A Complex Adaptive System Approach to Predictive Data Insertion for Missing Student Data

John Stamper[1], Theodore Carmichael[2]

University of North Carolina at Charlotte, University of North Carolina at Charlotte

Key words: *Adaptive Learning, Educational Data Mining, Predictive Data Insertion, Complex Adaptive Systems*

Abstract:

Educational Data Mining techniques generally rely on complete sets of student data for analysis. In order to limit the number of students that have to be removed due to missing data, various methods have often been used to fill in these missing data points. We propose a novel method utilizing a Complex Adaptive System to predicatively insert these missing data. This system compares very favorably to other commonly used algorithms, and it is generally more flexible and scalable in approach. What's more, the CAS can output results in real-time, and can quickly adapt to newly added data. We present our results of comparison between a CAS and other common methods of Predictive Data Insertion on a set of real student data, and discuss some of the considerations when implementing a CAS.

1 Introduction

Adding intelligent tutoring capabilities to an e-learning system will allow the system to adapt to an individual student's needs. These adaptive tutoring capabilities have been shown to be effective in increasing student outcomes [3,5], but time consuming to implement due to the complex cognitive modeling required to produce such systems. This research is part of an effort to ease the time to create these cognitive models by using Educational Data Mining (EDM) techniques to automatically generate them[7,8]. Some of the most effective EDM algorithms require complete data sets to generate results [1]. In a true e-learning course that runs over an entire semester it is likely that many students will be missing some data such as a missed homework or quiz. While it is common practice in EDM to just remove the students with missing data from the analysis, this presents a problem when analyzing an entire semester where 20% or more of the students may have missed something. Other techniques such as Zero-Fill or simple mean have been used, but they present problems with skewed results. More advanced techniques such as K-Means clustering are more effective, but scale poorly with large datasets that change often since they must be run every time data changes. We propose a novel technique utilizing a Complex Adaptive System (CAS) to predict the missing student data and compare the results to several other methods including the traditional K-means data mining technique.

2 Background and Related Work

Within the EDM community, the simple removal of rows with missing data has been common for algorithms that require a complete data set. This practice becomes problematic when larger amounts of data are missing, since the students missing data may represent a unique group that should be recognized. Predicting missing data is a useful exercise in many data-

mining situations, and there have been many techniques applied to the problem. In this study we look at several universal methods and compare them to our CAS approach. The data for this study, which is fully described in the experiment section, consists of binary data that corresponds to correct (1) and incorrect (0) answers.

2.1 Zero-Fill

Aside from simply removing the entire row of data with missing values, the Zero-Fill method is the simplest method of dealing with missing data. This method is often used when an educator gives a student a zero for missed work. Obviously, this method automatically assumes the student has missed the question which may not be the case if the student actually had the opportunity to answer the question. Of all the methods studied, the Zero-Fill has the greatest potential to negatively skew the results.

2.2 Simple Mean

The Simple Mean method uses the average response from all students for a particular question and all questions across a particular person to predict the value of a missing person/question pair.. In the case of binary data a value less that 0.5 results in a zero, and equal to or above 0.5 returns a value of 1.

2.3 K-Means Clustering

A K-Means clustering algorithm is a partitional clustering technique that can be used to predict values by creating a number of clusters, and using the closest cluster for prediction [9]. In order to perform the K-Means clustering, the number of clusters must be predetermined at the start of the run. The steps to the algorithm are as follows:

1. *Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *Assign each object to the group that has the closest centroid.*
3. *When all objects have been assigned, recalculate the positions of the K centroids.*
4. *Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.*

Although the algorithm works surprisingly well, there are some known issues. First, the location of the starting points has a great effect on the solution. Usually, points are randomly placed which can sometimes lead to items too close together. To mitigate this issue, implementations of the K-Means clustering will run the algorithm multiple times using different starting points. Second the distance measure may not be relevant. Euclidian distance or Manhattan distance are the mostly widely used distance functions, but care must be taken when calculating these values over multiple attributes that may or may not be scaled in the same way. Finally, there is the issue with selecting K, the number of groups to cluster. There are no known ways to find the optimal number of clusters. Generally, researchers will start with a lower K, then work their way up to more clusters until some threshold of improvement gain is reached. This method is time consuming and can also lead to overfitting of the data.

