Applying Machine Learning the Assessment of Problem-Solving Skills

Max Arseneault (marsenea)
Department of Computer Science
Stanford University

Abstract

This project explores how machine-learning algorithms could be applied to automate assessment of college students' scientific-problem-solving skills, using log data generated in the PhET "Black Box Problem" simulation. Specifically, we investigated the performance of different machine-learning algorithms trained on summary vectors of students' interactions to predict their problem-solving performance. The Artificial Neural Network (ANN) model achieved relatively high performance as measured by traditional binary classification metrics such as F! score.

1 Introduction

Our project explores how machine-learning algorithms could be applied to assess students' problem-solving skills through the large volume of log data generated in an interactive, science simulation,specifically the PhET "Black Box Problem" simulation (7). Problem-solving skills refer to students'ability to solve scientifically-oriented questions through data collection, interpret data, as well as formulate solutions. In the past, the assessment of such skills has been limited, as students' answers to multiple-choice assessments capture little information about the problem-solving process that they went through. The interactive simulation environment affords and necessitates key problem-solving practices (e.g. collecting real-time data, testing a proposed solution) that would be infeasible to carryout with paper-and-pencil tests. The simulation thus yields rich and detailed data on the problem-solving process that individual participants have gone through and the problem-solving practices they have adopted. Luckily, the proliferation of machine-learning algorithms affords education researchers the opportunity to measure students' problem-solving skills in an automated fashion and at scale.

One promising recent approach is Deep Knowledge Tracing (DKT), a recurrent neural network (RNN) model that makes predictions on learning outcomes based on a sequence of interactions that students undertake (5). The model intends to address the problem of knowledge tracing in computer-supported education, in which "a machine models the knowledge of a student as they interact with coursework" (5). Knowledge tracing, or machine modeling of student knowledge during coursework interaction, can be used for better automated and more detailed assessment, intelligent curriculum design, discovery of structure in student tasks, personalized allocation of resources based on students' individual needs, and the extension of learning gains provided by one-on-one tutoring to anyone in the world, thus advancing global education equity (5). Last quarter in the Wieman Group,we adopted the approach of Deep Knowledge Tracing by using an RNN model on user-backend data from the "Black Box Problem." This is a simulation embedded in the PhET Interactive Simulations, a STEM-education project founded by Carl Wieman. With over 100 million uses worldwide this past year alone, augmentations to the PhET framework would be globally impactful (7). However, our approach was plagued by overfitting; the large number of parameters required for an RNN vastly exceeded our relatively small dataset.

CS229: Machine Learning, Spring 2019, Stanford University, CA.

We decided to rewind and divert our efforts to more simple, vanilla neural network architectures in tandem with more complex input features. In addition to much more intelligently engineered features, we have grounded the problem in a theoretical framework which attempts to quantify accumulated knowledge in a grocery-list fashion and applied various unsupervised learning algorithms to our engineered feature data.

2 Related work

The task of modeling and evaluating students' knowledge and problem-solving skills has received extensive interest from the fields of educational data mining and learning analytics (1; 4); however, most of this interest has been directed towards Massive Open Online Courses (MOOCs) rather than interactive-learning environments (2). One of the sparse, researched examples of a interactive-learning environment is IPRO, a programming game for iOS devices where students program soccer-playing "bots" to compete in a multiplayer, online game (2). This research was quite analogous to our own, but focused on programming knowledge and mining of structure from student tasks rather than physics knowledge and automated assessment. As for research related to Deep Knowledge Tracing, the seminal paper, authored by Piech et al., introduces the technique and argues for its predominance over other relevant dynamic probability models at the task of knowledge tracing. We were particularly influenced by the paper's method of encoding student interactions as input to a neural network. Deep Knowledge Tracing has a robust presence in the educational-analytics literature; its shortcomings have been addressed with augmentations such as the regularization techniques outlined in Yeung et al (8).

