Using simulation to evaluate data-driven agent-based learning partners

Ilknur Icke[1] and Elizabeth Sklar[1,2]

[1] Dept of Computer Science,
The Graduate Center, City University of New York
365 Fifth Avenue, New York, NY 10016 USA
iicke@gc.cuny.edu
[2] Dept of Computer and Information Science,
Brooklyn College, City University of New York
2900 Bedford Ave, Brooklyn, NY 11210 USA
sklar@sci.brooklyn.cuny.edu

Abstract. We use simulation to evaluate data-driven agents that will act as learning partners in multi-user educational interactivities. Data on human learners has been collected in an adaptive educational assessment environment, and we use these data to train agents to emulate humans. These agents will be deployed in a companion on-line interactive system designed to help students learn new skills and overcome deficiencies highlighted in the assessment portion of the environment. Human users are grouped according to particular features, and agents are trained to embody the group's behavior. The burden of creating a meaningful training set is shared across a number of users instead of relying on a single user to produce enough data to train an agent. This methodology also effectively smooths out spurious behavior patterns found in individual humans and single performances, resulting in an agent that is a reliable representative of the group's collective behavior. Our demonstrated approach takes data from hundreds of students, learns appropriate groupings of these students and produces agents which we evaluate in a simulated environment. We present details and results of these processes.

1 Introduction

Computerized student assessment tools provide educators with a number of different ways to assess student knowledge via standardized tests. While most standardized tests provide results in terms of a single numeric score and its relation to the mean, the on-line, adaptive, multi-dimensional testing tool developed by Children's Progress Inc (CPI)[3] gathers a robust data set for each student, highlighting specific areas of strength and weakness. We are working with CPI to develop individualized tutoring modules and two-player educational games that will be provided for students after they take the assessment, tailored to offer

[3] http://www.childrensprogress.com

targeted instruction and learning experiences, personalized to the needs of individual students, as identified by the test.

To support this work, we employ agent-based simulation techniques for three tasks. First, we construct agents that simulate human learners, by training probabilistic controllers from the assessment data set. The goal is to deploy these agents as artificial learning peers to engage in two-player games with human learners. Second, we evaluate (and validate) these agents by placing them in a simulated assessment environment, measuring their performance in comparison to the corresponding behavior patterns of their human trainers. Third, we use the validated agents to help predict outcomes in newly developed assessments. The work described here presents results from the first two tasks.

Most intelligent tutoring systems that employ agent technology do so to provide a knowledgeable, automated teacher. These types of agents are typically designed by computer scientists, in collaboration with human teachers, pedagogical specialists and developmental psychologists [1]. While others have explored the notion of *simulated students* [2], few have deployed these as learning peers in an interactive system. Some of our earlier work has accomplished this with simple games and data collected during the games [3], but not using *a priori* assessment data to train agents, as is the case here.

We are not the first to suggest constructing agents to emulate humans interacting in an on-line system [4–6]. Previous work has shown that training agents to emulate humans produces better results if the training set is a composite of multiple humans grouped according to application-specific metrics [7]. A large data set is generally desirable when training agents, and grouping human data sets with similar characteristics helps smooth out anomalies. In the study presented here, we analyze log files from student interactions with CPI's standardized testing system, called the Children's Progress Academic Assessment (CPAA), using a data-mining perspective. Related work has been reported previously (e.g., [8–10]), but within different user modeling contexts and not with the purpose of subsequently using the models to control agent-based learning partners.

Here, we outline our processes for creating agents to simulate humans based on standardized test data and for evaluating the agents in a simulated assessment environment. First, we partition the large standardized test data set to serve as the basis for training a suite of distinct, agent-based, probabilistic controllers. This step is crucial to the success of each agent as an emulator of a specific class of human behaviors. If the data used to train the agents is not well defined, then the agents in the suite will not be distinct from each other—their behavior patterns will overlap too much. The longterm goal is to create individual agents whose behavior patterns differ, each designed to target particular curricular deficiencies and help human learners advance in these areas. Second, we compare two different techniques for partitioning the data, using standard clustering techniques from the literature. For each cluster, we produce an agent whose actions should typify behaviors characteristic of members of the cluster. Finally we evaluate the results, by placing all the agents we generated in a simulated assessment environment and measuring which agents most closely

resemble the human counterparts they are meant to be simulating. Our results demonstrate that one clustering technique clearly produces a better approximation of group behavior patterns than the other, as evaluated in the simulated assessment environment.