2.4 Complex Adaptive Systems

Complex Adaptive Systems are a specialized form of a Multi-Agent System (MAS). In a CAS, the agents are generally less complex, more homogenous, and greater in number than a traditional MAS[10]. Each agent in a CAS has only limited, local knowledge of the 'landscape' of the large dataset. The agents communicate with other agents at random to acquire this local knowledge. The emergent property of the system – that is, the predictive power of the system as a whole – is derived from aggregating each agent's local knowledge, and the agents themselves adapt during the training period based on their contribution to both positive and negative predictions[4,6].

In terms of data-mining and Predictive Data Insertion, a CAS derives its usefulness from the fact that after the training period is complete, new data can be added easily to the dataset. This flexibility ensures that the data insertion process on the new data can occur in real-time as the data is added, rather than necessitating performing an analysis on the entire data set from scratch.

What's more, a CAS system provides an extremely useful framework for analyzing large datasets as the shear volume of data increases. If, instead of a few hundred entries along each axis of the matrix, a database contained in the range of thousands or millions of columns and rows, then traditional methods of correlating the data can quickly become overwhelmed. In a CAS, however, each agent continually polls other agents at random, and preserves a list of those that would be the most useful in predicting missing data points. The list can be used to make predictions in real time even as the agent continues to poll other members of the CAS. In this way, users of the data set can not only get a "best guess" at virtually anytime, but can also judge the relative usefulness of allowing the agents to continue their searches, in terms of the computing power needed vs. the increase in accuracy.

3 Experiment

3.1 Data

The data used for the study was drawn from a random set of available student data consisting of 219 students answering 250 questions. For the purpose of training, complete data was used to produce a student data matrix. Then 10% of the known data points were removed from the matrix and predictions were made. The removal of random data points was done five separate times, so that five individual trials could be performed for each data-insertion method. The predictions for each trial were compared to the actual values and the percentage of correct predictions was calculated. The mean and standard deviation of the results of all the trials for each method are included in the graph of results.

3.2 Setup

The Zero-Fill and Simple Mean methods were evaluated using simple java programs. The K-Means clustering was performed using the open source Cluster 3.0 software, which is an enhanced version of the original Cluster software developed at Stanford University [2]. The CAS software was modeled with Net Logo software and then implemented in java. The same data was run through each of the methods and the results were compared.

In terms of a CAS, our system used the students as "agents;" each agent polled a certain number of other agents at random and used a simple calculation to determine how close each

was to the other, in terms of how well their individual answers matched across all the questions. Each agent kept and continually updated a list of five "friends" it would use to later make the predictions for its own missing data points. The maximum number of agents polled was varied between 20 and 500, and the accuracy was recorded for each level.

It's important to note that the choice of students-as-agent – although intuitive – is not necessarily the only was to implement a CAS algorithm. The questions themselves could also be used as agents, and they could individually determine other questions that track well and would be used as predictors of missing data. Alternatively, both concepts of agents could be used in tangent, whereby each student-as-agent keeps a separate list of "friends" for each of a few collections of correlated questions, as determined by the questions-as-agents.

The choices of CAS implementation should be carefully considered. The inherent flexibility of the system is generally advantageous in many regards; but if used improperly, it can lead to overfitting of the data, in which case the predictive power of the system would be seriously degraded.

4 Results and Discussion

We compared our CAS's predictive data insertion technique to both the K-Means clustering algorithm and simple, randomly generated data insertions. The K-Means algorithm is a commonly used technique for this type of process, and is therefore a good model of comparison when considering the benefit of our method. The randomly generated data gives a baseline of comparison, and a simple formula was used to measure the error rate across all the inserted (predicted) data and their true values. A percentage of correctly inserted data was then used to compare each of the four methods considered in this work.

During the training period, the CAS was run so that each agent polled a number of other agents at random, and kept a list of the five who were closest in agreement across all the questions. For each of the five sets of data with 10% of the datapoints randomly removed, separate trials were performed where the number of other agents polled was varied. Experiments were run where the agents polled 40, 80, 120, 160, 200, 300, 400, and 500 other agents. Of course, the agents chosen for comparison are done so at random; therefore, redundant instances could occur, particularly where the number of agents polled at random exceeds the number of students in the dataset. The random nature of choosing other agents was preserved on purpose, in keeping with the spirit of non-complex agents interacting within the system. Depending on the particular application of a CAS, it may be more beneficial to ignore agents who have already been asked. The cost of preserving a list of contacts should be weighed against the cost of performing the comparison again.

