Student Model Learning

Francisco Manuel Franco Azeiteiro
franciscoazeiteiro@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2017

Abstract

This work attempts to have the Zone of Proximal Development and Empirical Success (ZPDES) algorithm receive information for grouping students by estimating students' learning speed and proposing the model with approximately optimal parameters for students of that competence level. This was attempted by several different means including performing a diagnostic test before applying the algorithm itself, estimating the learning speed during the execution of activities and utilizing the optimal parameters for the most similar student profile. The obtained results show that this methodology allows for a better balance between the amount of skills learned and the number of time steps required to learn them than using the average parameters for every student, although the created model would need several adaptations in order to be used with real students.
Keywords: Student; Tutoring; Learning; Intelligent tutoring systems

1. Introduction

Intelligent Tutoring Systems (ITS) are tutoring systems based on Artificial Intelligence programming techniques, providing some degree of individualized instruction [2]. Student model learning has been a popular area within ITS, its main goal is to accurately predict how the student will answer to a series of questions. Being able to achieve that goal would greatly increase the progress toward making ITS a powerful tool for teaching in general, allowing for more effective teaching systems to exist in several different mediums, from school education to advanced training in highly specialized environments.

Several ITS-based systems already exist and are used for different types of learning experiences in grade school environments [7] and otherwise, such as the *SHERLOCK* system [8], used to train Air Force technicians in diagnosing electrical systems issues on F-15 jets, providing different levels of assistance depending on the performance of each trainee.

1.1. Problem

All algorithms currently used in ITS either assume that all students are the same or that all students are different, the difference being that algorithms that treat students equally do not model individual student parameter, while algorithms that treat students differently attempt to explicitly model them, usually via estimation based on their performance during an exercise, or by giving these parameters to the students prior to solving exercises. While it is clear that all students are not equal, the systems that treat them differently do not take advantage of the degree of similarity between students. It would be interesting to analyze the results of a system which is able to take advantage of the similarity between students by estimating the attribute values of certain key characteristics and proposing an algorithm with optimal or approximately optimal parameters for students of that competence level.

1.2. Hypothesis

The hypothesis being attempted will be the possibility of grouping students together based on certain criteria in such a way that the students are able to use models with optimized parameters for similarly competent students to their advantage. Whenever a new student uses the system, his or her performance in certain exercises will allow the system to decide what group of students is estimated to have the most similar students in terms of learning capabilities. Challenges of this approach include accurately placing students on their correct group and making sure that the model parameters for each group of students are optimal.

1.3. Objectives/Expected contributions

The following are the expected contributions and objectives of this work:

- To implement a version of the ZPDES algorithm supported by parameter estimation algorithms which allow the discovery of grouping criteria based on parameters of both the algorithms and the students. This new al-

gorithm is called the Group-Optimal Zone of Proximal Development and Empirical Success (GOZPDES) algorithm

- To make the above system capable of grouping students based on their similarities under certain criteria, optimizing the learning sequence for each group
- To test the new implementation with the virtual students via simulation, and perform statistical analysis of the obtained results

2. Background

2.1. Hidden Markov Model (HMM)

A HMM is a type of Markov model in which the system respects the Markov property and the states are not directly observable. However, the output, which is state-dependent and based on emission probabilities, is observable [5]. A HMM is able to model this situation in such a manner that it is possible to make assumptions on what state the model is on at any given time based on the output, allowing estimations on future output and states to be made and consist of a tuple with the following parameters (for a thorough explanation see [1]):

- A set of states S
- A set of actions A
- Transition probabilities $P_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$, representing the probability of transitioning from state s to s' at time $t+1$ knowing that the action performed at time t was a
- A reward indicator $R_a(s, s')$, which states what is the reward for passing from state s to s' by performing action a
- A discount factor γ, which may be used to discount the value of future rewards

2.2. Expectation Maximization (EM)

The EM algorithm attempts to iteratively calculate the maximum likelihood of a set of parameters. It is split into two steps: The E step, which finds the distribution for the unobserved variables which, in the ITS domain, often refers to the student's knowledge state of skills and, when applicable, sub-skills. This distribution is found through the current estimate of the parameters, as wells as the values of the observed variables. The M step consists on re-estimating the parameters to be those with maximum likelihood, assuming that the distribution found in the E step is correct [10] [9]. There are several variants of this algorithm depending on the domain. When working with HMMs, the Baum-Welch algorithm can be used to estimate the model's transition, emission and initial probabilities [11]. This algorithm will be used to estimate student parameters when required.

