims to prevent $P(L)$ from growing too fast or too slow. The value is calculated based on the student's performance on the previous problems. For every problem, a performance metric is defined as:

$$p = 0.3 \times \frac{c}{3} + 0.7 \times a \tag{11}$$

Where c is the number of correct answers for the conceptual questions and a is an indicator variable, which takes a value of 1 if the student got the core question correct. This performance metric is stored in a list, and for every new problem, the average μ and variance σ^2 of all the performance metrics in the list are calculated. The intuition for the adaptive learning rate is that if the variance is high, the learning rate should be lower due to the inconsistency the student showed. If the mean is high, the adaptive learning rate should be high because that means the student got many answers correct previously. In other words, the adaptive learning rate is directly proportional to the mean and inversely proportional to the variance. The guess probability is also included here:

$$\alpha = 2 \times \frac{\mu}{3 \times P(G) + 5 \times \sigma^2 + 1} \tag{12}$$

4. **Calculating P(T)** To calculate $P(T)$, the model interacts directly with a JSON file which stores the information about all the times required for previous students to solve a problem. In this case, the time needed for a student to solve a particular problem is assumed to be a normal distribution with mean μ_i and standard deviation σ_i. $P(T)$ is then calculated by taking the weighted average of the prior value and 1 - $\phi\left(\frac{t-\mu}{\sigma}\right)$:

$$P(T) = 0.8 \times P(T) + 0.2 \times \phi\left(\frac{t-\mu}{\sigma}\right) \tag{13}$$

Equation (13) ensures that in order to maintain a decent value of $P(T)$, a student must be consistently ace the problems.

5. **Difficulty Level** Each problem is also assigned a difficulty value (δ), which is also a Beta random variable where a represents the number of previous students who got the question right and b represents the number of previous students who got the question wrong. The difficulty level is also calculated by sampling, but now 1000 values are sampled to ensure a more stable difficulty value for the same parameters.

6. **Problem Determination** To determine the next problem to show up, a score is evaluated with this formula for all problems:

$$score = |p(L) - (1 - \delta)| \tag{14}$$

   The selected next problem is a problem with the lowest score, reflecting the most suitable difficulty level for the student's current knowledge.

7. **Detecting Outliers** In calculating $P(T)$, the time required for the current student to solve a problem is also fetched to the JSON database. This allows the database to be updated as more students attempt to solve the problems. There are two problems that might emerge if a student cheats. First, the a parameter for the difficulty beta distribution will get higher when it is not supposed to, which will lower the difficulty level and undermine the algorithm. Second, if a student's time to solve a problem is unreasonable (too quick or too slow), this time value will be updated to the database and can undermine the distribution. In order to handle this issue, a function called is_outliers is introduced. This function basically rejects all time values below three standard deviations below the mean and above three standard deviations above the mean.

4 Limitation

This model occasionally works differently from the expected behavior due to the simplicity of the model. Some formulas presented above also involve factors which were figured out by brute force tuning: trying out different values and seeing how the model behaves. The more robust and accurate way to create this same functionality might require complex deep learning or other massive probabilistic models.

References

[1] Michael V. Yudelson, Kenneth R. Koedinger, and Geoffrey J. Gordon. Individualized bayesian knowledge tracing models. In H. Chad Lane, Kalina Yacef, Jack Mostow, and Philip Pavlik, editors, *Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings*, volume 16, pages 171–180, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.