One Decision Tree is Enough to Make Customization

Hao Wan
Worcester Polytechnic Institute
Worcester, USA
hale@wpi.edu

Joseph E. Beck
Worcester Polytechnic Institute
Worcester, USA
josephbeck@wpi.edu

ABSTRACT
The ability to customize instruction to individuals is a great potential for adaptive educational software. Unfortunately, beyond mastery learning and learner control, there has not been much work with adapting instruction to individuals. This paper provides an approach to determine what type of learner does best with a different intervention. We focused on constructing a decision tree that discriminated difference between tutoring interventions, and thus to make customization for each student. We evaluated our model on simulated and on real data. In the simulated data set, it outperformed other methods and the constructed models captured a pre-defined customization structure. With the real data, the customized learning approach achieved stronger learning gains than simply picking the best overall teaching option. Surprisingly, it was difficult to outperform a decision tree that simply used how quickly students tended to learn a skill. That is, more features and more complex models did not result in a more effective system.

Author Keywords
Decision tree; customization; treatment effect

ACM Classification Keywords
Algorithms

INTRODUCTION
Even since Bloom's paper showing a 2 standard deviation effect size of individual tutoring [1], computer's ability to adapt instruction has been pitched as a solution. However, most educational software does little adaption to the individual beyond mastery learning and learner control. Each student (largely) sees the same help messages and the same instruction. Mastery learning enables students to keep practicing a skill until they have mastered it. Learners maintain a degree of control, such as selecting which story to read next or which type of help to receive (e.g., [2]). Those two approaches overlook the fact that students' learning outcomes might differ in tutoring interventions, because they have different learning styles [3, 4], prior knowledge [5, 6], or other factors that affect which type of instruction is most effective for *this* learner.

In this paper, we explore finding which intervention works best for what type of learners. This can be divided into two sub-problems: identify the different "types" of learners; estimate the outcome for a type of learner in each intervention. In the work [4], the authors state that the tutorial methods are effective if and only if the optimal tutorial method differs in different types of learners, which is called "acceptable evidence". Likely, given a set of students, we separate them into groups, such that the optimal intervention differs in the groups. In this work, we use decision tree algorithm to solve the two sub-problems. In a constructed decision tree, we consider the students fall into a leaf node are in the same group.

Unlike traditional classification problem, where every student is marked with which option is optimal for him/her, we focus the problem where we do not know *a priori* how each student best learns. Even worse, we only know how a student would perform with one of the possible options, this is very likely to happen in the randomized control experiments. Therefore, we must find commonalities in what types of students learn better from one intervention vs. another. Consequently, we need to employ the decision tree to determine difference of outcome between options for each student, based on the corresponding features, and then output the best option for this student.

One possible approach to solve the problem is to build a model to estimate the effect of each possible option, and then select the one with the best estimated outcome for this student [3, 7]. However, it has three disadvantages.

1. The mechanism would be very complicated if the size of possible options is large, like in course recommendation systems [8, 9], each course could be considered as an option, and there might be tens of different options.
2. It would be hard to extract rules from the decision trees if they are constructed on different set of features or nodes are split with different values.
3. The constructed models might be meaningless, and learn about what is termed "unacceptable evidence" [4] of a meaningful interaction. Like in Figure 1, the optimal option for students, no matter with high or low knowledge level, is always option 1. This conclusion does not need a decision tree. This is because traditional decision tree algorithms use criteria such as information gain, Gini

L@S 2017, April 20-21, 2017, Cambridge, MA, USA
 ISBN 978-1-4503-4450-0/17/04…$15.00
DOI: http://dx.doi.org/10.1145/3051457.3053983

index, and impurity, to construct a tree with the least error in prediction [10-12]. However, in this problem, we should focus on building decision trees to find splits that would alter which option works best. As a result, the decision tree is an "acceptable evidence", i.e., the best option differs for different types of students.

To overcome those disadvantages, we will introduce a new decision tree algorithm in this paper that constructs a decision tree for all possible tutoring options. In this manner, we address our research question: which option is optimal for a particular type of student? Moreover, many studies focus on investigating the effect of **an** intervention. However, this paper explores a set of 22 experiments to find a mechanism for customizing instruction to an individual student.

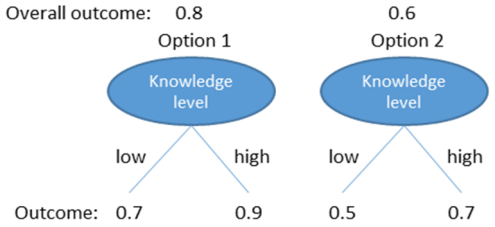

Figure 1. Example of decision trees based on Pashler et al.'s framework of "unacceptable evidence" [4]. These two decision trees are useless, because students would always have a better outcome with option 1 with or without the decision trees.

