Prediction of Student Actions Using Weighted Markov Models

Xiaodi Huang[1], Jianming Yong[2], Jiuyong Li[3], Junbin Gao[4]
1. School of Business and Information Technology, Charles Sturt University
2. School of Information Systems, University of Southern Queensland
3.School of Computer and Information Science, University of South Australia
4.School of Accounting and Computer Science, Charles Sturt University
xhuang@csu.edu.au, yongj@usq.edu.au, jiuyong.Li@unisa.edu.au, jbgao@csu.edu.au

Abstract

The Markov model has been applied to many prediction applications including the student models of intelligent tutoring systems. In this paper, we extend this well-known model to the weighted Markov model, and then apply it to student models in order to predict student behaviors. The prediction using our models is based not only on the frequency of collective behaviors of previous users, but also on the degrees of the relations between the predicted user and others. In doing so, a novel way is presented to quantify the similarities between previous students and the current active student. These similarity scores will be used as weights in the weighted Markov model.

1. Introduction

As personalization and recommendation functionalities have become integral part of intelligent information systems, particularly for the e-commerce systems, user modeling has recently received a great deal of attention [2, 3, 7]. Similarly, intelligent tutoring systems (ITS) are capable of providing effective instruction through their adaptiveness and personalization. This personalization cannot be achieved without its critical component -- the student model [4, 5, 6, 7, 9]. As one type of user models, student models analyze students' behaviours, and predict their future behaviours, as well as learning performance. In particular, relying on knowledge about students contained in learning profiles, the models deduce what the students want and predict what actions they will take in the next step. Characterizing the students, the profiles form compact representations, which can be leaned from the training data, by maintaining the results from the students' interaction with the system.

Apart from the student model, an ITS contains an expert module that contains the domain knowledge of the system, and a pedagogical module that chooses appropriate teaching strategies. As the core component of an ITS, the student model plays a central role in an ITS. The vital issue of a student model is how to make the predictions of students using their profiles.

Several existing systems employ machine-learning techniques [4, 8] to discover student knowledge behind the descriptions of the user's behavioral patterns. Substantial research [5, 10, 11, 12] has been devoted to using probabilistic reasoning frameworks to deal with the inherent uncertainty of the student actions and goals. A Bayesian network, for example, is used to predict what a student will do next after each action. Learning systems normally store detailed traces of students' activities, thus producing huge sets of training data.

As being utilized for studying and understanding stochastic processes, Markov models [1] have been shown to be well suited for modeling and predicting a user's behavior in different applications. In the context of student models, the input is the sequence of actions taken by a student, and the goal is to build Markov models by which to predict the action that the student will most likely take next.

In many applications, lower-order Markov models are not very accurate in predicting the user's behavior, since these models do not look far into the past to correctly discriminate the different observed patterns. As a result, higher-order models are often used. Unfortunately, these higher-order models have a number of limitations associated with high state-space complexity, reduced coverage, and sometimes overall prediction accuracy.

In this paper, we will present a technique for effectively combining observed patterns of users into lower-order Markov models so that the coverage and

the accuracy are basically retained without using the higher-order Markov models. The key idea behind our technique is that observed patterns of different students that are to be utilized in higher-order models are implicitly be combined into lower-order models in the forms of various weights with respect to the action pattern being predicted. In particular, we present a weighted Markov models that weight the similarities between observed users and current users when calculating the next-selected possibility of each candidate state. The users who are observed to have similar behaviour patterns to the current user's will have more influence than others on predicting this user's behaviour. In a word, we employ the lower-order models without affecting the performance of the overall scheme.

Even though our approaches are developed in the context of student models, they are suitable for prediction in different applications as well.

The rest of this paper is organized as follows. Section 2 presents an overview of the problem of predicting a student next-action using Markov models. A detailed description of our weighted Markov models is presented in Section 3. Section 4 provides an approach to computing the weights, followed by an example in Section 5. After related work is presented in Section 6, Section 7 offers some concluding remarks.