2 Our approach

The CPAA is organized around an underlying *lattice* structure [11]. An illustration is shown in figure 1. Each node represents a question in the assessment, and links indicate possible paths from one node to another. All students start at the same node, the leftmost one in the figure. Students who answer the question correctly move along the green link (up and to the right). Students who answer the question incorrectly move along the short red link (down and to the right), where they are given a follow-up "hint" question. Lattices are designed by developmental psychologists and educational specialists for a set of *core concepts* that are essential to early learning of fundamental numeracy and literacy skills. Each lattice contains different numbers of nodes and patterns of links. Within each core concept, the questions in the test are organized into four sub-concept levels and each sub-concept has a prime question and a hint. The content and structure of the tests are designed manually, with the educational specialist developer virtually drawing links from one node to the next indicating the order in which questions should be presented to students, based on student performance. We created the plot shown in figure 1 by manually assigning (x, y) coordinates to each node, corresponding to its relative location within the 2-dimensional landscape of the assessment "map". The lattice shown is used to assess "phonemic awareness" (PA) — the ability to recognize sounds made by single and combinations of letters at the beginning and end of words, as well as in the middle.

We were given the log files (records of user interactions) from 117 first grade students (ages 6-7 years old) who took the assessment in Fall 2006. From the student log files, we extracted the sequence of the questions, student answers and elapsed time information into separate files for each portion of the test. We built an application to visualize the paths each student took during the assessment using the JUNG Graph Visualization tool [12]. Figure 2 illustrates the paths taken by two different students in the PA portion of the assessment, drawn using our visualization tool. In these drawings, the green nodes represent correct answers, red nodes incorrect answers and the numbers on the edges represent the time it took the student to answer that particular question. Student A (top half of the figure) did well, answering most questions correctly. Student B (bottom) did poorly at first, rallied, slumped, rallied, and slumped again to the end of the test. The differences in performance of different students is clearly visible.

2.1 Partitioning training data

Most assessments compare students by examining the final score achieved; but we are interested in grouping students by examining the similarities in their *tra-*

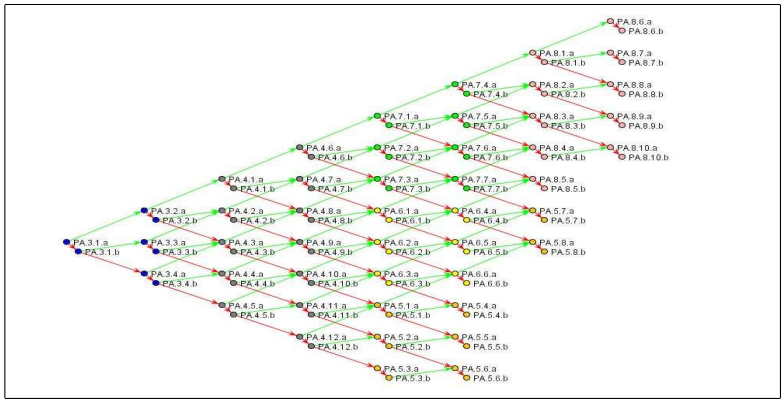
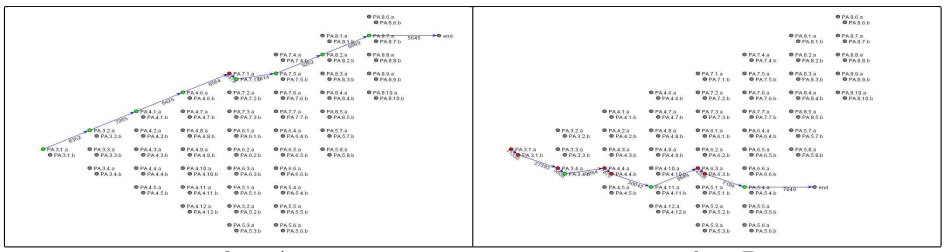


Fig. 1. Example lattice ("PA", for phonemic awareness). The node labels are not important here; what is important is the structure of the graph.



student A student B

Fig. 2. PA paths for two students ("A" and "B"). The node labels are not important here; what is important is the shape of the trajectory of each student.

jectories through the lattice landscape. It is important to note that the students do not make directed choices about which paths to take, but rather the system chooses each next node in reaction to the student's performance so far. Currently, CPI has over 50,000 students taking the assessment three times per year. This large data set gives us the opportunity to create agents that simulate a range of human behaviors. As mentioned earlier, we wish to create a suite of agents that each mimics certain categories of human behavior. We partition the complete data set into clusters, grouping humans with similar behavioral characteristics, as exhibited by following similar trajectories through the CPAA.