METHODOLOGY

Split Criterion

We define the cross effect as the improvement of target value from picking the overall best option to picking the best option for each group. If there is no cross effect, then there is no difference with disaggregated by the feature, like the sample decision tree in Figure 1, and thus the improvement is 0. In this work, we focus on dealing with the continuous target variable, so the leaf node in the decision tree constructed by our method represents the mean target value of instances in that node. The cross effect of a feature f in a data space D is computed as:

$$CE = \sum_{g_i \in G} \frac{mean\left(D_{g_i, a=\hat{a}(D_{g_i})}\right) * |D_{g_i}|}{|D|} - mean(D_{a=\hat{a}(D)})$$

$$\hat{a}(D) = argmax_{a_j}\left(mean\left(D_{a=a_j}\right)\right)$$

Where G is the set of all groups disaggregated by the feature f, a is the option variable, $mean(D)$ is the mean target value of all instances in D, and $\hat{a}(D)$ represents the best option in the data space D.

Discretization

In this work, we deploy binary split in decision tree induction. To evaluate a discrete feature f, for each possible value f_i, we consider the data space is divided into two groups, the one with "$f = f_i$" and the other with "$f \neq f_i$", and then we compute cross effect according to this division. Finally, the best one is marked as the cross effect of f.

```
PROCEDURE split(D, R, n, l):
1:   best_feature ← '', cut ← −∞, best_effect ← −∞
2:   FOR each feature f in R:
3:     (f_i, e) ← pickBestCutpoint(D, f, n)
4:     p ← computePValue(D, f, f_i)
5:     IF p ≤ l AND e > best_effect:
6:       best_effect ← e, best_feature ← f, cut ← f_i
7:   IF best_feature ≠ '':
8:     root ← createInternaNode(D,best_feature,cut)
9:     R ← R − f
10:    (D_1, D_2) ← divide(D,best_feature,cut)
11:    root.left = split(D_1, R, n, l)
12:    root.right = split(D_2, R, n, l)
13:  ELSE
14:    root ← createLeafNode(D)
15:  RETURN root
```

Figure 2. The split process in decision tree induction. It takes current data set, a set of features, and other two parameters as inputs, and it outputs the root node of the tree constructed based on the input data set.

To discretize a continuous feature f, as its values are denoted in order as $\{f_1, f_2, \ldots, f_m\}$, each value will be considered as a cut point, so that a value, f_i, will divide the data into two groups, one containing the instances with $f \leq f_i$, and the other with $f > f_i$. The cross effect according to this division will be computed. Therefore, we need to examine $m - 1$ possible values.

After computing the cross effect for all possible values of a continuous feature, we are not going to pick the value with the best cross effect as the cut point for this feature. To reduce the effect of noise, we smooth the result by replacing the cross effect of each value with the average of its 5 neighbors, including itself. Finally, we pick the value with the best smoothed effect as the cut point for a continuous feature. We also use factorial ANOVA on the resulted disaggregation to compute p-value for both continuous feature and discrete feature. This p-value will be used in decision tree induction.

Decision Tree Induction

Our method is a top-down induction method, starting with the whole data set, it keeps splitting the data set into two sub data sets, until all possible splits meet one or both of following stop criteria:

Stop criterion 1: in one of the two sub data sets, there is an option, such that the number of instances with the option is less than n, a parameter of minimum size used in the induction.

Stop criterion 2: in a split, the corresponding p-value $>$ l, where l is another parameter in the induction, which indicates the significant level of splitting.

The split procedure is shown in Figure 2, line 1 is initialization; line 2-6 picks the feature alone with its corresponding cut point that produces the best significant split. A significant split means the corresponding p-value is less than or equal to a pre-defined significant level l; line 7-12 constructs an internal node with the feature and cut point,

and then recursively runs the split procedure on the two split sub data sets; if stop criterion 2 has been reached, then the tree stop growing, so line 14 creates a leaf node. Finally, this procedure outputs either an internal node or leaf node.