2.3. Bayesian Knowledge Tracing (BKT)

BKT [4] has been a popular approach to the problem presented by ITS in the student model. The original work consists on having students trying to learn a set of skills by asking them to solve exercises which require the use of those skills in a practice environment. The system also attempts to infer if the student has learned any one skill by using a HMM [12] with two states for each one: the learned state, where the student knows the skill, and the unlearned state, where he doesn't. It is assumed that once a student learns a certain skill it cannot be forgotten.

The model possesses four parameters:

- $P(L_0)$ - the probability that the student possesses a particular skill prior to the first opportunity to apply it
- $P(G)$ - the probability that the student will answer correctly to the question if the skill is in the unlearned state
- $P(S)$ - the probability that the student will answer incorrectly if the skill is in the learned state
- $P(T)$ - the probability that the skill being learned transitions from the unlearned to the learned state after an opportunity to apply it.

Corbett and Anderson attempt to calculate how likely it is that a student understands a particular subject matter using the following formula:

$$P(L_n) = P(L_{n-1}|evd) + (1 - P(L_{n-1}|evd)) \times P(T) \quad (1)$$

Where evd is the evidence, or the information that can be derived from previous results.

2.4. ZPDES

The ZPDES algorithm [3] is a tutoring model algorithm which requires little information from the student and cognitive models by attempting to estimate student competence online via a reward function. The subjects being taught are organized in Knowledge Component (KC)s, and the objective of this algorithm is to make sure that the students learn every KC. The reward is computed using Equation (2), calculating the difference in amount of correct answers of the last $d/2$ samples and the previous $d/2$ samples.

$$r = \sum_{k=t-d/2}^{t} \frac{C_k}{d/2} - \sum_{k=t-d}^{t-d/2} \frac{C_k}{d - d/2} \quad (2)$$

Since there are typically too many activities to be explored it is expected to have a previously defined Zone of Proximal Development (ZPD), which connects different activities via dependence relationships.

3. Algorithm

3.1. Clustering and Classification

Regarding the issues related to the grouping aspect, a variant of the EM algorithm will be used to attempt to extract features which may be used as criteria for the system. Other techniques for criteria selection will be manual and take into consideration the exercises proposed. For example, if the system is made to offer the same initial exercises for every student, it is possible to use the value of r from the ZPDES algorithm after a fixed amount of exercises (Equation (2)).

The classification task in this work consists in deciding what group each new student belongs to. The learning speed is used as the criterion used to distinguish among different student groups, meaning that the most important part in this step is to accurately estimate this value, which is programmed as an attribute of each student. The system cannot directly obtain this attribute, as this would be impossible to do when interacting with real students, so the Baum-Welch algorithm will be used instead.

3.2. Student comparison

One of the main challenges of this work is in understanding how the comparison of students can be made. This is a difficult problem because when working on a simulation the implementation of students needs to have a learning component which is similar to how real students learn, but at the same time remain relatively simple, as a complicated combination of variables would make the comparison process difficult. Simultaneously, the goal of this work is to create a system that ensures that students learn a set of skills. Therefore, comparison methods which focus directly on student correctness are not proper for this environment, as ideally all students will finish the GOZPDES algorithm having learned every skill.

Results obtained from performed tests focusing on changing the difficulty of skills, as well as the number of exercises and students show that student comparison based on their rate of correct answers in exercises is decreasingly accurate as the number of students increases, meaning that directly comparing the students to each other based on how they answer certain exercises would come with several issues. An alternative to this would be performing the comparison based on one or more student attributes, but since it is not possible to directly obtain such a value from real students directly then whatever chosen attribute or attributes are chosen need to be estimated and since the only difference that can be seen between students with unknown internal variables is the correctness of their answers, these must be used as information to perform the estimations.

3.3. Profile students/Student profiles

The clustering problem of this work is resolved through the use of profile students. A profile student is no more than a student that has a GOZPDES with optimal parameters associated to it, meaning that the given GOZPDES is optimal for all students with a similar *baselearn* value. As such, the classification problem can be resolved by assigning students to their most similar student profile in terms of *baselearn*. This attribute is known by us beforehand since profile students would not be replaced by real students for this system's eventual transition into a real learning environment. However, the same cannot be said about the other types of students, as transitioning this system into the real world would mean that the *baselearn* value of these students cannot simply be obtained by calling a method. This work operates under the philosophy that it must be adaptable into a real world environment, meaning that in this case certain attributes must be estimated instead of simply obtained.

4. Implementation

4.1. Architectural overview

The original Python program with the ZPDES and algorithm is available on `https://github.com/flowersteam/kidlearn_lib` [3], with the solution of this work being developed in Java based on the previously existing functionalities. The reasons for changing the programming language used is to allow a reconstruction of the code, both as a learning process and as a way to modify the underlying structure of the classes in order to allow a new architecture with simpler, easier to understand classes and methods, as well as allowing the use of already existing Java libraries. In particular the Baum-Welch algorithm - the used implementation of this algorithm was obtained from the *jahmm* library [6].