Data clustering is a well-studied field in the literature, and the particular algorithm chosen for a clustering task varies depending on the characteristics of the data set and the goals of the task. Our aim is to produce coherent grouping that will train agents that are distinct from each other. We investigated a variety of techniques and here we compare two of these methodologies: Euclidean

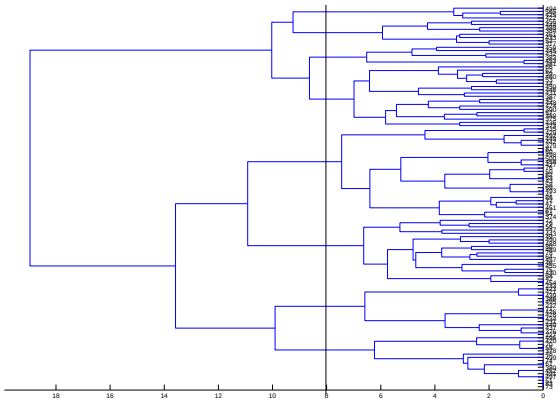


Fig. 3. Hierarchical clustering of PA student response vectors, using Euclidean distance with 012 coding. Vertical axis indicates ID numbers of individual students. Horizontal axis contains h values.

distance based on feature vectors and Hausdorff distance based on geometric similarity.

Euclidean distance. We generated "feature vectors" to encode the student responses during the test. For example, for PA, we generated 94-dimensional feature vectors for each path, each dimension representing a student's response to one question. Note that sequences of questions are chosen for students dynamically, based on the their performance and the connections in the lattice; so students do not see all the questions. We experimented with different codings for the student responses including: 0=incorrect, 2=correct, 1=not seen (021 coding), −1=incorrect, 1=correct, 0=not seen (-110 coding), 0 =incorrect, 1=correct, 2=not seen (012 coding), and also a 3 variable coding: seen ($\{1|0\}$), incorrect($\{1|0\}$), correct ($\{1|0\}$). We used a hierarchical clustering algorithm in Matlab [13] using the Euclidean distance between the feature vectors as the distance measure. As can be seen in figure 3, the algorithm found 8 main clusters at level $h = 8$ using the 012 coding. The 012 coding gave the best results compared to the other Euclidean codings.

Examing the clusters, we realized that the groups were not really homogenous, either with respect to the assessment scores or the paths taken during the assessment. This could be because Euclidean Distance was not an appropriate measure for this type of data, since it does not contain any information on the relationship between the questions. As will be shown later, this intuition is correct, and the next methodology provides better results.

Hausdorff distance. Since our aim is to group students according to the similarity of trajectories followed on the assessment maps, we decided to employ clustering techniques on sequential data. Similar work has been reported in [14] on classifying Linux users with respect to their experience level based on

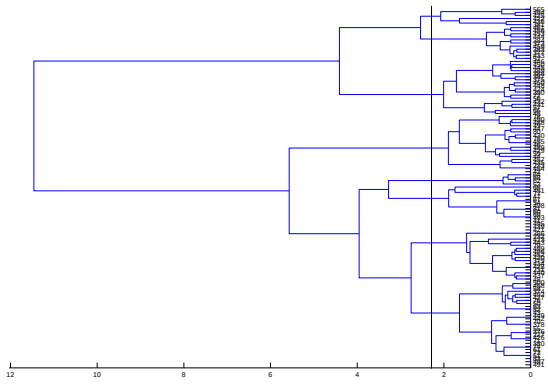


Fig. 4. Hierarchical clustering of PA trajectories, using Hausdorff distance. Vertical axis indicates ID numbers of individual students. Horizontal axis contains h values.

command logs and in [15] on clustering financial time series data. We assigned coordinates to each node with respect to their closeness on the assessment maps and used Hausdorff Distance to compute the dissimilarity between any two paths. After computing the pairwise distances between each path, we applied a hierarchical clustering algorithm in Matlab. As can be seen in figure 4, the algorithm found 8 main clusters at level $h = 2$. Also, note that within each cluster there are smaller groupings of paths that are closer to each other than the others in the same cluster.