2. Markov Models for Predicting the Student Next-Action

In this section, we present the problem of the student next-action predication. We assume that there is a set of m students $S = \{s_j : 1 \le j \le m\}$, and a set of n actions $A = \{a_j : 1 \le j \le n\}$. Active student $s \in S$ who is now in the state Ai, is referred to as the student whose next action through A needs to be predicted.

Definition 1 Action graph: a directed graph that models all possible actions and their relations within an ITS system. The nodes represent actions, and the edges indicate the sequence relations between these actions. An action graph is denoted as $G = (A, E)$, where A is a set of actions, and E is a set of their relations.

Note that the actions in this paper refer to either knowledge points in domain knowledge networking or help actions. As an example, a set of actions can be A= {fact 1, principle 3, example 2, supplementary materials 1.2}. An action graph is depicted as Figure 1 where we have $A = \{a_1, a_2, a_3, a_4, a_5, a_7\}$.

Definition 2 Learning path: a sequence of actions $< A_l, A_{l-1}, \cdots, A_1 >$ by which a student has chosen during interacting with an ITS at different stages.

As a subset of G, a learning path refers to action paths which either previous students or the active student has chosen so far. Different students may have various learning paths. It is likely that students with similar backgrounds may choose the similar learning paths. As illustrated in Figure 1, the learning paths of six prior students are denoted as s_1, s_2, s_3, s_4, s_5, and s_6, respectively. These learning paths result from students traversing through the action graph, each beginning with the corresponding student's first action, and terminating with his/her most recent action.

ITS systems normally keep record of learning paths of different students, which consist of history training data.

Definition 3 Problem of student next-action predication: Given the history data of different students and an active student using an ITS system, we will predict the next action of the active student, based on the data and the actions that the student has taken so far.

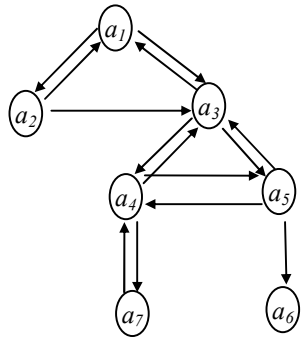
The next-action prediction problem can be solved using a probabilistic framework as follows. Let $\Omega \subseteq G$ be a student's learning session of length l (i.e., it contains l actions), and let $P(a \mid \Omega)$ be the probability that the student takes next-action a. Then, the action al+1 that the student will take next is given by

$$\begin{aligned} a_{l+1} &= \arg\max_{a \in A} \{P(A_{l+1} = a \mid \Omega)\} \\ &= \arg\max_{a \in A} \{P(A_{l+1} = a \mid A_l, A_{l-1}, \cdots, A_1)\} \end{aligned} \quad (1)$$

where A is the total set of actions being present on the action graph. Essentially, for each action a, this approach computes its probability of being taken next, and then selects the action that has the highest probability.

The key step in determining al+1 from Equation 1 is to be able to compute the various conditional probabilities given the student's current learning session Ω. In general, it is not feasible to accurately calculate these conditional probabilities because (i) the learning sessions can be arbitrarily long, and (ii) the size of the available training set is not big enough to accurately estimate them for long learning sessions. For this reason, the conditional probabilities are commonly estimated by assuming that the sequence of actions taken by the student follows a Markov process. This implies that the probability of taking an action a does not be computed from all the actions in the learning session, but only from a small set of k preceding actions, where $k < l$. Using the Markov process assumption, Equation 1 for predicting the action al+1 that the student will visit next is simplified by

$$a_{l+1} = \arg\max_{a \in A}\{P(A_{l+1} = a \mid \Omega)\} = \arg\max_{a \in A}\{P(A_{l+1} = a \mid A_l, A_{l-1}, \cdots, A_{l-(k-1)})\} \quad (2)$$