4.2. Class rundown

Students of different capabilities are implemented through two attributes called *baselearn* and *baseinit*, with *baselearn* influencing the probability that the student has of learning skills and *baseinit* influencing the probability of the student already knowing each skill before solving an exercise containing said skill.

Students are modeled differently from the typical BKT formula - there is a possible transition from not knowing to knowing the skill, but it is not made in the same way as originally done in Corbett and

Anderson's work. Given a skill sk and a student st, the probability that the student has of learning the skill after each exercise, assuming that the student didn't know the skill before, is given by Equation (3):

$$P(T_{st}) = baselearn_{st} \times x + P(T_{sk}) \times y \quad (3)$$

Where $baselearn_{st}$ is the student's *baselearn* attribute, $P(T_{sk})$ is skill sk's $P(T)$ value, and x and y are two factors that influence the weight of $baselearn_{st}$ and $P(T_{sk})$, with $x+y=1$. As a result from tests created with the objective of determining the x and y values to be used while making sure that both the student's learning speed and the skill's learning difficulty are taken into consideration, the used values are $x = 0.55$ and $y = 0.45$.

A similar equation was used to determine the student's probability of knowing the skill before attempting to apply it in any exercise (Equation (4)):

$$P(L_{0st}) = baseinit_{st} \times z + P(L_{sk}) \times t \quad (4)$$

In this case the values used for z and t are 0.1 and 0.9 respectively, focusing on how well known the skill's domain is in general.

4.3. Parameter optimization

Each group of students will have a set of parameters which determine how the GOZPDES algorithm will function for each group. Within each group the parameters are chosen as the optimal values that minimize the average number of time steps for that group's student profile while maintaining the probability of learning all skills above a certain threshold. Since the parameters are optimal for the profile, then as long as the classification step is done correctly all students will have approximately optimal parameters.

Student profiles were created for *baselearn* values from 0.1 to 0.9 in increments of 0.1, for a total of 9 profiles. Ultimately, the optimal parameters for each profile (the nomenclature used for student profiles is P_x, where x is that profile student's *baselearn* attribute) are shown in Table 1, along with the time taken by each profile if using the optimal parameters of $P_{0.1}$ or $P_{0.4}$.

In addition to the student profiles taking longer when not using optimal parameters in Table 1, $P_{0.6}$, $P_{0.7}$, $P_{0.8}$ and $P_{0.9}$ did not reach the 97% threshold when using the optimal parameters for $P_{0.1}$. Analyzing this table's results in terms of the difference in the average number of time steps, it seems that the closer the profile's optimal parameters were to the optimal $P_{0.1}$ parameters the less difference there is in regards to time. This means that eventual small errors in the estimation of *baselearn* and classification of students based on this parameter are not very harmful for the student's performance. Additionally, the time required for the students under the optimal parameters for $P_{0.4}$ shows that for Students $P_{0.3}$, $P_{0.5}$ and $P_{0.6}$ there is little difference between their optimal parameters and the "default" parameters. Given this similarity between optimal values for students with approximate *baselearn*, an eventual error in selecting the optimal parameters would likely be of little consequence, as the optimal parameters for an almost equivalent student profile are very similar.

4.4. Optimal parameters identification

The following GOZPDES parameters directly influence the number of time steps and probability of knowing every skill that the students have at the end of the ZPD:

- The β and η values, determining the confidence in a new reward value for $w_a \leftarrow \beta w_a + \eta r$
- The γ parameter, which determines the exploration rate when calculating the probability of selecting an activity to perform.
- The ω and θ values determining when the ZPD can skip ahead and add a new activity when successfully answering hard activities. These parameters were removed for the final implementation of the GOZPDES algorithm as they were not improving the system's results in any way.
- The d parameter, which determines the number of most recent answers used to calculate the reward value from an activity, as well as the minimum amount of times that an activity needs to be proposed before the ZPD can expand from it.
- The removal threshold - the success rate that an activity needs to have before it can leave the ZPD.
- The expansion threshold - the success rate that an activity needs to have before its dependent activities may join the ZPD.

All of these parameters, with the exception of the expansion threshold, were tested and manually optimized for all student profiles.

For reference, the first test performed with this new model, to be used as a baseline, contained the following parameter values:

- $\beta = 0.6$
- $\eta = 0.4$
- $\gamma = 0.5$

Table 1: Optimal parameters and average number of time steps for a series of student profiles along with a comparison to using the optimal parameters for $P_{0.1}$ and $P_{0.4}$. Results with a * afterwards did not reach the required 97% threshold.