We evaluate the clustering techniques in two ways: first, by measuring the coherence of the clusters generated by each. We use metrics described in [16] which are used to compare clusters of trajectories. Two metrics are employed. The first is "shape complexity", or σ, which is computed as:

$$\sigma = disp/length$$

where *disp* is the displacement or the distance between the first and last points in a trajectory and *length* is the number of points in the trajectory. The second is "divergence", or covariance of the first, middle and last points in the trajectory. Figure 5 compares these values for the two clustering techniques: Euclidean distance and Hausdorff distance. The size of each cluster (number of students belonging) is shown as well as the average number of points in the trajectories of all student members. The columns to focus on are the two rightmost, which contain the standard deviation of σ and *cov* for the trajectories that comprise each cluster. The absolute numbers are not important here; what is important is the relationship between the numbers within each column. Smaller numbers indicate tighter coherence amongst cluster members—this is our aim. The average σ for the Hausdorff is 3.95, whereas for Euclidean, the average $\sigma = 4.77$. The

Hausdorff clustering

cluster	size	points	σ	*cov*
1	11	121	2.64025	5687.83
2	6	64	2.42758	3871.61
3	19	203	2.22529	6487.44
4	28	298	4.43650	6841.44
5	4	37	4.89523	6909.63
6	12	108	4.55438	5109.01
7	18	197	4.91468	8704.28
8	19	206	5.40067	8579.10

Euclidean clustering, 012 coding

cluster	size	points	σ	*cov*
1	6	68	4.87123	8353.60
2	19	204	8.83936	15166.20
3	4	41	2.20674	4062.49
4	8	89	2.14947	7424.48
5	16	175	3.29415	4157.30
6	15	159	1.60082	4469.70
7	21	223	7.63581	12024.50
8	28	275	7.57357	7320.25

Fig. 5. Cluster similarity measures, showing for each cluster: the number of trajectories in the cluster, the total number of points covered by all the trajectories in the cluster, the standard deviation for σ and the standard devation for *cov* (see text for further explanation).

average covariance for Hausdorff is 6523.79, whereas for Euclidean, the average $cov = 7872.31$. Using these metrics, the Hausdorff distance clustering technique results in better coherence. We note that a cluster-by-cluster comparison reveals that the Hausdorff coherence is better for the clusters on either end of the spectrum, while the Euclidean is better for those in the middle. This is an interesting result which bears further investigation.

The second method we used for evaluating clustering techniques is described in the next section, where we used the clusters to train agents and our evaluation is based on the correlation between the behavior of the trained agents and the groups of humans the agents are emulating, as well as the separation between agents (i.e., distinctiveness of behavior patterns).

2.2 Training agents

The next step in our procedure is to create agents whose behavior typifies that of each cluster. We did this for both sets of Hausdorff distance and Euclidean distance clusters. First, we generated a profile for each cluster as follows. For each node in the lattice, we tally the number of students in the cluster who visited that node. We compute statistics based on students' responses to each node as the basis, aggregated for all members of a cluster into an agent training set. For each cluster, we generate one representative agent.

Each node represents a question in the assessment, and each question is designed to elicit information about students' skills and *deficiencies*. Each question has four multiple-choice answers, only one of which is right. Associated with each incorrect answer are one or more *error codes*. There is a master list of error codes and a mapping between that and each node in the lattice. As an example, take the generic lattice illustrated in figure 6a. All students start at the node labeled $\mathsf{q0} \in Q$ in the figure. There are one or more deficiencies, each of which we will refer to as $\mathsf{d}_j \in D$, that each node (question q_i) is assessing. Essentially a table

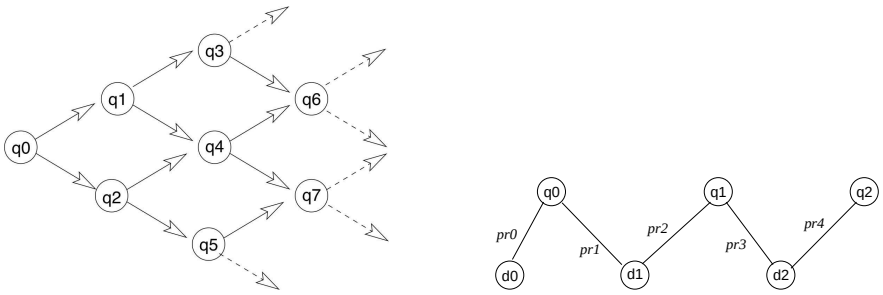
with $|Q|$ rows by $|D|$ columns is engineered when the lattice is designed, by education specialists who assign a Boolean value to each cell in the table indicating which deficiencies are intended to be revealed by each question. We use this table to pose two types of questions:

1. *a modeling question*—what is the probability that a student possesses deficiency d_j, given that they answered question q_i incorrectly, i.e., what is $Pr(d_j|q_i)$?
2. *a prediction question*—what is the probability that a student will answer question q_i incorrectly, given that they possess deficiency d_j, i.e., what is $Pr(q_i|d_j)$?

In actuality, there is not a one-to-one correspondence between deficiencies and questions. In other words, for each deficiency column in the table, multiple question rows will contain data; and for each question row, multiple deficiency columns will contain data. This multiplicity is one of the great strengths of CPI's assessment system. It is also a feature that we capitalize on here. Thus we rephrase our two questions:

1. *modeling*—what is the probability that a student possesses the deficiencies in set D, given that they answered the questions in set Q incorrectly?
2. *prediction*—what is the probability that a student will answer the questions in set Q incorrectly, given that they possess the deficiencies in set D?

The influence diagram [17, 18] shown in figure 6b provides a graphical illustration of this situation. There are two types of variables represented: deficiencies $\{d_0, d_1, d_2\}$ and questions $\{q_0, q_1\}$. Question q_0 is designed to assess whether a student possesses deficiencies in set $\mathrm{D}' = \{d_0, d_1\}$; question q_1 is designed to assess whether a student possesses deficiencies in set $\mathrm{D}'' = \{d_1, d_2\}$.



(a) generic lattice (b) example influence diagram

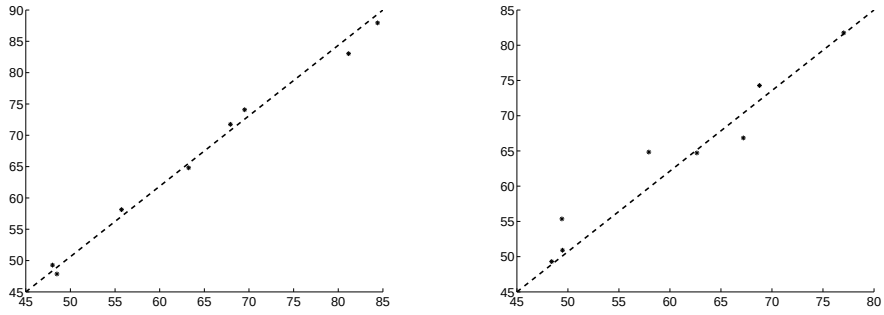
Fig. 6. Agent training structures.

We use the student logs and cluster assignments, in collaboration with CPI's error code definition table, to fill in probability tables, one per cluster. We tally

the number of students within the cluster who visited each node and the percentage of them who answered the question incorrectly, indicating particular deficiencies. Thus, for each cluster, we have a table that indicates how likely it is that a member of that cluster possesses each deficiency. This probability table becomes the heart of the control function for each "cluster agent." Eventually, these tables will be used to guide the behavior of an agent acting as a peer learner in the educational games component of our project. For now, we use them to simulate each agent stepping through the lattice.

2.3 Evaluating agents

Finally, we evaluate the efficacy of our methodology by simulating an assessment using each agent generated. We performed 999 evaluation trials for each agent—since the agents are controlled probabilistically, they will not behave exactly the same way in each simulation run. In order to evaluate these agents, we wish to determine how well each agent fits the cluster profile from which it was modeled. Based on our experiments during clustering, we had determined that taking into account the geometry of the assessment paths rather than a vector of student responses resulted in more meaningful clusterings. We decided to utilize the same notions to evaluate the agents as well. We used the shape complexity (σ) metric described earlier to compare the relationship between trajectories generated by agents with those generated by humans. Figure 7 plots the average σ for each cluster based on the trajectories exhibited by humans (x-axis) against the corresponding value for trajectories generated by the agent representing each cluster in the 999 evaluation runs. The correlation with the Hausdorff technique is quite high.