3 Dataset and Features

For this project, we used the event-log data generated during 178 researcher-administered, student attempts at solving the PhET "Black Box Problem" simulation, specifically black box problems 8, 9, and 14 (7) (Figure 1). Problem solvers investigated the electrical configuration behind the black box by connecting electrical components and measurement tools to the black box and observing the outcomes. Students were given 15 minutes to solve the problem at varying difficulty, and their attempted solution was given an objectively-calculated assessment score (0-6).

Since this user-backend data, which consists of console logs from 178 sessions of students attempting to solve the black-box problem, is particularly tortuous, data processing was our main encumbrance in both quarter of work. Each session was composed of over 10,000 JSON lines corresponding to console events ranging from clicking-and-dragging to changing a light bulb's resistance. The composition of each JSON line is shown in Figure 2. Due to poor documentation of console event nomenclature, our first step was to author a comprehensive event taxonomy, detailing each node in the JSON-line tree. This taxonomy served as a dictionary mapping applicable student behaviors to console outputs. Subsequently, as a proof-of-concept and a means of familiarizing ourselves with the data, we coded an event description "chatbot," code which narrated students' actions and the current circuit state given this data as input. Once we had written enough functions enabling us to process and manipulate our data, we diverted our attention to feature extraction. We began with macroscopic, cumulative statistics enumerating the number of times each circuit component was clicked on, added, and deleted, the number of circuit connections made, and the number of cuts made to the circuit. Finally, we moved on to extract more microscopic features which ideally unveil the user's decision process. To evolve beyond our more primitive features, we used NetworkX, a Python package for analysis of complex networks, to maintain an isomorphic model of the on-screen circuit as the student tinkered (Figure 3).

We worked towards our goal of more intelligently-engineered features by decomposing the "Black Box Problem" into six subproblems, one for each combination of the four measurement ports (Figure 1). Since a user can only perform meaningful measurements when they have constructed one loop across two of these ports, we constructed action sequence vectors that compiled user actions during periods in which exactly one loop was active. Possible user actions include creating a loop with a lightbulb, battery, resistor, or wire and/or measuring with a voltmeter or ammeter. We constructed a matrix denoted as the "Knowledge Base (KB)," which quantified with 1's and 0's what the user *must* know about the underlying circuit architecture between the two ports in question. For each of the six possible loops, we awarded one point for knowledge of whether or not the equivalent circuit between

the two ports was empty or contained a wire, resistor, or battery, and knowledge of the value of the component, if applicable. For instance, if the user constructed a loop with a lightbulb across two ports whose equivalent connecting circuit was a single battery, the user would be able to reason that there must be a battery underneath, but unable to ascertain the value without a voltmeter measurement. We constructed an extensive taxonomy of all possible user actions and the knowledge that logically entails from the action, presuming the user is a perfect logician. We were able to produce stack plots such as those in Figure 4 and Figure 5 visually displaying a participant's accumulation of knowledge through the session with respect to each of the six subproblems. We also produced the plot in Figure 6 showing the correlation between the problem solver's final "knowledge score" (0-29) (the Frobenius norm of the KB matrix) and the solution assessment score (0-6). To account for underestimates, we changed the underlying theoretical assumptions to incorporate Bayesian reasoning, such as the probability that a battery is between two given ports given that the user observed the lightbulb illuminate. However, this lead to worse performance and a negative correlation between the "knowledge score" and the solution assessment score. We also compared this "knowledge score" to a prior-calculated "problem-solving score," which is a conglomerate metric noting the number of loops made, circuit decomposition, and other problem-solving strategies. The correlation between the "knowledge score" and this alternative score is shown in Figure 7. Inspired by the potential of this entirely human-engineered technique, we decided to allow a neural network to *learn* the knowledge gains for each of these actions rather than rely on theoretical models of the problem structure.

4 Methods

Grafting off our previous code, we built our feature vectors by considering whether problem-solvers built various loop structures. Specifically, for each of the six black box subproblems, we considered seven different features: whether there was a loop built with a battery and lightbulb, battery and resistor, battery, lightbulb, resistor, only wire, and if there was a valid voltmeter measurement. This amounted to $6 \cdot 7 = 42$ total features per session, each of which corresponding to whether or not the action was performed at any point (0/1). With 178 of these 42-dimensional vectors as our input, we attempted to learn and predict the solution-assessment score (0-6).