The results for each trial are shown in the following table. The CAS system results are for 200 agents polled at random; other levels of polling will be compared and discussed in the next section.

Method	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Mean	Std Deviation
Zero-Fill	0.7737	0.7734	0.7715	0.7781	0.7728	0.7739	0.0025
Simple Mean	0.2019	0.2043	0.2054	0.2014	0.2070	0.2040	0.0024
Nearest Neighbor	0.1542	0.1691	0.1623	0.1723	0.1639	0.1644	0.0069
K-Means	0.1529	0.1552	0.1568	0.1498	0.1576	0.1545	0.0032
CAS	0.1466	0.1576	0.1525	0.1498	0.1552	0.1523	0.0043

The CAS system performed slightly better than the K-Means algorithm at the 200 level. (It also performed slightly better at the 160 level, and slightly worse at the level of 120 agents polled at random, as seen in the next chart.) The CAS, Nearest Neighbor, and the K-Means algorithms were all more accurate for inserting missing data than the Simple Mean and the Zero-Fill methods.

As expected, the Zero-Fill method performed abysmally. Given that students in this dataset tend to get a majority of questions correct, a One-Fill method would perform much better, at about a 23% error rate. However, the Zero-Fill method was included to highlight the inherent inaccuracies of not using any of the available information to fill in missing datapoints. Such a method – while preserving the ability to analyze partially complete entries in a database – may, in fact, artificially skew the results of that analysis in unpredictable ways.

4.2 Improvements Over Time

The CAS method performed well – better than the Simple Means – even at the level of only 40 agents polled. This is an indication of the usefulness of the algorithm even at a small level of interaction among agents. Below is the table showing the increase in accuracy over time.

Polling Level	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Mean	Std Deviation
40	0.1779	0.1847	0.1868	0.1806	0.1829	0.1826	0.0035
80	0.1651	0.1693	0.1576	0.1638	0.1689	0.1649	0.0047
120	0.1607	0.1605	0.1563	0.1585	0.1604	0.1593	0.0019
160	0.1479	0.1547	0.1540	0.1516	0.1553	0.1527	0.0030
200	0.1477	0.1530	0.1516	0.1487	0.1516	0.1505	0.0022
300	0.1463	0.1512	0.1472	0.1491	0.1513	0.1490	0.0023
400	0.1425	0.1530	0.1459	0.1500	0.1488	0.1480	0.0040
500	0.1434	0.1496	0.1452	0.1478	0.1507	0.1473	0.0030

As expected, the CAS system improved its results by continually polling more and more agents, which increased the likelihood that the preserved list of "friends" was closer in agreement to the polling agent. However, the *amount* of improvement decreased substantially, so that there was little gain, for example, in doubling the number of agents polled from 200 to 400.

Since a CAS can produce results at any point in the agents' interaction, the amount of polling can be easily tailored to fit the needs of a particular dataset. Improvements can be tracked as the algorithm continues and the agents update and adapt to new information. What's more, the amount of improvement can be easily assessed, so that decisions about how long to allow the algorithm to run can be made with that information in mind.

In the case of extremely large datasets, improvements may very well be ongoing as the algorithm runs for days, or even weeks. However, information can be extracted for preliminary analysis at any time while the polling process continues. Also, new data that is added can be easily assessed using the current state of the agents, even as they improve based on this new information.

4.3 CAS as Nearest Neighbor

The CAS used for this set of student data is essentially a "find good candidates for nearest neighbors" algorithm, but without the need to exhaustively compare every student to every other student. Of course, a traditional "nearest-neighbor" algorithm – such as the one used in this study – only uses one "friend" to predict all the missing data for a particular student. By using five friends who are all relatively close in agreement, the CAS system was able to generally perform better than the Nearest Neighbor algorithm. What's more, the results across the five trials varied less than Nearest Neighbor, indicating that a CAS may also be more consistent. This is not surprising, since using five friends rather than one allows the CAS to better manage possible instances of an atypical data-point; there is information available from four other friends to counteract against such an outlier.