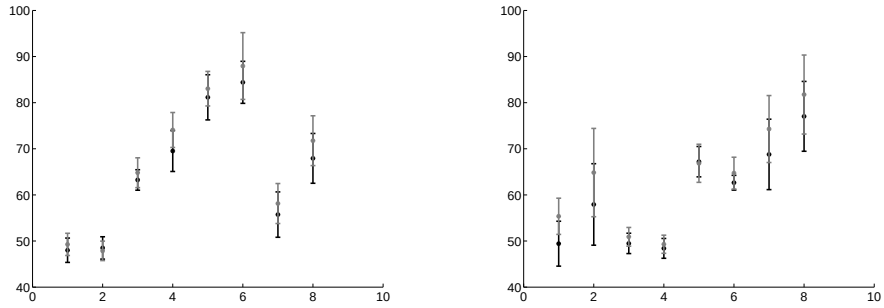


(a) Hausdorff (b) Euclidean (012 coding)

Fig. 7. Correlation between Agents and Humans.

We also want to compare the separation between agents, recalling our goal to produce a suite of agents, each of which represents different behavior patterns. Figure 8 shows the mean and standard deviation of σ for each of the clusters

computed for each clustering technique. The black bars represent the clusters based on human trajectories; the grey bars represent the trajectories generated by the agents in the 999 evaluation runs. Once again, the Hausdorff produces superior results because greater distinction between each cluster can be seen in the lefthand plot.



(a) Hausdorff (b) Euclidean (012 coding)

Fig. 8. Separation between Agents.

Finally, a sampling of trajectories for each cluster (using the Hausdorff technique) are shown in figure 9. For each cluster, the first (leftmost) plot shows the trajectories (blue lines) over 999 test runs of the agent. The remaining plots show a representative sample of human student trajectories (red lines) for each cluster. The point is that the blue lines should represent a composite set of red lines within the same cluster.

3 Conclusion

We have described a methodology for generating agent-based simulations of human learners using probabilistic student models based on data collected during students' interactions with a specialized on-line, adaptive, multi-dimensional assessment environment. We used the data to create clusters of students with similar behaviors and then trained agents whose actions typify cluster members. We explored two methods of clustering, one based on a feature-vector comprised of right/wrong answer choices made by each student and employing a Euclidean distance metric to determine groupings. The second is a graphical approach, based on examining the paths students take through the underlying lattice structure of the assessment and employing a Hausdorff distance metric to determine groupings. From these, we generated a profile for each cluster of students based on inferred deficiencies of students on that skill. We used these deficiency profiles to train agents to emulate cluster members, and finally, we evaluated the efficicacy of these methods by comparing the trajectories produced by agents