4.1 Supervised Learning

Due to poor initial performance on our seven-category (0-6) classification problem, we collapsed the categories into low-performing and high-performing (0-1), turning our tasking into a binary classification problem. We converted all scores below 2 into 0 and all scores above 2 into 1, where 2, the floor of the median, was selected to ensure a balanced class split.

Through trial and error, we found that more simplistic architectures with a single hidden were most effective for our dataset. Despite experimenting with batch normalization, the Adam optimization algorithm, multi-layered architectures, and various activation functions, we ultimately settled on a design with a single, 1024-unit, hidden layer and 42 features for our input layer (Figure 8). We used Xavier initialization, the binary cross-entropy loss function, gradient descent with Nesterov momentum, the ReLU activation function, a weight decay of $.0075$, and dropout regularization with a rate of 0.7 (3; 6; 9). With an 80-20 train-test set split, we trained for 600 epochs. The epoch number was chosen via the heuristic of early stopping. Although we used fixed randomization seeds, the results had a large variance depending on initial conditions. To alleviate this, our metric calculations were an average of 10 randomized train-test set splits.

4.2 Unsupervised Learning

Initial clustering attempts with scikit-learn showed strong clustering between sessions of a particular black box problem number, rather than between the sessions of a particular user, suggesting characteristic problem-solving practices for each of the three black box problems. Convinced of structure in the feature vectors alone, we attempted to condense the 42-dimensional vectors using dimension reduction techniques. We used the HyperTools Python toolbox extensively in our explorations with dimensionality reduction and clustering algorithms. Experimentation with 16 different reduction models, including PCA, led us to select Uniform Manifold Approximation and Projection (UMAP) as our preferred model. For our clustering algorithm, we applied K-Means with $k = 2$.

5 Results/Discussion

The results of our learning algorithms are summarized in Table 1. In addition to overall accuracy, we consider the precision and sensitivity metrics along with their harmonic mean, the F1 score.

The gap between the training accuracy and test accuracy suggests that our neural network is overfitting the data. We attempted a variety of regularization techniques and reduced the number of parameters in our trial architectures, but were unable to further close the gap with our limited data size. All metrics are far above 0.5, corresponding to random guessing. Performance in all metrics also vastly exceeded that of our baseline, although its dataset processing was different enough to discourage direct comparison. Attempts to balance the two classes appear successful, with high precision and sensitivity values, and we champion a high F1 score. In order to be used in automated assessment, additional work should be done to raise both accuracies to at least 95%.

As for our unsupervised learning algorithms, graphical representations of the data can be seen in Figure 9 and Figure 10. Similarly, our metrics were also far above the 0.5 baseline. Our unsupervised learning algorithms were slightly less successful than our supervised learning algorithms, but nevertheless high-performing. Further mining of latent structure should hopefully illuminate student problem-solving practices.

Learning Algorithm	*Neural Network*	*K-Means (k = 2)*
Training Accuracy	0.888	0.751
Test Accuracy	0.757	N/A
Precision	0.778	0.758
Sensitivity	0.875	0.781
F1 Score	0.824	0.769

Table 1: Results and assessment statistics for each learning approach.

6 Conclusion/Future Work

Machine learning algorithms hold promise to model students' problem-solving skills based on large-scale log data. However, the performance of our algorithms was constrained by the small, dataset size. The size of our dataset ideally could be expanded by providing the entire graph to the neural network; however, this would require a dynamic number of features and learned understanding of graph structure. Hamilton et al. notes that traditional machine approaches often rely on summary graph statistics, kernel functions, or carefully engineered features to measure local neighborhood structures in order to extract structural information from graphs, remarking on the lack of a clear solution to this problem. Thus one further approach we might attempt is to reduce our internal, graph representation of the circuit to an equivalent circuit by contracting redundant wires and trimming invalid appendages of the students' circuits. This may allow us to not only simplify our representation of the program state, but also will allow us to better codify the student's knowledge state. Additionally, we would like to map students' various knowledge states to concrete data in our analyses. Ideally one could make our models' "learning" more comprehensible to humans, ideally revealing structure in the student's learning progression.