Parameter	$P_{0.1}$	$P_{0.2}$	$P_{0.3}$	$P_{0.4}$	$P_{0.5}$	$P_{0.6}$	$P_{0.7}$	$P_{0.8}$	$P_{0.9}$
Rmv thrshld	0.6	0.6	0.6	0.55	0.55	0.55	0.55	0.55	0.55
β	0.9	0.9	0.9	0.8	0.7	0.65	0.6	0.5	0.45
d	8	8	6	6	6	6	4	4	4
γ	0.2	0.1	0.2	0.25	0.35	0.4	0.45	0.45	0.5
Optimal time	271.9	202.9	160.7	133.5	116.9	105.0	89.5	81.9	76.0
$P_{0.1}$ time	N/A	204.6	166.7	142.3	125.9	114.3*	105.4*	98.5*	92.9*
$P_{0.4}$ time	264.9*	195.5*	157.6*	N/A	117.3	105.7	96.0	88.8	83.4

Table 2: Average probability of knowing every skill and number of time steps after going through the sequence of activities for students S1,S2,S3 and S4 (with *baselearn* values of 0.1, 0.8, 0.65 and 0.2 respectively)

S1 time	S2 time	S3 time	S4 time
402.5	115.3	134.5	290.0
S1 %skills	S2 %skills	S3 %skills	S4 %skills
97.8	99.7	99.5	98.2

- $\theta = 0.85$
- $\omega = 0.8$
- $d = 4$
- removal threshold = 0.8

The results can be seen in Table 2.

Given the high probability of knowing all skills being displayed in Table 2, adjustment to the parameters should seek to decrease the number of required time steps, without compromising the probability of knowing all skills too much.

5. Results

5.1. Student competence estimation

As it was previously mentioned, the probability of student *st* learning a skill *sk* after performing an activity (or exercise) A which includes it is given by Equation (5):

$$P(T) = (0.55 \times baselearn_{st} + 0.45 \times P(T_{sk})) \times w_{skA} \quad (5)$$

Which can be solved for *baselearn* as shown by Equation 6:

$$baselearn = \frac{P(T) - 0.45P(T_{sk})w_{skA}}{0.55w_{skA}} \quad (6)$$

Thanks to the Baum-Welch algorithm, it is possible to estimate the value of $P(T)$, which can then be used to recalculate the student's *baselearn*. Having determined the optimal parameters and possessing a way to estimate the students' learning speed through re-estimation of the *baselearn*, it is now time to see if it is possible to have the virtual students take advantage of these techniques. The next test performs the *baselearn* estimation on students with different values of the attribute, assign them the optimal ZPDES parameters of the student profile most similar to them and run the ZPDES algorithm for the students using these parameters. The results are shown in Table 3. It is worth mentioning that the average number of steps and probability of knowing all skills for the random students is sorted by which profile they were assigned as most similar, meaning that errors in classification are possible.

Along with the results of Table 3, the average parameters obtained were $d = 5.628$, Remove threshold = 0.566, $\beta = 0.688$ and $\gamma = 0.339$. Since d is necessarily an even number, it is rounded up to 6.

Analysis of Table 3 shows that comparing to the default parameters, the estimated optimal parameters greatly decrease the average number of time steps at the expense of a penalty in the probability of knowing all skills. Still, with the exception of the students assigned as most similar to profile student $P_{0.6}$, the average probability for all students is a minimum of 90%. Overall, these results are better than the ones obtained when using the default parameters. For example, the random students assigned as most similar to profile student $P_{0.7}$ show better results in both of the observed values than the respective profile student using the default parameters. Comparing to the profile students using their optimal parameters, the number of time steps is, as expected, lower than the ones obtained by the random students.

5.2. Online grouping

So far, the estimation of the *baselearn* attribute is obtained before the students actually enter the ZPDES algorithm, generating observations through a diagnostic test of sorts. While this approach is valid and can also be used on real students, the number of time steps taken does not take this test into account. The next test shall measure the average number of time steps and probability of know-

Table 3: Average number of time steps and probability of knowing all skills for students with random *baselearn* values, sorted by which profile student was considered to be the most similar to each student, along with the average number of time steps for optimal and default parameters, as well as the probability of knowing all skills for each profile when using default parameters. The results show that the random students are on average less likely to learn all skills, but are faster at going through the ZPD

Parameter	$P_{0.1}$	$P_{0.2}$	$P_{0.3}$	$P_{0.4}$	$P_{0.5}$	$P_{0.6}$	$P_{0.7}$	$P_{0.8}$	$P_{0.9}$
Num steps	321.5	235.0	179.0	158.2	134.8	124.7	106.7	98.1	89.7
% all skills	99.3	97.5	92.5	97.3	90.6	86.6	98.4	93.8	91.6
Default steps	364.3	267.2	213.1	179.2	155.5	139.2	126.3	116.3	108.4
Default % skills	99.9	99.9	99.8	99.5	99.0	98.5	97.6	96.5	95.3
Optimal time	271.9	202.9	160.7	133.5	116.9	105.0	89.5	81.9	76.0

ing all skills for when initially using default parameters - the average parameters obtained from Table 3. Several different conditions determining when the Baum-Welch algorithm is executed are tried. The results are shown in Table 4.

