Student Coding Styles as Predictors of Help-Seeking Behavior

Engin Bumbacher, Alfredo Sandes, Amit Deutsch, and Paulo Blikstein

Stanford University, Stanford, California

{buben,alfredos,adeut195,paulob}@stanford.edu

Abstract. Recent research in CS education has leveraged machine learning techniques to capture students' progressions through assignments in programming courses based on their code submissions [1, 2]. With this in mind, we present a methodology for creating a set of descriptors of the students' progression based on their coding styles as captured by different non-semantic and semantic features of their code submissions. Preliminary findings show that these descriptors extracted from a single assignment can be used to predict whether or not a student got help throughout the entire quarter. Based on these findings, we plan on developing a model of the impact of teacher intervention on a student's pathway through homework assignments.

Keywords: Computer Science Education, Machine Learning

1 Introduction

Recent work in CS education has leveraged machine learning techniques to gain insight into the ways in which students approach a given programming assignment. Piech et al. [2] created a graphical model of how students in an introductory programming course progressed through a homework assignment. They were able to extract characteristic pathways, which can be used to predict their midterm grades.

Our own research examines the relationship between students' coding styles and their general help-seeking behaviors; we want to know when students learning to program get help, why they get help, and how the help impacts their progression. We hope that this work could be used to determine potential points on a student's learning path where help interventions would be most effective; this could transform into a technology feature for recommendation of "help" in tutor learning systems.

In this preliminary study, we used machine learning techniques to show that the evolution of a student's code in a single assignment could be predictive of whether or not that student sought help throughout the academic quarter. This suggests that student coding patterns might be indicative of relevant behavioral or cognitive processes of students learning to program that give rise to certain help-seeking behaviors.

2 Data Sources

We collected data from a Stanford introductory course on programming methodologies in Java. Every time a student tried to compile their program we collected text snapshots of their code, regardless of whether or not their code compiled. We had access to a subject pool of 370 students. The target assignment we analyzed contained 8,772 snapshots of code across all students. To measure help-seeking behavior, we collected tracking data from an on-campus homework help service, where teaching assistants (TAs) track student visits. Thus, help-seeking behavior here refers to whether or not a student got help. Over the span of the quarter, there were 1,148 visits in the help center from 172 distinct students. Of these students, 91 sought help 1 or 2 times, and 81 sought help three times or more.

For this study, we analyzed a single assignment in which students were tasked with writing a program that accepts an arbitrary list of numbers and outputs the maximum and minimum values.

3 Methods

Our basic methodology, from data preprocessing to classification, can be broken down into three stages: characterizing code snapshots, characterizing students based on the ensemble of their snapshots, and classification of TA help data.

3.1 Characterizing code snapshots

We created a set of both semantic and non-semantic features with which we tried to capture what we refer to as "coding styles". The non-semantic features are: number of lines of a code, number of comments, and number of comment blocks. The semantic features are: number of variable declarations, number of method declaration, and the number and nesting level of loops and conditional statements within the code. Through a preliminary examination of student code submissions, we found that these features best describe the constrained solution space of the target assignment.

As a metric for dissimilarity measures, we used a simple Euclidian distance. For the clustering step, the data was normalized by the mode of each feature.

3.2 Characterizing students: Cluster-based student feature selection

We clustered a student's snapshots based on structure similarities representative of different possible program structures. This allowed us to characterize the progression of a student through the assignment as a progression through clusters. In the unsupervised learning step, these clusters were generated using kernelized k-means with Gaussian kernels [3]. The number of optimal clusters was determined by a combination of silhouette value maximization [4] and Davies-Bouldin index minimization [5].

Assigning each snapshot to the corresponding cluster, we defined the students with a new feature set consisting of: the number of different clusters visited, the total number of cluster changes, a measure of the variance of the number of successive snapshots within the same cluster, the time to solution, and the total count of clusters visited.

3.3 Classification of the TA intervention data

In order to classify the TA intervention data, we trained a nonlinear Support Vector Machine (SVM) with a Gaussian radial basis function kernel with the student feature data by means of 10-fold cross-validation. Given the highly non-linear feature space, kernelized SVM was best suited for the binary classification task [6]. We also ran a Naïve Bayes Classifier with less promising results (data available upon request).

4 Results

As shown in Table 1, the kernelized SVM trained on the student population features predicts whether a student got help or not performs with an accuracy of 66.5% with a precision of 63.6% and a recall of about 71%.