Training set

$s_1 :< a_1, a_3, a_4, a_3 >$

$s_2 :< a_1, a_2, a_3, a_5, a_6 >$

$s_3 :< a_1, a_2 >$

$s_4 :< a_3, a_4, a_3, a_2 >$

$s_5 :< a_1, a_2, a_3, a_5, a_3 >$

$s_6 :< a_1, a_2, a_3, a_2, a_1, a_3, a_4, a_5 >$

Test set

$s :< a_1, a_2, a_3, a_3, a_4, ? >$

Figure 1: An action graph, training and test sets

In order to use the *k*th-order Markov model, we need to learn Al+1 for each learning path of k actions, $S_j^k =< A_{l-(k-1)}, A_{l-(k-2)}, \cdots, A_l >$, where $j \in S$, by estimating the various conditional probabilities, $P(A_{l+1} = a \mid A_l, A_{l-1}, \cdots, A_{l-(k-1)})$ of Equation 2.

Using the maximum likelihood principle, the conditional probability is computed by

$$P(A_{l+1} = a \mid S_j^k) = \frac{\sum_j n_j(S_j^k, a)}{\sum_j n_j(S_j^k)} \quad (3)$$

where $n_j(S_j^k)$ amounts to the number of times that the sequence S_j^k was chosen, and $n_j(S_j^k, a)$ the number of times action a was chosen immediately after S_j^k, both by student j $(1 \le j \le m)$ in the training set. In other words, the conditional probability equals the ratio of these two frequencies. When calculating the frequency of Sk, it is in fact counting the number of students who visited this path sequence.

Based on Equation 2, among all candidate actions, the action with the highest possibility will be predicted as the next action of the active student.

Of the different variations of Markov models [1], it is generally found that higher-order Markov models display high predictive accuracies on various applications that they can predict. As mentioned before, however, we use the k-th order Markov model instead of the original l length one because of infeasibility of determining the conditional probabilities of large number of states. In practice, one normally employs the 2-th order Markov model. This simplification comes at a price of making use of only part of information in the training data. The data beyond 2-th order sessions, which is the learning paths with the length of (l-2), has totally been ignored. One of our contributions in this paper is to utilize this useful higher-order information in such a way that does not increase much the space and run-time requirements.

3. Weighted Markov Models

The next-action prediction problem using the Markov model is basically based on the following idea. From Equation 1, from the candidates of the current state, the active student is predicted to choose the next-action that has already been taken by the highest proportional number of prior students. In other words, the higher number of students has chosen an action, the more likely the active student will follow. The prediction result is in principle rooted in the voting, regardless of what kinds of students have made the votes.

The problem of using the Markov models in student models is that it completely ignores the differences between students. It is natural that a student will likely choose the learning path that is more similar to that chosen by those who exact more influence on this particular student. That is to say, we should not only consider the number of a particular action being chosen, but also take into account the influence of the persons who have made such choices. In particular, we will assign a higher weight to the students who have more influence on the active student. Based on the learning path this student has already chosen, we will develop an approach to calculating the similarity values between this student and others. By considering these factors, Equation 3 for computing the probability is adjusted accordingly:

$$P(A_{l+1} = a \mid S_j^k) = \frac{\sum_j w_j n_j(S_j^k, a)}{\sum_j n_j(S_j^k)} \quad (4)$$

That is, the number of the learning paths that student j visited is weighted by the different weight w_j of each student. The conditional probability is now regarded as the ratio of weighted frequencies. Unlike Equation 3, Equation 4 differentiates who has visited the learning paths, not just counting the number of visited paths.

The implications of Equation (4) are at least of twofold: (1) the more number of previous students have chosen an action, the more likely the active student will do the same one. This is as same as the original philosophy of Markov models; and (2) the active student will more likely take the next action that was chosen by the students who are similar to her/him in terms of choosing the similar past learning paths.

Using Equation (3) of the normal Markov model, the predicted action will always be the same one if the number of students following the leaning path $< S_j^2, a >$ remains unchanged, regardless of which students chosen these paths. From Equation (4), in contrast, we can conclude that different active student s may have different results of next-action prediction even though there exists the same data $< S_j^2, a >$ in the training data set S_j^k .We also note that Equation (3) is equivalent to be Equation (4) if all weights w_j are 1s.