The results in Table 4 show that dynamically determining the optimal parameters for each student is a viable option.

Out of the conditions tested, the one that penalizes these students the least is the parameter update after performing 30 time steps. The most successful results in Table 4 in terms of minimizing the number of time steps while keeping a high probability of knowing all skills is the "30 steps +" condition.

Figures 1 and 2 show the decrease in the average number of time steps and increase in the probability of knowing all skills respectively for using the "30+", after 20 time steps, removing activities A1 and B1 from the ZPD for the different student profile values or the a priori estimation in comparison to simply using the average parameters throughout. The a priori method has a 20 time steps penalty which corresponds to the time taken to perform a diagnostic test. Figure 1 shows a decrease in the number of time steps for the "30+" and removing A1 and B1 methods, with the "30+" condition being the better method. While there is little to no gain for the worst students, as the average *baselearn* value of the students increases, the average number of time steps decreases further. The same can be said for the improvements in the average probability of knowing all skills - there is little to no improvement for the worse students, but the better the students are the greater the improvement is. This most likely happens because the GOZPDES algorithm naturally causes the worse students to say within it for such a long number of time steps that they are essentially guaranteed to learn every skill by the time that they leave the ZPD, allowing for essentially no gain in that regard. Simultaneously, there is little difference on the number of time steps for the slower students because the main responsible for their slower progress through the ZPD is the expansion threshold, which is deliberately the same for all students in order to ensure that slow learners are unable to progress via lucky guesses.

In comparison to the a priori *baselearn* estimate every other method takes a lower number of time steps due to not suffering the penalty for performing the diagnostic test. In terms of gain in the probability of knowing all skills, however, the "30+" condition is able to have considerably better results for average students while being almost as good for the faster learning students. The superiority of the a priori method is mostly shown there, as the optimal parameters for the best students are quite different from the average parameters which are used until the reestimation condition is met (Table 1). This necessarily leads to a period of time where suboptimal parameters are used, which has a particularly large impact in this case since the a priori method uses the same potentially optimal parameters from the beginning of the proposal of ZPD activities.

In order to obtain a better idea of how these methods behave on average, the mean and standard deviation for all methods was plotted using the values obtained for every student profile.

The results, visible in Figure 3, show that the "30+" condition has both a higher mean and lower standard deviation on average than every other method, as well as showing that the average parameters have a high standard deviation value. This proves that the average parameters are quite suboptimal for most students. It is worth mentioning that every method ensured an average probability of knowing all skills of at least 90%, lower than the 97% threshold. This was expected since that threshold was defined for perfect classification of student profiles as opposed to this case's occasionally wrong classification of random students which only use approximately optimal parameters even if classified correctly.

An additional comparison was also made using the mean and standard deviation for the average parameters, a priori and "30+" methods, grouping the profile students into those with low ($P_{0.1}$ to

Table 4: Average number of time steps and probability of knowing all skills, starting with default parameters and updating them dynamically at certain points through the model's execution. The conditions tested are performing the estimation after a certain number of time steps, after A1 has been removed from the ZPD, whenever A1 or B1 are removed from the ZPD and using the average parameters throughout

Condition/Profile	$P_{0.1}$	$P_{0.2}$	$P_{0.3}$	$P_{0.4}$	$P_{0.5}$	$P_{0.6}$	$P_{0.7}$	$P_{0.8}$	$P_{0.9}$
20 steps time	283.9	210.1	177.4	150.6	137.7	123.0	116.4	107.7	102.5
20 steps %skills	98.5	98.2	97.1	96.2	95.8	93.3	91.1	90.5	86.5
30 steps time	282.6	211.1	177.8	154.6	137.2	122.5	114.7	106.3	101.3
30 steps %skills	97.9	98.2	97.4	96.8	94.0	93.3	92.5	89.6	88.0
A1 time	285.0	216.2	182.4	157.0	141.2	125.7	116.9	106.3	101.0
A1 %skills	98.5	98.1	96.8	96.0	94.5	92.9	88.4	88.5	86.1
A1/B1 time	285.0	223.4	181.0	162.4	143.7	130.4	120.3	114.5	107.0
A1/B1 %skills	99.0	98.0	96.9	95.1	92.5	89.2	86.5	85.0	81.7
Avg params time	284.2	211.8	181.2	157.5	140.6	129.6	120.2	114.3	108.7
Avs params %skills	98.0	97.9	96.8	95.9	91.4	89.2	84.2	81.8	77.1
30 steps+ time	282.5	208.6	173.9	147.4	130.9	117.2	109.0	101.9	95.3
30 steps+%skills	97.5	98.5	97.4	97.7	95.9	96.8	94.1	91.5	90.9