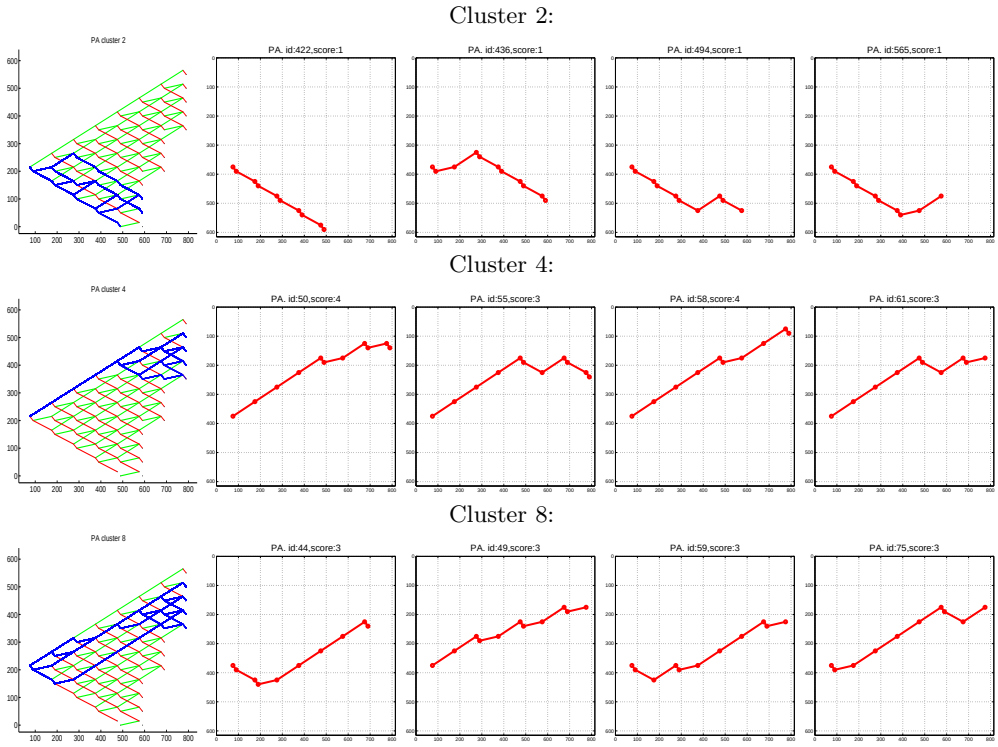


Fig. 9. Comparing agent and human trajectories.

acting in a simulated assessment to those of cluster members produced in the real assessment. Our results show that the trajectory analysis and Hausdorff metric is superior to the Euclidean methodology for our purposes.

Selecting the proper number of clusters for the problem at hand is an issue. As shown in figures 3 and 4, the choice of where to draw the vertical line determines how many clusters are selected. In the work presented here, we picked the number of clusters manually, by examining these dendrograms. In the future, we will explore a variety of automated methods for selecting the appropriate number of clusters, with the goal of finding groups that offer the most coherence while at the same time, the most data compression. We will also investigate the possibility of selecting clusters essentially by drawing vertically stepped lines across the dendrograms.

Our approach has proven to be promising, and we are looking into the application of an expectation-maximization (EM) algorithm to estimate the conditionals on the deficiencies in order to make the agents fit better to the underlying student models.

Acknowledgments

This work was partially supported by the National Science Foundation under NSF IIP #06-37713 and by the US Department of Education under #ED-07-R-0006.

References

1. Johnson, W.: Pedagogical agents for virtual learning environments. In: International Conference on Computers in Education. (1995)
2. VanLehn, K., Ohlsson, S., Nason, R.: Applications of simulated students: An exploration. Journal of Artificial Intelligence in Education **5**(2) (1996) 135–175
3. Sklar, E.: CEL: A Framework for Enabling an Internet Learning Community. PhD thesis, Department of Computer Science, Brandeis University (2000)
4. Cypher, A.: Eager: Programming repetitive tasks by example. In: Proceedings of CHI'91. (1991)
5. Maes, P.: Agents that reduce work and information overload. Communications of the ACM **37**(7) (1994) 31–40,146
6. Balabanović, M.: Learning to Surf: Multiagent Systems for Adaptive Web Page Recomendation. PhD thesis, Stanford University (1998)
7. Sklar, E., Blair, A.D., Pollack, J.B.: Chapter 8: Training Intelligent Agents Using Human Data Collected on the Internet. In: Agent Engineering. World Scientific, Singapore (2001) 201–226
8. Hofmann, K.: Subsymbolic user modeling in adaptive hypermedia. In: The 12th International Conference on Artificial Intelligence in Education, Young Researcher Track Proceedings. (2005) 63–68
9. Mavrikis, M.: Logging, replaying and analysing students' interactions in a web-based ILE to improve student modeling. In: The 12th International Conference on Artificial Intelligence in Education, Young Researcher Track. (2005) 101–106
10. Merceron, A., Yacef, K.: Educational data mining: a case study. In: The 12th International Conference on Artificial Intelligence in Education. (2005)
11. Galanter, E., Galanter, M.: Adaptive evaluation method and adaptive evaluation apparatus. United States Patent, no. 6,511,326 B1 (2003)
12. : Jung universal network/graph framework. `http://jung.sourceforge.net`
13. `http://www.mathworks.com/products/matlab/`
14. Jacobs, N., Blockeel, H.: User modeling with sequential data. In: Proceedings of the 10th International Conference on HCI. (2003) 557–561
15. Basalto, N., Bellotti, R., De Carlo, F., Facchi, P., Pascazio, S.: Hausdorff clustering of financial time series. Physica A **379** (2007) 635–644
16. Zhang, Z., Huang, K., Tan, T.: Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In: ICPR (3). (2006) 1135–1138
17. Pearl, J.: Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann, San Mateo, CA (1988)
18. Russell, S., Norvig, P.: Artificial intelligence: A modern approach. 2nd edn. Prentice Hall (2002)