EXPERIMENTS AND RESULTS

Experiments Setup

We use 5-fold cross validation to evaluate our method on two types of data sets, one is the simulated data set that is generated with pre-defined distributions, and the other is collected from real experiments. The process of evaluating trained model in this work is different from in the traditional classification problems, since we going to evaluate how well students would have done by given the customized options, not how well the model predicts. After a model is trained by the training set, each separated group, according to the model, is marked with an option which brings the best mean target values. To evaluate the trained model in the testing set, first, the testing set is also assigned into corresponding groups based on the model structure. And then for each group, the mean target value of the instances in the testing group with the marked option is the estimated value. Finally, how well a model is making customization is computed by taking the average target value of all groups.

Simulated Data

Data Generation

This simulated data set contains 6 features, 2 options, and 1 continuous target variable. The 6 features are generated with uniform distribution $U(0,1)$, and the target variable with normal distributions, of which parameters are defined in the Table 1. We generated 200 instances for each group, 100 for each option. According to these distributions, the optimal option for group 1 and group 4 is option 0, while option 1 for group 2 and group 3. With picking the optimal option for each group, the upper bound of this data set that the best method can achieve is 0.708.

Results

We use the method described in previous section to construct decision trees on this simulated data with different significant levels, 0.05, 0.1, 0.2, 0.3, 0.5, and 0.9, the other parameter, minimum size, is set to be 20 in all trees. We also compared the constructed decision trees with two methods, random selection and always pick the option that has the best overall mean target value in the training set. As shown in Figure 3, the results of decision trees are very closed to the upper bound, and they are much better than random selection and method of picking the best. Moreover, the decision tree with parameter 0.05 is better than the other models, and it is statistical significantly ($\alpha < 0.001$) better than the method of picking the overall best, the reason could be that it is less overfitting to the training set.

Another issue we want to focus on is the structure of the decision trees. The constructed trees have the same structure on the top 2 levels: the feature f1 is used in the root node, and f2 is used in both of nodes in the 2nd level. The mean of all cut points of f1 is 0.491, the mean of cut points of f2 in the left node is 0.665, and 0.685 in the right node. The trees have the structure that is similar with the pre-defined one, except that they have more levels since we impute the data with some noise – the other four features.

			option 0		option 1	
	f1	f2	μ	σ	μ	σ
group 1	≤0.5	≤0.8	0.5	0.2	0.4	0.2
group 2		>0.8	0.3	0.2	0.8	0.2
group 3	>0.5	≤0.8	0.4	0.2	0.6	0.2
group 4		>0.8	0.9	0.2	0.2	0.2

Table 1. Parameters of the distributions used to generate simulated data set.

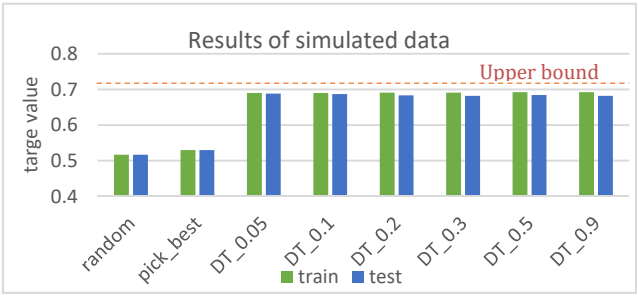

Figure 3. Results of running our decision tree algorithm with different significant levels on the simulated data, compared with random selection method and the method of picking the overall best option.

Real Data

Data Description

This data set is collected from 22 ASSISTments experiments. In each experiment, a student is randomly assign into one of two groups, and the two groups are associated with different tutoring conditions, such as video feedback and text feedback. And each experiment has a unique pair of conditions. The experiment with minimum size has 121 students, 1640 of maximum, and 10690 students in total.

In each experiment, students are learning a specific skill by practicing related problems, and they are required to obtain 3 correct in a row to complete the experiment. To investigate which condition is optimal for each student, we defined $target = (3/\#practices)^{0.7}$, if complete; otherwise, 0. And we use 6 student features in decision tree induction: prior completion rate, prior percent of correctness, prior average mastery speed, guessed gender, learning rate in previous 3 days, and percent of correctness in previous 3 days.

Results

In this experiment, we run our decision tree algorithm on the 6 features with different significant levels, 0.05, 0.1, 0.2, 0.3, 0.5, and 0.9. We also use only one feature, prior mastery speed, to build decision trees with significant level 0.9 on this data. The parameter of minimum size is also set to 20 in the decision tree induction. These methods are also compared with random selection and picking the overall best option.