Additional explorations will include running unsupervised clustering algorithms on sequence information, rather than summary information, to uncover patterns reflective of effective problem-solving practices. We will perform necessary data pre-processing to more easily categorize individual program states. We will run classical clustering algorithms, such as K-means, on program states and subsequently construct a state map, with edges denoting transition probabilities, representing these mined, program-state clusters (2). Such a state map can be taken as a Markov Decision Process (MDP) and used with reinforcement learning.

7 Acknowledgements

The project received advice and guidance from Professor Carl Wieman at the Graduate School of Education and Department of Physics, and Karen Wang, a research student in the Wieman Group.

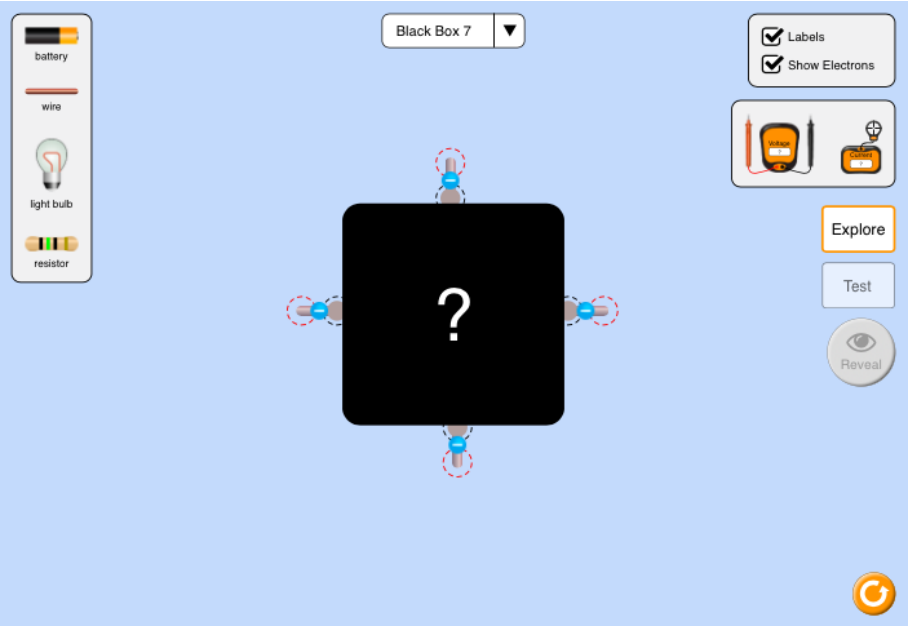

Figure 1: The Black Box Problem Interface

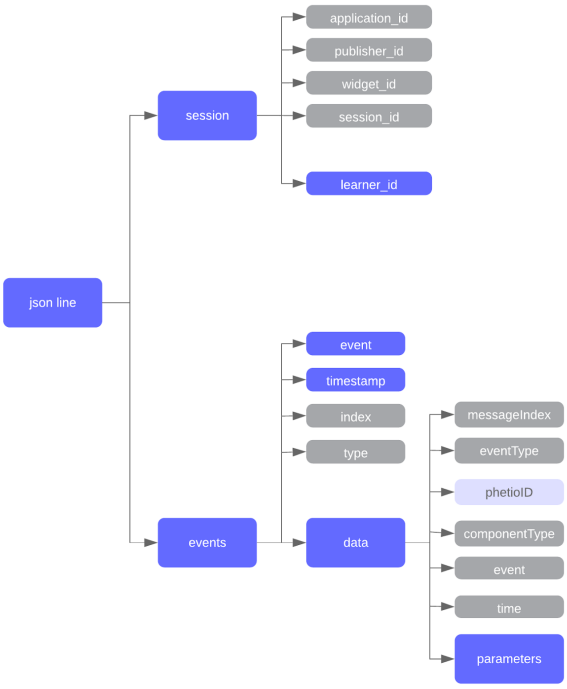

Figure 2: JSON-line hierarchical structure with highlighted portions relevant for feature extraction.

