ANALYSIS OF LEARNING CURVES FOR WEIGHTED CONSTRAINT-BASED TUTORING SYSTEMS

Nguyen-Thinh Le and Niels Pinkwart
Clausthal University of Technology, Department of Informatics
{nguyen-thinh.le, niels.pinkwart}@tu-clausthal.de

ABSTRACT

Weighted constraints have been proven a useful means to enhance the quality of error diagnosis, e.g., in the domain of natural language parsing. In this paper, we propose an heuristic approach using constraint weights to analyze learning curves for weighted constraint-based tutoring systems as an addition (or alternative) to the commonly used analysis of pre-/post-test comparisons. Using a tutoring system for logic programming on the basis of weighted constraints as an example case, we report on the effectiveness of the system with respect to helping students solve logic programming problems. The study showed that the group learning curves developed positively as students were solving five exercises.

KEYWORDS

Evaluation, learning curves, intelligent tutoring systems, weighted constraints.

1. INTRODUCTION

Researchers have applied different methods to evaluate the effectiveness of tutoring systems. Most prominently, comparisons between pre-test and post-test knowledge (often in combination with the comparison of an ITS to other learning tools) have been used. Martin et al. (2005) proposed to use learning curves in addition to other common evaluation indicators (such as pre-/post-test learning gains) in order to confirm the contribution of tutoring systems. Typically, learning curves represent a measure of student's performance (e.g., errors, or time on task) as a function of the number of attempts (Gallistel et al., 2004). Thus, learning curves can be used to plot the development of students' skills over time. This paper focuses on two goals: First, we demonstrate how to deploy weight values of constraints to plot learning curves for weighted constraint-based tutoring systems. Second, we report on the development of students' programming skills while using a tutoring system for Logic Programming (INCOM) with respect to learning curves.

In the next section, we first describe the weighted constraint-based approach for ITS development in more detail and introduce briefly the system INCOM which has been developed based on this approach. In the third section, we describe the evaluation of the system based on learning curves and report the results. In the last section, we conclude with some remarks on the effectiveness of the system in terms of learning curves.

2. WEIGHTED CONSTRAINTS FOR ITS DEVELOPMENT

Constraint-based approaches have been employed for a variety of tasks, including diagnosing grammatical errors in natural language utterances (Menzel, 1988) and building intelligent tutoring systems for various domains such as SQL (Mitrovic et al., 2001). The tenet of the constraint-based approach is modeling a space of correct solutions instead of creating an expert model by enumerating possible correct solutions. However, this approach is not sufficient to evaluate the plausibility of hypotheses about different solution variants for a problem which can be solved in many ways. Constraints might be satisfied under the assumption of one solution strategy, but could be violated in the context of another strategy. If the dependence between constraints and a specific solution strategy is not explicitly modeled, diagnostic information which is derived from a violated constraint might be deceptive to the student, because the solution strategy the student intended to implement might not be the same as the one based on which constraints are checked (Martin, 2001: 43; Kodaganallur et al., 2006: 321; Woolf, 2009: 85).

In order to hypothesize the most plausible strategy underlying a solution, we need to evaluate the plausibility of hypotheses. This can be done by assigning a heuristic value to each constraint which is referred to as a constraint weight. Weighted constraints have their root in the probabilistic constraint satisfaction area and have been applied successfully to enhance the quality of error diagnosis, e.g., in natural language processing (Foth, 2007). The overall plausibility score for a hypothesis is calculated by aggregating all weight values of violated constraints. Here, two overall approaches are possible – an additive model (weights of all violated constraints are added) or a multiplicative model. We propose to use the multiplicative model to calculate the plausibility score and constraint weights are taken from the interval [0;1]. The value 0 indicates the weight for constraints which model the most important requirements. As compared to the additive model, the multiplicative approach has the advantage that it allows tracing whether most important constraint violations have contributed to the plausibility score.

We applied the weighted constraint-based approach to develop INCOM, a system which is intended to coach students solving homework assignments of a Logic Programming course. The system informs the student about possible errors occurring in her solution attempts and gives feedback to improve the solutions. As a tutoring model, the system INCOM employs a two-stage coaching strategy: it first guides the students to analyze a task and then supports them during the implementation of a solution. During the first phase, the system requests the student to analyze a given problem statement and to transform the analysis result into an adequate signature for the logic programming predicate to be implemented. If the specified predicate signature is appropriate, the student is allowed to enter the second stage, where she is asked to define a predicate. During both stages, the system generates hypotheses about the student's input (a predicate signature or an implementation) by matching it against a set of possible solution strategies coded into the system. Once the hypotheses have been generated, the system evaluates each hypothesis with respect to their plausibility using constraint weights. The hypothesis which has the highest plausibility score is considered the most plausible one, and the system thus assumes that the student followed the solution strategy that this hypothesis represents. Feedback to the student is then given with respect to this solution strategy. Details about the system and its underlying technology have been published in (Le & Menzel, in press).