Five friends were chosen as a reasonable balance of simplicity, accuracy, and stability. This number, of course, should be chosen beforehand, or without input from the testing data to prevent a tendency to overfit. Also, as previously discussed, a student-as-agent model could be coupled with a question-as-agent model, so that "nearest neighbors" for each student are defined in terms of one sub-set of questions, rather than across all the questions. Again, the number of sub-sets – if used – should be limited; otherwise, overfitting could occur.

In the case of extremely large datasets, the student-as-agent method could also use a statistically similar sub-set to train on, learning who the closest agents are without the need to look at *all* the questions for every interaction. This particular idea is not novel, of course; however, as the CAS framework is ideally suited for a reasonable application of limiting the number of comparisons across people, it is also adaptable for limiting the number of questions considered, which helps address scalability issues of large datasets.

One final note about the missing data used in these experiments: we did consider that real student data may have missing data points for various reasons, and that those reasons may be correlated. In other words, there may be information in the pattern of missing data. In our experiments, of course, the data was removed completely at random, and done so five separate times, in order to add confidence that the results reflected this approach. The reason we chose this method is due to the fact that missing data often is not correlated in a meaningful way, or it is unknown what the pattern, if there is one, may mean. Our CAS approach can be easily adjusted to account for these patterns if desired; therefore, we decided to base this study on a "pure" approach, as if the missing data points were due to simple, random corruption, or due to reasons that are unrelated to the analysis of the data set. When adapting a CAS algorithm for Predictive Data Insertion, it is therefore beneficial to consider why some of the data is missing, and if any useful information may be derived from this.

5 Conclusion and Future Work

Although Complex Adaptive Systems have been used for prediction before, we have not seen any employed in the educational e-learning or EDM communities. The power of this method comes from its ability to constantly run and improve over time. We hope to continue our research by completing additional experiments with our system and apply additional learning criteria to our CAS. The method we provided is extremely robust and adaptable. Although binary data was used for the experiments in this study, multiple choice data would work equally well, and we plan to test this level of abstraction next.

References:

[1] Barnes, T., D. Bitzer & M. Vouk. 2005. The q-matrix method: Mining student response data for knowledge. Proceedings of the AAAI 20th National Conference on Artificial Intelligence Educational Data Mining Workshop (AAAI2005), Pittsburgh, PA, July 9-13, 2005.
[2] Cluster 3.0. Ed. Michiel de Hoon. University of Tokyo, Human Genome Center. 2 March 2007 <http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/>.
[3] Conati, C., A.S. Gertner, & K. VanLehn. 2002. Using Bayesian networks to manage uncertainty in student modelling. User Model. User-Adapt. Interact. 12(4): 371-417.
[4] Crutchfield, J.P. and Mitchell, M., 2005, The Evolution of Emergent Computation, Proceedings of the National Academy of Sciences, USA.
[5] Heffernan, N.T. & K.R. Koedinger. 2002. An intelligent tutoring system incorporating a model of an experienced human tutor. Intelligent Tutoring Systems, p. 596-608.
[6] Kennedy, J., Eberhart, R.C., 2001, Swarm Intelligence. Morgan Kauffman Publishers, San Francisco, CA.
[7] Stamper, J.. Automating the Generation of Production Rules for Intelligent Tutoring Systems. The 9th International Conference on Interactive Computer Aided Learning (ICL2006), IEEE Student Track, October 2006.
[8] Stamper, J. Automating the Generation of Student Models for Intelligent Tutoring Systems. Proceedings of the 13th International Conference on Artificial Intelligence in Education (AIED2007), Marina del Rey, CA, July 9-13, 2007. (To Appear)
[9] Tan, P., Steinbach, M., Kumar, V. 2006. Introduction to Data Mining. Boston, MA: Pearson Education Inc.
[10] Weiss, G. (ed.), 1999, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA. The MIT Press.

Author(s):

John Stamper, Ted Carmichael
University of North Carolina at Charlotte, Department of Computer Science
Charlotte, NC 28223
jcstampe@uncc.edu, tdcarmic@uncc.edu