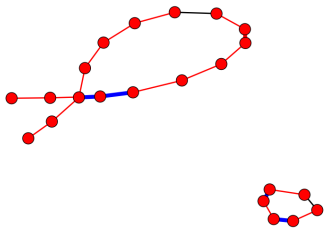

Figure 3: Graphical representation of circuit made with NetworkX package.

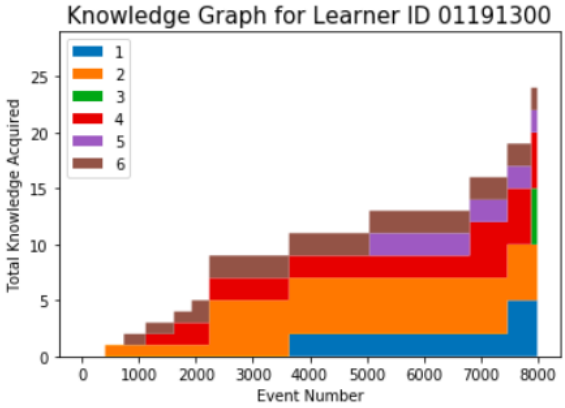

Figure 4: Knowledge gain over time by participant 01191300 in black box 8.

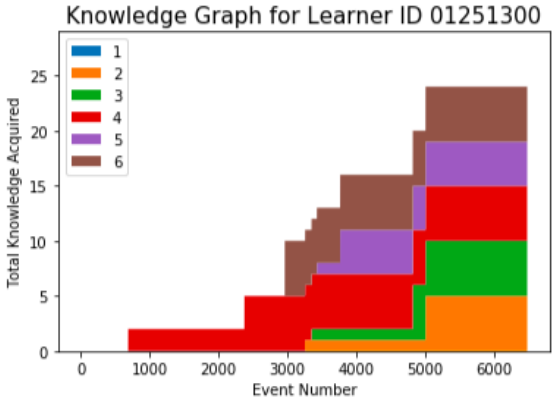

Figure 5: Knowledge gain over time by participant 01251300 in black box 8.

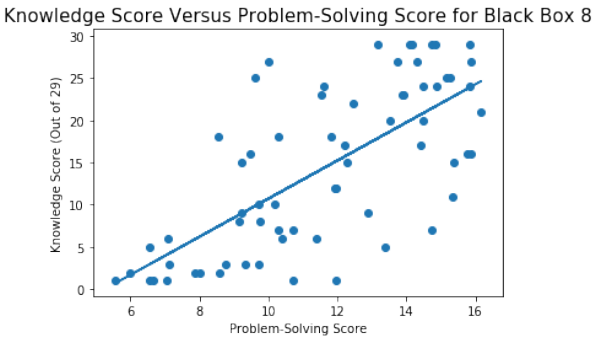

Figure 6: Scatter plot of knowledge scores versus problem-solving scores for black box 8 with linear fit.

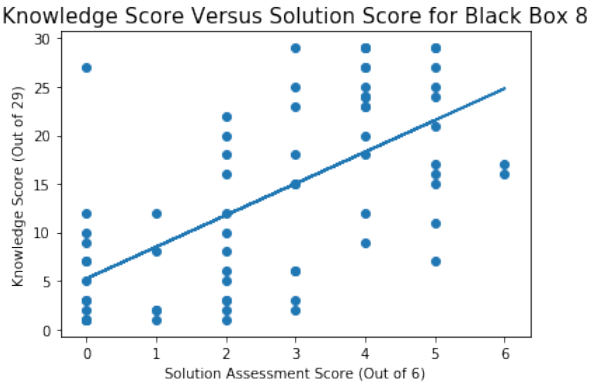

Figure 7: Scatter plot of knowledge scores versus solution scores for black box 8 with linear fit.