4. An Approach for Calculation of Weights

Up to this point, we have presented our weighted Markov model. The question of how to calculate the weights remains unsolved.

The weights in our model reflect the degrees of similarities between the active student and other students in S who have used the system. In the following, we will present an approach to computing such similarities.

It is assumed that the active student s has chosen the l-length learning path (l> 2) $< S_j^k, p_{il} >$ when we predict her next probable action. Due to the computational complexity, we normally use the lower order Markov model as mentioned before. In this paper, we employ the second-order Markov model; that is, $< S_j^2, a >$. This means the action sequences occurring before the latest two actions ($\Omega - S_j^2$) are ignored. However, we make use of these sequences by using the similarity score.

The basic idea of measuring student the similarity is (1) to represent a student as a sequence of actions that have been taken; (2) to find the longest-common-sub-sequence (LCS) through the best matching between two sequences using dynamic programming techniques; and (3) to calculate the similarity value between the action sequence of the active student and the LCS.

4.1. Student represented as action sequences

In order to quantify the similarity between students, we must choose features to characterize a student. Students in ITS systems are different from each other in that each one chooses various actions, and the sequences of these actions, and their lengths. One difference between our similarity measure and others: we consider a learning session as a sequence rather than a set. For example, Jaccard coefficient, $sim(S_1, S_2) = | S_1 \cap S_2 | / | S_1 \cup S_2 |$ is widely used. We argue that a sequence can better represent the nature of a learning session than a set. For example, using Jaccard Coefficient similarity measure there is no difference between three learning sessions $< a_1, a_2, a_3, a_4 >$, $< a_2, a_3, a_1, a_4 >$ and $< a_1, a_2, a_4, a_3 >$. Intuitively, students who study with "a_1", then with "a_2", are different from those who study "a_2" directly, because "a_1" may be the pre-requisite knowledge for "a_2". In ITS systems, it matters in terms of pedagogies what kinds of actions a student chooses and its chosen sequence.

How to define a learning session for students? We can pre-assign a time window size, say T. For comparison we select all learning paths with the time interval T. This is based on the fact each action is associated with a timestamp.

TimeStamp (the last action in a learning path) = timestamp (the first action in the learning path) + T

The problem of calculating the similarity score is equivalent to finding the longest common subsequence between two sequences

4.2 Finding the longest-common-subsequence

This problem is referred to as the longest-common-subsequence problem [13]. We rephrase the problem as follows: Given a learning sequence $A =< a_1, a_2, \cdots, a_m >$ by the active student, another sequence $Z =< z_1, z_2, \cdots, z_k >$ is the sub-sequence of A if there exists a strictly increasing sequence $< i_1, i_2, \cdots, i_k >$ of indices of A such that for all $j = 1,2, \cdots, k$, we have $a_{ij} = z_j$. Given two learning sequences A and B, we say that a sequence Z is a common subsequence of A and B if Z is a subsequence of both A and B. For examples, if $A =< a_1, a_2, a_3, a_2, a_4, a_1, a_2 >$ and $B =< a_2, a_4, a_3, a_1, a_2, a_1 >$, the LCS of A and B are

$< a_2, a_3, a_2, a_1 >$ or $< a_2, a_4, a_1, a_2 >$ with the length of 4. rather than being $< a_2, a_3, a_1 >$.

4.3 Calculating the similarity score

The similarity between two sequences can be quantified by the number of changes needed to turn one into another. This idea is similar to the well-known the edit distance that describes how many fundamental operations such as insertions and substitutions are required to transform one string into another. We therefore arrive at

$$w(A,B) = \frac{|\, LCS(A,B)\,|}{|\, A\,|}$$

In the previous example, we have $A =< a_1, a_2, a_3, a_2, a_4, a_1, a_2 >$ $LCS(A,B) =< a_2, a_3, a_2, a_1 >$ or $< a_2, a_4, a_1, a_2 >$. So we have $w(A, B) = 4/7$. It is obvious that $w(A,B)$ ranges in [0,1]. That is, when A is a subset of B, their similarity reaches the maximum value 1.