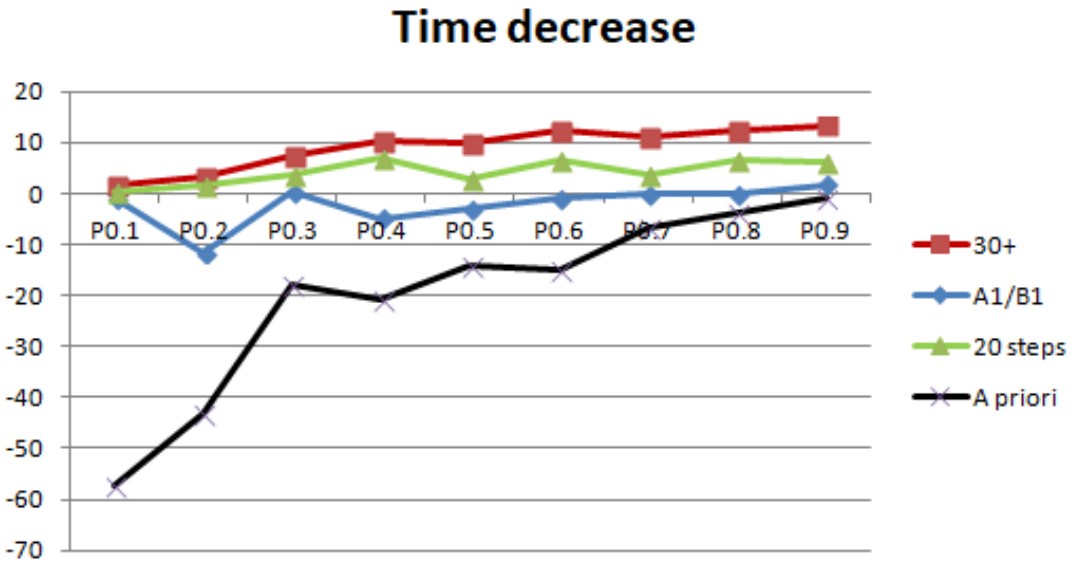

Figure 1: Time gain by different conditions and a priori estimation in comparison to using average parameters from the start

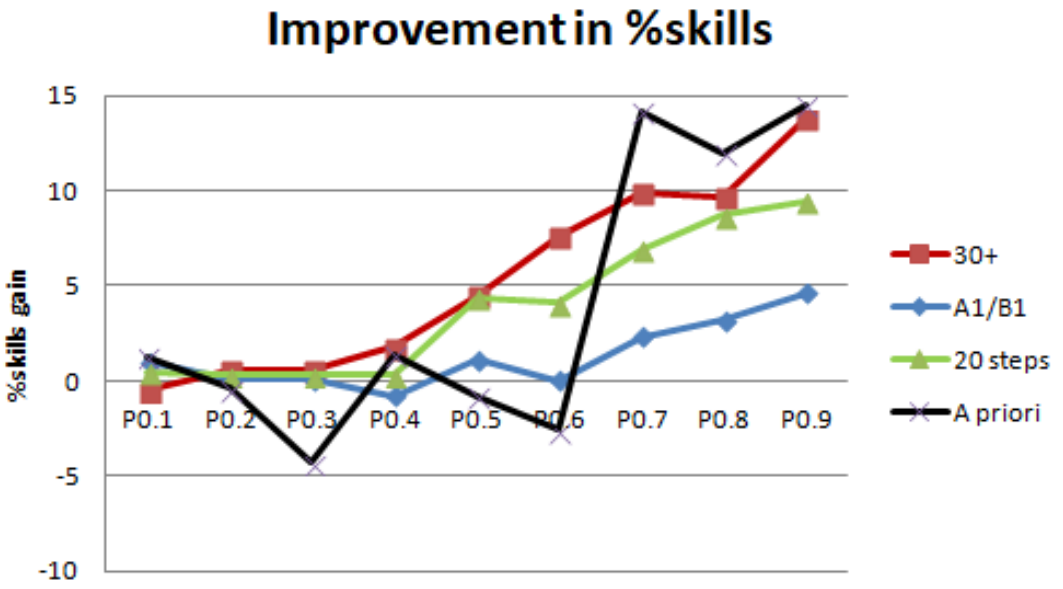

Figure 2: Gain in the average probability of knowing all skills by different conditions and a priori estimation in comparison to using average parameters from the start