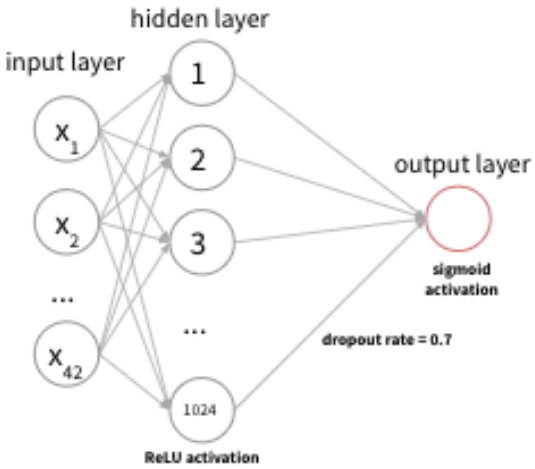

Figure 8: Neural network architecture schematic.

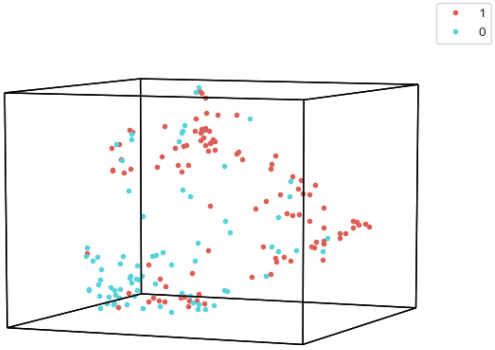

Figure 9: True solution score labels on feature data dimensionally reduced with UMAP.

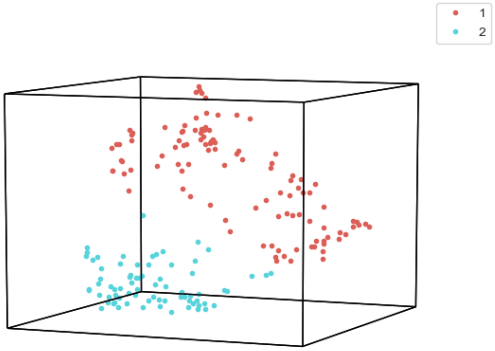

Figure 10: K-Means (k=2) applied to feature data dimensionally reduced with UMAP.

References

[1] Berland, M., Baker, R.S. and Blikstein, P., 2014. Educational Data Mining and Learning Analytics: Applications to Constructionist Research. Technology, Knowledge and Learning, 19(1-2), pp.205-220.

[2] Berland, M., Martin, T., Benton, T., Smith, C.P., Davis, D., 2013. Using Learning Analytics to Understand the Learning Pathways of Novice Programmers. Journal of the Learning Sciences, 22(4), pp.564-599

[3] Bjorck, Nils, et al. "Understanding batch normalization." Advances in Neural Information Processing Systems. 2018.

[4] Gobert, J.D., Sao Pedro, M., Raziuddin, J. and Baker, R.S., 2013. From Log Files to Assessment Metrics: Measuring Students' Science Inquiry Skills Using Educational Data Mining. Journal of the Learning Sciences, 22(4), pp.521-563.

[5] Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications." arXiv preprint arXiv:1709.05584 (2017).

[6] Li, Xiang, et al. "Understanding the disharmony between dropout and batch normalization by variance shift." arXiv preprint arXiv:1801.05134 (2018).

[7] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J. and Sohl- Dickstein, J., 2015. Deep Knowledge Tracing. Advances in Neural Information Processing Systems, pp.505-513

[8] Perkins, K., Adams, W., Dubson, M., Finkelstein, N., Reid, S., Weiman, C. and LeMaster, R., 2006. PhET: Interactive Simulations for Teaching and Learning Physics. The Physics Teacher, 44(1), pp.18-23

[9] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[10] Yeung, Chun-Kit, and Dit-Yan Yeung. "Addressing two problems in deep knowledge tracing via prediction-consistent regularization." arXiv preprint arXiv:1806.02180 (2018).

[11] https://hypertools.readthedocs.io/en/latest/index.html

[12] https://matplotlib.org/

[13] https://networkx.github.io

[14] http://www.numpy.org

[15] https://pytorch.org

[16] https://scikit-learn.org

[17] https://seaborn.pydata.org/