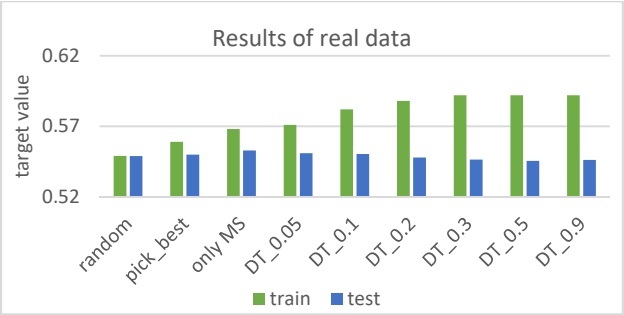

Figure 4. Results of running our decision tree algorithm on 6 features with different significant levels on real data, compared with the decision tree constructed with only one feature, prior master speed, and another two methods, random selection and picking the overall best option.

The average of the 22 results is shown in Figure 4. In the training set, the decision tree with larger significant level performs better, but this also could result in more likely to be overfitting. Such as in the testing set, the decision tree with parameter of significant level 0.05 is better than the other decision trees that are constructed on the same set of features. This might be one reason that no remarkable improvement is obtained from decision trees. Another reason might be no treatment effect exists in some experiments, which results in the "faked" patterns of cross effect captured in the decision trees. Therefore, we examined the test results in each individual experiment, and found that in some experiments, the decision tree algorithm had poor performance, while in other experiments, it performed reliably better than the other methods.

More interesting, the decision tree based on only one feature, prior master speed, is even better than the ones built with more features, but it is not statistical-significantly ($\alpha = 0.9$) better than picking the overall best. Finally, we can conclude that even though some decision trees might be overfitting, if we use appropriate features and parameters, we could get a better result, at least as well as, than just picking the overall best option for all students.

CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an algorithm that constructs a decision tree to predict the optimal tutoring option for each student. We demonstrated our algorithm could capture the pre-defined customization structure in a simulated data set. We also illustrated how to use the constructed decision tree to make customization in real experimental data sets, and compute the how many learning gains we could obtain with the customized options. Although there have been many A/B to evaluate the effect of individual interventions in the learning process, little effort has been done on searching for educationally useful aptitude-treatment interactions. This work represents a step in that direction.

It is interesting to use our method to explore other tutoring systems, such as Reading Tutor [2]. The challenge is to obtain a good data set that are large with a lot of interventions. Are we able to find interesting patterns, and what are the potential gains for customization?

REFERENCES

1. Benjamin S Bloom, *The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring.* Educational researcher, 1984. **13**(6): p. 4-16.
2. Jack Mostow and Joseph Beck, *When the rubber meets the road: Lessons from the in-school adventures of an automated Reading Tutor that listens.* Scale-Up in Education, 2006. **2**: p. 183-200.
3. Hyun Jin Cha, Yong Se Kim, Seon Hee Park, Tae Bok Yoon, Young Mo Jung, and Jee-Hyong Lee. *Learning styles diagnosis based on user interface behaviors for the customization of learning interfaces in an intelligent tutoring system*. in *International Conference on Intelligent Tutoring Systems*. 2006. Springer.
4. Harold Pashler, Mark McDaniel, Doug Rohrer, and Robert Bjork, *Learning styles concepts and evidence.* Psychological science in the public interest, 2008. **9**(3): p. 105-119.
5. Anthony Botelho, Hao Wan, and Neil Heffernan. *The prediction of student first response using prerequisite skills*. in *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. 2015. ACM.
6. Hao Wan and Joseph Barbosa Beck, *Considering the Influence of Prerequisite Performance on Wheel Spinning.* International Educational Data Mining Society, 2015.
7. Jong Woo Kim, Byung Hun Lee, Michael J Shaw, Hsin-Lu Chang, and Matthew Nelson, *Application of decision-tree induction techniques to personalized advertisements on internet storefronts.* International Journal of Electronic Commerce, 2001. **5**(3): p. 45-62.
8. Gerhard Weber, Hans-Christian Kuhl, and Stephan Weibelzahl. *Developing adaptive internet based courses with the authoring system NetCoach*. in *Workshop on Adaptive Hypermedia*. 2001. Springer.
9. Joseph Jay Williams, Na Li, Juho Kim, Jacob Whitehill, Samuel Maldonado, Mykola Pechenizkiy, Larry Chu, and Neil Heffernan, *The MOOClet framework: Improving online education through experimentation and personalization of modules.* Available at SSRN 2523265, 2014.
10. Hendrik Blockeel and Luc De Raedt, *Top-down induction of first-order logical decision trees.* Artificial intelligence, 1998. **101**(1): p. 285-297.
11. Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen, *Classification and regression trees*. 1984: CRC press.
12. J Ross Quinlan, *C4. 5: programs for machine learning*. 2014: Elsevier.