3. EVALUATION

3.1 Study design

The goal of our evaluation for INCOM is to answer the question whether the system's feedback is useful for helping students solve logic programming problems. The evaluation has been carried out during regular classroom hours of a course in Logic Programming running at the Department of Informatics, University of Hamburg. Students were given credits for participating in the study but had the possibility to opt out. They were assigned randomly to two groups: a control and an experimental group. The balance of two groups has been established in terms of the students' achievement score of the preceding sessions. The study consisted of a pre-test (10 minutes), an experiment session (60 minutes) during which the control group used the SWI Prolog interpreter and the experimental group used INCOM to solve five exercises, and a post-test (10 minutes). The experiment exercises were collected from the homework and examinations of former years of the same course. Note that Exercise 1.1 and 1.2 belong to the same domain context (product inventory), and that Exercise 2.1, 2.2 and 2.3 require students to solve the same problem (return on investment) applying different solution strategies:

- Exercise 1.1: Define a predicate which calculates the current value of the product inventory.
- Exercise 1.2: Define a predicate to create a new product list according to the following rules: Value of products less or equal 3000$ will be raised 3% and value of products above 3000$ will be raised 2%.
- Exercise 2.1: The return of investing an amount of money at a constant yearly interest rate can be computed according to the following recursive rule: $B_i = B_i$ for i=0, and $B_i= (1+Z)*B_{i-1}$ otherwise, where B_i represents the return of an investment period of i years, and Z is the yearly interest rate. Please map the given recursive rule to a recursive Prolog predicate.
- Exercise 2.2: Define a non-recursive predicate with the same signature as in Exercise 2.1

- Exercise 2.3: Convert your solution for Exercise 2.1 into a tail recursive predicate, i.e., the recursive subgoal must be the last one in a clause body.

In the first study, Le, Menzel & Pinkwart (2009) reported that the experimental group (18 participants) outperformed their peers of the control group (17 participants) by an effect size of d=0.23 between conditions (yet, these results were not statically significant at the 5% level). The second study, which has been conducted in 2010 with 16 participants for each group, showed an even better performance of the experimental group than the first study: the experimental group outperformed the control group by an effect size of d=0.33. These effect sizes indicate an educational significance, i.e., something was learned due to the use of INCOM compared to a standard programming environment. In addition to comparing the learning gains between the experimental and the control group (which was detailed in a previous paper), we are interested in the question how effective our system was in helping 34 participants of the experimental group of both studies while solving logic programming problems. To answer this question, the next sections of this paper use learning curves to represent the error rate of each problem solving attempt over time.

3.2 Analysis method

For a learning curve analysis, researchers usually plot the learning curve by counting the number of errors occurred in a solution attempt to compute the error rate (Martin et al., 2005) or by counting the number of attempts required to solve a problem type (e.g., Koedinger & Mathan, 2004). In the approach proposed here, to compute the error rate for each solution attempt, we consider the severity of each error (i.e., the importance of the error in the context of a solution) in terms of the associated constraint weight. The error rate of each solution attempt is calculated based on the severity of all errors. This approach is feasible for tutoring systems that make use of weighted constraints.

Since each constraint is associated with a constraint weight, the error rate of each solution attempt can be accumulated according to the following formula: Error-Rate = log($\prod_{i=1}^{n} \mathbf{Wi}$), where Wi is the weight of the i-th violated constraint. If a solution is correct, i.e., no constraint is violated, then the error rate for this attempt is log(1)=0. If a solution contains many errors, then the product of the weights of the violated constraints can approximate 0, and thus the error rate will be negative and can converge to $-\infty$ as more and more constraints are being violated.

In this paper, the diagrams are plotted by the error rate on the y-axis and the number of solution attempts on the x-axis. Thus, the learning curve represents the development of the error rate over time. We hypothesize that the error rate of a student will decrease after a number of attempts due to the use of the ITS, i.e., the students learn. Here, a learning curve is meant a group curve which represents the averaged error rates of solution attempts over a group of students who solved the same problem. If (as in our scenario) students may need different numbers of attempts to solve the same problem, the overall group curve can be truncated if some students solve a problem with few attempts while other participants, who have a high error rate, still require more time (in terms of the number of attempts). This would result in a highly misleading group curve (Gallistel et al., 2004), since each data point is calculated based on a different number of students, with better students “dropping out” earlier and thus causing the curve to indicate a group performance that is much worse than it actually is). To address this effect, the diagrams in this paper are based on a dataset which includes the (realistic) assumption that the student who has finished solving a problem in some attempt would not make any errors in next attempts of the exact same problems. For instance, let us assume that a student achieves a correct predicate definition after his 2nd attempt, and thus the error rate for the 2nd attempt is 0. Let X be the highest number of attempts that another participant needs to achieve a correct predicate definition for the same problem. Then, in our approach, the first student would be included in the calculation of the group error rate for the attempts in the interval [3;X] with an error rate of 0. This approach results in smoother group curves, in line with the observation of Ritter and Schooler (2002) that the learning curve appears smoother when the data is averaged across subjects, problems or both.