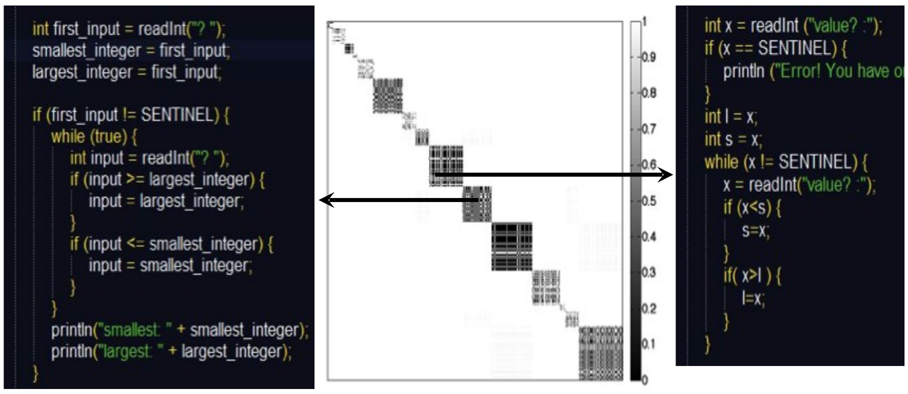
Table 1. Performance of Binary SVM Classifier

Accuracy	66.5%
Precision	63.6%
Recall	71.1%

Figure 1 shows the dissimilarity matrix after clustering the student snapshots into 16 clusters and arranging them according to the clusters. Each matrix entry m_{ij} represents the dissimilarity in terms of Euclidian distance between snapshot i and snapshot j, with black being a dissimilarity of zero. As can be seen, the snapshots are well separated into the clusters (which is further supported by the silhouette value of about 0.72 in Table 2). The selection model based on the Davies-Bouldin Index and the silhouette value suggests 16 clusters as a good representation (see Table 2).

Table 2. Characteristics of the k-means clusters of code snapshots

Optimal choice of clusters:	16
Silhouette index:	0.72
DB-index:	0.43



```
int first_input = readInt("? ");
smallest_integer = first_input;
largest_integer = first_input;

if (first_input != SENTINEL) {
    while (true) {
        int input = readInt("? ");
        if (input >= largest_integer) {
            input = largest_integer;
        }
        if (input <= smallest_integer) {
            input = smallest_integer;
        }
    }
    println("smallest: " + smallest_integer);
    println("largest: " + largest_integer);
}
```

```
int x = readInt ("value? :");
if (x == SENTINEL) {
    println ("Error! You have o
}
int l = x;
int s = x;
while (x != SENTINEL) {
    x = readInt("value? :");
    if (x<s) {
        s=x;
    }
    if( x>l ) {
        l=x;
    }
}
```

Fig. 1. Dissimilarity matrix of the k-means clusters, and 2 snapshots representative of their clusters

To illustrate how codes within different clusters can differ from each other, we have added two code snapshots representative of their clusters in Figure 1. As can be seen, the code snapshot on the right of the dissimilarity matrix has two if statements nested within a loop; the code on the left has two if statements nested within a loop, which is in turn nested in another if statement.

5 Conclusions

Using a simple measure of a student's progress and representation of their code in a single assignment, we were able to predict with accuracy of about 66.5% the student's help-seeking behavior across the whole quarter. In light of the fact that the representation is very simplistic, and that we have excluded any complex measures entailing temporal dimensions, these results indicate that there is structure in the relationship between a student's progression through an assignment and their help-seeking behavior, and this relationship requires further exploration. Nonetheless, these results are especially interesting because they suggest that there are generalizable characteristics found in a small sample of code from one assignment early in the class that can be indicative for help seeking behavior across the entire quarter.

This project is the start of an extended investigation of student programming data. Based on the preliminary findings, we intend to integrate the TA help data and weekly survey data about motivation and perceived difficulty into a Markov model of assignment progress that can predict student grades and suggest critical points for intervention.

6 References

1. Blikstein, P.: Using Learning Analytics to Assess Students' Behavior in Open-Ended Programming Tasks. In: Proceedings of the Learning Analytics and Knowledge conference (LAK11), Alberta, Canada (2011).
2. Piech, C., Sahami, M., Koller, D., Cooper, S., & Blikstein, P.: Modeling how Students Learn to Program. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, pp. 153–160, Raleigh, NC (2012)
3. Sewell, G. & Rousseau, P. J.: Finding Groups in Data: An Introduction to Cluster Analysis, Wiley & Sons (2005)
4. Wang, K., Wang, B., & Peng, L.: CVAP: Validation for Cluster Analyses. Data Science Journal, vol.8, pp.88-93 (2009)
5. Petrovic, S.: A Comparison Between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS clusters. In: Proceedings of the 11th Nordic Workshop of Secure IT Systems, pp.53–64, Linkoping, Sweden (2006)
6. Stanevski, N., & Tsvetkov, D.: Using Support Vector Machine as Binary Classifier. In: Proceedings of the International Conference on Computer Systems and Technologies, Varna, Bulgaria (2005).