Like other approaches, the application of weighted Markov models cannot avoid the cold-start problem. When the active student starts to use the system, there is no an established profile including the chosen action sequences associated with the student. Therefore, the similarity scores cannon be calculated. A straightforward way of overcoming this problem is to utilize the normal Markov models where all weights are set to the same values.

As the active student continues using the system, his/her learning paths will become longer and longer. The predication of next-action is more accurate by using this longer sequence for computing the similarity scores. In addition, the more accurate the predication result will be, the later actions being taken by the active student is utilized. This suggests that the weighs should take into account the length of learning paths, and the time of these actions taken. Formally, we have the following equation

$$w'(A,B) = \frac{l}{k}(1 + e^{-t})w(A,B)$$

where l is the window size as the length of learning paths, and t the latest timestamp associated with the action of the learning path. Obviously, the similarity score is damped exponentially by time, while being increased linearly by the window size. When the learning paths are formed long time ago, i.e., $t \to \infty$, and k=2 (the second-order model), we have $w'(A,B) = w(A,B)$.

5. An Example

Comparing the Markov model to our weighted Markov model, we provide an illustrative example in Figure 1 in this section.

5.1 The Markov model

As depicted in Figure 1, the history data are $S = \{s_j : 1 \le j \le 6\}$ which records the learning path of prior six students. The action set is $A = \{a_j : 1 \le j \le 7\}$. The active student s is currently in state a_4. The *problem of student next-action predication* of this particular case is to predict which action the active student most probably chooses from three possible candidate actions a_3, a_5 or a_7. According to Equation 3, we have from the training data

$$P(A_{l+1} = a_3 \mid a_4, a_3) = \frac{1(w_1) + 1(w_4)}{1(w_1) + 1(w_2) + 1(w_4) + 1(w_5) + 1(w_6)} = 0.40$$

$$P(A_{l+1} = a_5 \mid a_4, a_3) = \frac{1(w_6)}{1(w_1) + 1(w_2) + 1(w_4) + 1(w_5) + 1(w_6)} = 0.20$$

and

$$P(A_{l+1} = a_7 \mid a_4, a_3) = 0$$

According to Equation 2, we have

$$a_{l+1} = \arg\max\{P(A_{l+1} = a_3 \mid a_4, a_3), P(A_{l+1} = a_5 \mid a_4, a_3), P(A_{l+1} = a_7 \mid a_4, a_3)\} = a_3$$

The action a_3 will be predicted.

5.2 Weighted Markov model

We first compute the similarity score between active student s and s_1, s_4 and s_6, respectively:

$$w_1 = \frac{|<a_1>|}{|< a_1, a_2, a_3 >|} = \frac{1}{3}, \quad w_4 = \frac{|<a_3>|}{|< a_1, a_2, a_3 >|} = \frac{1}{3},$$

$$w_6 = \frac{|< a_1, a_2, a_3 >|}{|< a_1, a_2, a_3 >|} = 1$$

According to Equation 4, we then have

$$P(A_{l+1} = a_3 \mid a_4, a_3) = \frac{w_1 \times 1(w_1) + w_4 \times 1(w_4)}{1(w_1) + 1(w_2) + 1(w_4) + 1(w_5) + 1(w_6)} = 0.13$$

$$P(A_{l+1} = a_5 \mid a_4, a_3) = \frac{w_6 \times 1(w_6)}{1(w_1) + 1(w_2) + 1(w_4) + 1(w_5) + 1(w_6)} = 0.20$$

and

$$P(A_{l+1} = a_7 \mid a_4, a_3) = 0$$

Finally, we predict that the next-action of the active student is the action a_5 instead of a_3:

$$a_{l+1} = \arg\max\{P(A_{l+1} = a_3 \mid a_4, a_3), P(A_{l+1} = a_5 \mid a_4, a_3), P(A_{l+1} = a_7 \mid a_4, a_3)\} = a_5$$

Examining the leaning paths of S_6 and s, we should not surprise this prediction result because they have much common in the learning paths.