3.3 Results

Figure 1 shows the positive development of learning curves of students during the analysis (slope=0.50, fit R2=0.99) and the implementation phase of Exercise 1.1 (slope=0.51, R2=0.89). This indicates that system’s feedback was effective at helping students analyze the problem of Exercise 1.1 and specify a predicate signature. We also notice that the learning curve for implementation decreases after x=4 (the fourth attempt). That is because there were five students who finished their implementation successfully at the fourth attempt. After this point, the development of the learning curve depends on the performance of the rest of the group who were not as good.

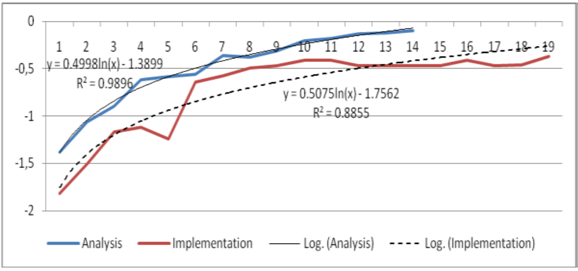

Figure 1. Learning curves during solving Exercise 1.1

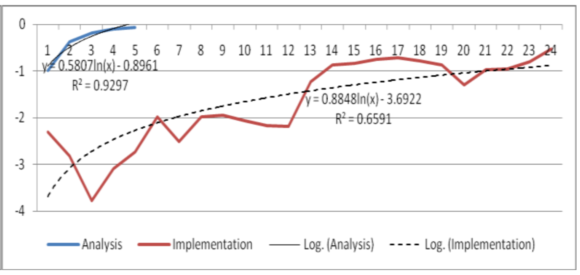

Figure 2. Learning curves during solving Exercise 1.2

Figure 2 represents the learning curves for Exercise 1.2. In overall, learning curves of students while solving Exercise 1.2 are better than those while solving Exercise 1.1: slope=0.58 (R^2=0.93) during the analysis phase, and slope=0.88 (R^2=0.66) during the implementation phase. This can be explained by the fact that Exercise 1.2 builds up on Exercise 1.1, and thus students may have had a better understanding when analyzing the problem of the same domain context. Indeed, students needed only 2.5 attempts to specify an appropriate predicate signature for Exercise 1.2, less than for Exercise 1.1 (5.6 attempts). We also notice that the learning curve during the implementation phase shows a decline between the 1st and the 3rd attempt. This effect in the figure is due to the fact that the error rate of a few students developed negatively after the first attempt. Thus, their performance strongly affected the learning curve during the implementation phase after the 1st attempt negatively. Due to the different development of performance among the students solving this exercise, the fit of the learning curve for the implementation phase is relatively low (R^2=0.66).

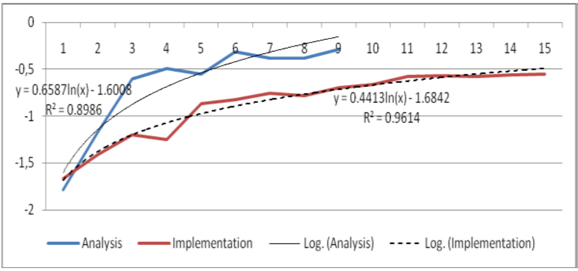

Figure 3. Learning curves during solving Exercise 2.1

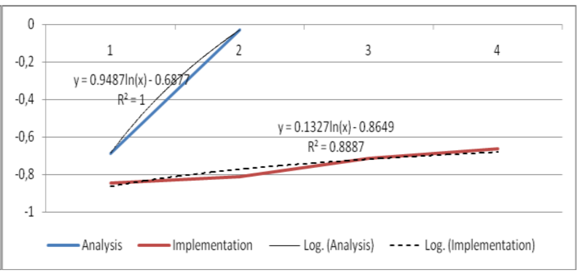

Figure 4. Learning curves during solving Exercise 2.2

Figure 3 visualizes the learning curves for Exercise 2.1. The mean slope of learning curves during the analysis phase is relatively steep (slope=0.66, R^2=0.90). This can be caused by the students having gained experience when solving Exercise 1.1 and 1.2. However, during the implementation phase, the mean slope of the learning curve is 0.44 (R^2=0.96), which is lower than the one for the previous exercises. This may be due to the fact that this exercise is considerably more complex than Exercises 1.1 and 1.2.