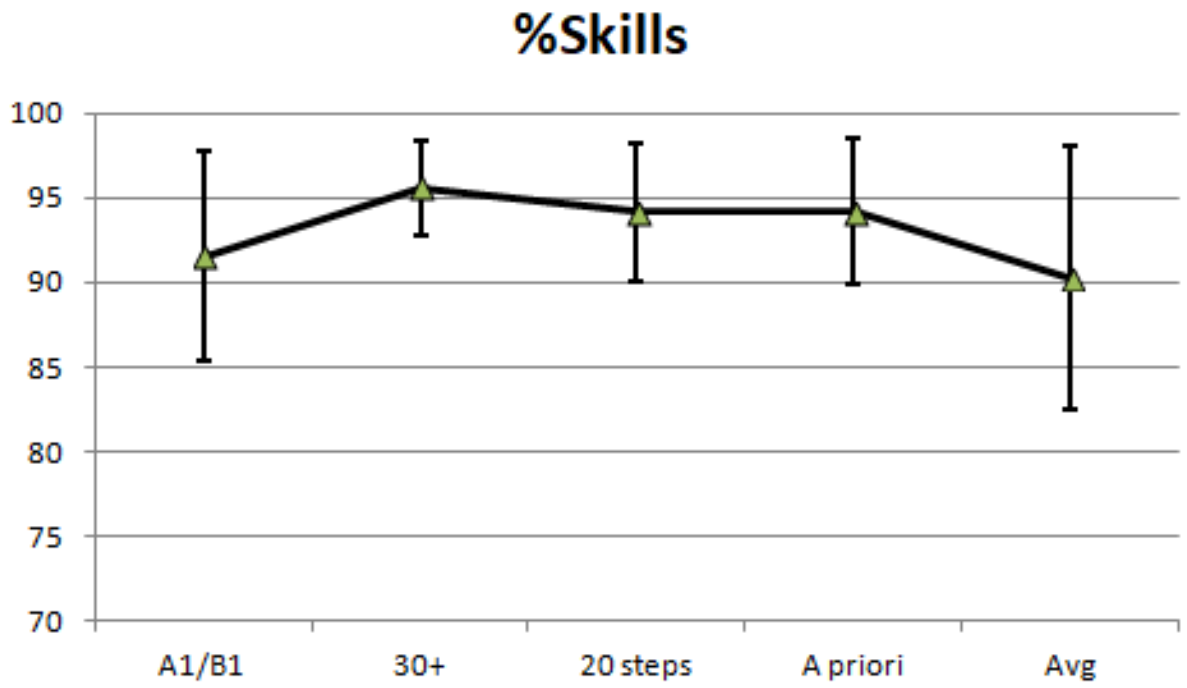

Figure 3: Mean and standard deviation for the average probability of knowing all skills for each reestimation condition

$P_{0.3}$), medium ($P_{0.4}$ to $P_{0.6}$) and high ($P_{0.7}$ to $P_{0.9}$) learning speed.

The results are shown in Figure 4, and while the probability of knowing all skills are all very similar for low *baselearn* students, the "30+" method clearly has the highest probability and lowest standard deviation for the students classified as "Medium". For the high *baselearn* group, however, the a priori method shows a higher average probability of knowing all skills. This happens due to the previously explained reason that faster learning students spend less time in the ZPD, meaning that the relative number of time steps that they are using the optimal parameters obtained by methods which require a certain number of time steps for their condition to trigger is greater than for slower students.

6. Conclusions

This work shows that it is possible to group students through the dynamic assignment of approximately optimal parameters based on the estimation of each student's learning capabilities, this approach only requires the estimation of one attribute per student and, although the computation of the Baum-Welch algorithm is somewhat slow, it can be used as each students performs an exercise without issue while also ensuring a relatively low number of time steps and a high probability of learning every skill that is being taught.

6.1. System Limitations and Future Work

It is very much possible that the behavior of real students cannot be fully captured within the attributes of the created student class. In particular, the adaptation of the *baselearn* attribute into reality is critical in allowing this model to be used with real students, along with the creating of an interface model. In the subject of the GOZPDES algorithm itself, since it is possible to preemptively adjust its parameters in order to fulfill certain thresholds in terms of number of time steps and probability of learning the taught skills, it may be interesting to create a mode of the model which allows the user to select a threshold for one of these measurements, with the program then automatically determining the optimal parameters for each student profile. Speaking of student profiles, an alternative clustering technique such as hierarchical clustering would have been interesting to attempt, as it would show an alternative approach to the problem.

Acknowledgements

The author would like to thank his supervisor, Professor Manuel Lopes, for his availability and patience.