6. Related Work

Several improvements on the Markov models have been proposed. Cadez et al.[2] utilize Markov models for classifying browsing sessions into different categories. Users with similar navigation patterns are grouped into the same cluster using a clustering approach, and each cluster is then represented by a Markov model. Deshpande et al. [1] propose techniques for combining different order Markov models to obtain low state complexity and improved accuracy. The method starts by building the all *k*th-order Markov model, and employs three schemes to eliminate states that have low prediction accuracy. Our approach starts with a low-order Markov model, and effectively makes use of the useful information included in the higher-order model.

7. Conclusion

In this paper, we have presented a weighted Markov model that distinguishes the users in training data for computing the probabilities of states. By doing so, the observed patterns occurring in higher-order models can effectively be used in lower-order models without increasing computational complexity. A novel way of computing the weights in the context of student models was also presented. The future work includes experiments on the real data.

8. References

[1] M.Deshpande, and G.Karypis, "Selective Markov Models for Predicting Web Page Accesses", ACM Transactions on Internet Technology, Vol.4, No.2, May 2004, pp.163-184.

[2] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Visualization of navigation patterns on a web site using model based clustering", In Proceedings of the Sixth International KDD conference, 2000, pp. 280-284.

[3] M. Bauer, "A Dempster-Shafer approach to modeling agents preferences in plan recognition", User Modeling and User-Adapted Interaction, 5(3-4), 1995, pp.317-348.

[4] J. Freyberger, N.T. Heffernan, and C. Ruiz, "Using Association Rules to Guide a Search for Best Fitting Transfer Models of Student Learning," Proc. Workshop Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes at the 7th Ann. Intelligent Tutoring Systems Conf., 2004

[5] D.W. Albrecht, I. Zukerman, and A. E Nicholson, "Bayesian Models for Keyhole Plan Recognition in an Adventure Game" User Modeling and User-Adapted Interaction, 8(1-2), 1999,pp.5-47.

[6] P. Albacete, and K. VanLehn, "The conceptual helper: An intelligent tutoring system for teaching fundamental physics concepts", Proceedings of Intelligent Tutoring Systems, ITS2000, Montreal, Canada, Lecture Notes in Computer Science 1839, Springer, 2000, pp.564-573.

[7] Antonija Mitrovic, Brent Martin, and Pramuditha Suraweera, "Intelligent Tutors for All: The Constraint-Based Approach" IEEE INTELLIGENT SYSTEMS, JULY/AUGUST 2007, pp 38-45,

[8] R. Sison and M. Shimura, "Student Modeling and Machine Learning," Int'l J. AI and Education, vol. 9, 1998, pp. 128–158.

[9] F. Esposito, O. Licchelli, and G. Semeraro, "Discovering Student Models in E-learning Systems," J. Universal Computer Science, vol. 10, no. 1, 2004, pp. 47–57.

[10] M. Mayo and A. Mitrovic, "Optimizing ITS Behavior with Bayesian Networks and Decision Theory," Int'l J. AI and Education, vol. 12, 2001, pp. 124–153.

[11] C. Conati, A.S. Gertner, and K. VanLehn, "Using Bayesian Networks to Manage Uncertainty in Student Modeling," User Modeling and User-Adapted Interaction, vol. 12, no. 4, 2002, pp. 371–417.

[12] S. Suebnukarn and P. Haddawy, "A Bayesian Approach to Generating Tutorial Hints in a Collaborative Medical Problem-Based Learning System," Artificial Intelligence in Medicine, vol. 38, 2006, pp. 5–2.

[13] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest.4, Introduction to Algorithms, McGraw-Hill, July, 2001.