Figure 4 shows that the students' performance when specifying a predicate signature for this exercise (slope=0.95, R^2=1) is better than for Exercise 2.1, because students may have acquired experience when specifying the predicate signature for the previous exercise. However, the slope of the learning curve during the implementation phase (slope=0.13, R^2=0.89) is worse than the one for Exercise 2.1. This may be explained by the fact that the solution strategy, which is expected to solve Exercise 2.2 (this exercise explicitly requires an analytic formula to calculate the return on investment), is totally different from than the one required for Exercise 2.1 (recursive computation), so no knowledge transfer concerning the implementation can be expected.

Figure 5 represents the learning curves when solving Exercise 2.3: slope=1.24 (R^2=0.98) during the analysis phase and slope=0.62 (R^2=0.58) during the implementation phase. Remarkably, the development of the learning curves for this exercise is better than for Exercise 2.2 and 2.1. However, the fit of the curve during the implementation phase is relatively low. This can be explained by the fact that the number of participants, who attempted to implement a predicate for this exercise, is relative small (6 students). In addition, most of the participants could provide a correct predicate on the second attempt, but there was one participant who could not construct a correct implementation even after twelve attempts.

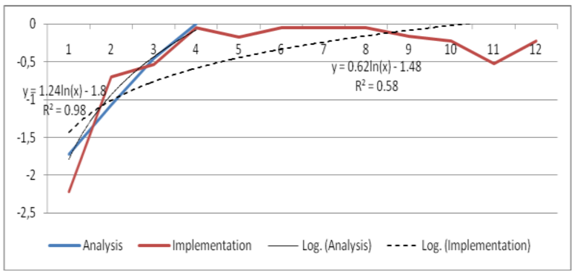

Figure 5. Learning curves during solving Exercise 2.3

4. CONCLUSION

In this paper, we have shown how constraint weights can be used to compute error rates for student solution attempts in a constraint-based Intelligent Tutoring System. We have conducted an evaluation study and used learning curves to analyze the learning performance of students using the homework assistance system for learning logic programming. We could determine that the slope of group learning curves when solving exercises using the system is positive and most learning curves fit well in logarithmic functions. This agrees with our hypothesis that system's feedback is useful to help students solve logic programming exercises. Two learning curves (during the implementation for Exercise 1.2 and 2.3) do not have a good fit due to the diversity of performance among students. We also conclude that the slope of learning curves during the analysis phase becomes steeper over time, indicating that students learn faster. It remains open whether this approach is more realistic at showing student's knowledge acquisitions than merely counting the number of errors occurring in a solution attempt, as traditional learning curve analysis approaches would foresee. Comparative analyses would be required to answer this question.

REFERENCES

Foth, K. A., 2007. *Hybrid Methods of Natural Language Analysis*. Shaker Verlag, Germany.

Gallistel, C. R. et al., 2004. The learning curve: Implications of a quantitative analysis. *Proceedings of the National Academy of Sciences of the USA*, Vol. 101, No. 36, pp. 13124-13131.

Kodaganallur, V. et al., 2006. An assessment of constraint-based tutors: A response to Mitrovic and Ohlsson's critique of "A comparison of model-tracing and constraint-based intelligent tutoring paradigms". *International Journal of Artificial Intelligence in Education*, Vol. 16, pp. 291-321.

Koedinger, K.R. and Mathan, S., 2004. Distinguished qualitatively different kinds of learning using log files and learning curves. *Proceedings of ITS 2004 Log Analysis Workshop, Maceio, Brazil*. pp. 39-46.

Le, N.-T. and Menzel, W. Using weighted constraints to diagnose errors in logic programming - The case of an ill-defined domain. *Int. Journal of Artificial Intelligence in Education - Special Issue on ill-defined domains*, in press.

Le, N.-T., Menzel, W., and Pinkwart, N., 2009. Evaluation of a constraint-based homework assistance system for logic programming. Proceedings of *the 17th International Conference on Computers in Education*.

Martin, B., 2001. Intelligent Tutoring Systems: The practical implementation of constraint-based modeling. *Ph. D. thesis*, University of Canterbury.

Martin, B. et al., 2005. On using learning curves to evaluate ITS. *Proceedings of the conference on AIED*. pp. 419-426.

Menzel, W., 1988. Diagnosing grammatical faults - a deep-modelled approach. *Proceedings of AIMSA*, pp. 319-326.

Mitrovic, A. et al., 2001. Constraint-based tutors: A success story. *Proceedings of the 14th Int. conf. on Industrial and engineering applications of AI and expert systems.* pp. 931-940.

Ritter, F.E. and Schooler, L. J. (2002). The learning curve. In *International encyclopedia of the social and behavioral sciences. 8602-8605. Amsterdam: Pergamon.*

Woolf, B. P, 2009. *Building intelligent interactive tutors*. Morgan Kaufmann.