This work was created as part of an investigation scholarship financed by the Fundação para a Ciência e a Tecnologia (FCT) and under the Seventh Framework Programme for Research and Technological Development (FP7) programme (FP7-ICT-2013-10610878).

References

[1] E. Altman. *Constrained Markov decision processes*. Chapman & Hall/CRC, 1999.

[2] W. J. Clancey. Intelligent Tutoring Systems: A tutorial Survey. page 58, 1986.

[3] B. Clement, D. Roy, P. Oudeyer, and M. Lopes. Multi-armed bandits for intelligent tutoring systems. *Journal of Educational Data Mining*, 7(2):1–22, 2013.

[4] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge, 1994.

[5] S. R. Eddy. Hidden Markov models. *Current Opinion in Structural Biology*, 6(3):361–365, 1996.

[6] J.-M. Francois. JAHMM: An implementation of hidden Markov models in Java, 2010.

[7] K. R. Koedinger, J. R. Anderson, W. H. Hadley, M. A. Mark, J. R. Anderson, W. H. Hadley, and M. A. Mark. Intelligent Tutoring Goes To School in the Big City. 1997.

[8] S. P. Lajoie and A. Lesgold. Apprenticeship training in the workplace: Computer-coached practice environment as a new form of apprenticeship. *Mach. Mediat. Learn.*, 3(November):7–28, 1989.

[9] T. K. Moon. The Expectation-Maximization Algorithm, 1996.

[10] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, 89:355–368, 1998.

[11] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, 1989.

[12] L. R. Rabiner and B. H. Juang. An introduction to hidden {M}arkov models. *IEEE Signal Proc. Magazine*, 3(1):4–16, 1986.

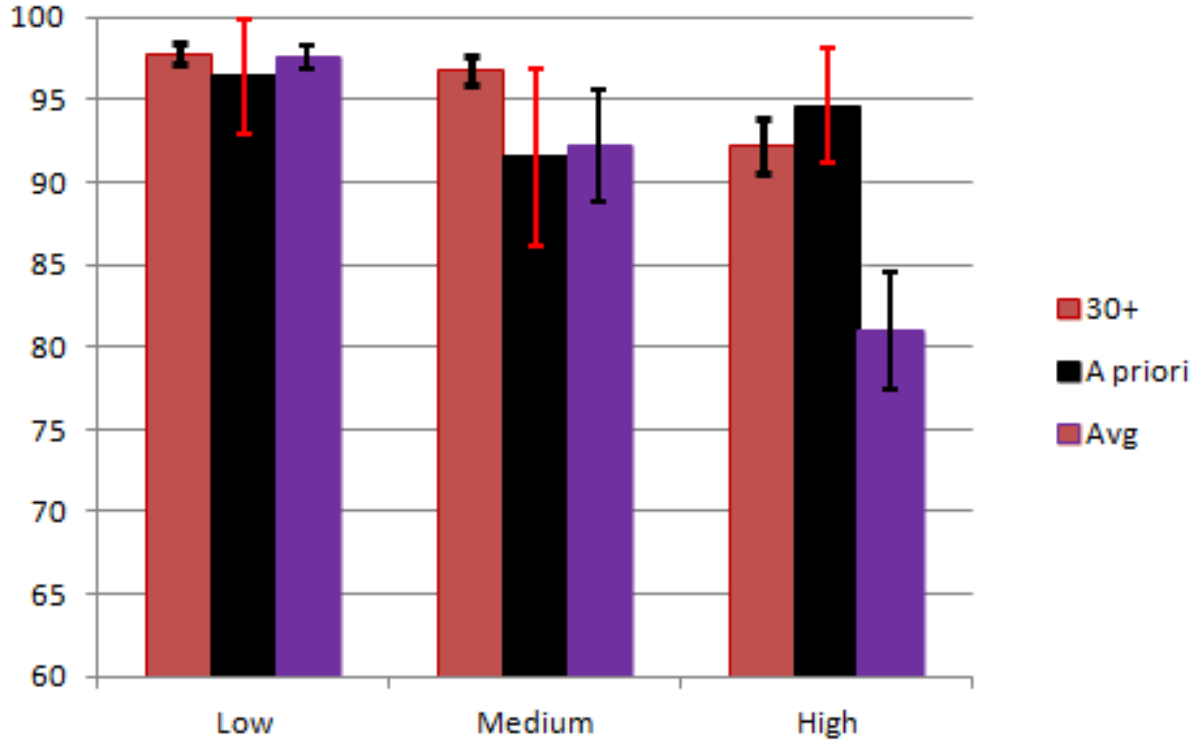

Figure 4: Comparison between the mean and standard deviation for the "30+"condition, the a priori method and using the average parameters for students with different *baselearn* values