CS230

Applying a Recurrent-Neural-Network Model to the Assessment of Problem-Solving Skills

Max Arseneault (marsenea), Karen Wang (kdwang)
Department of Computer Science
Stanford University

Abstract

This project explores how deep-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 deep-learning algorithms trained on sequences of students' interactions to predict their problem-solving performance. The Recurrent Neural Network (RNN) model achieved relatively high performance as measured by accuracy, with improvements needed on the precision and recall metrics for the positive class.

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 (5). 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 carry out 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 (4). 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"(4). We intend to adopt the approach of Deep Knowledge Tracing on user-backend data from the "Black Box Problem" 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 (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 (4).

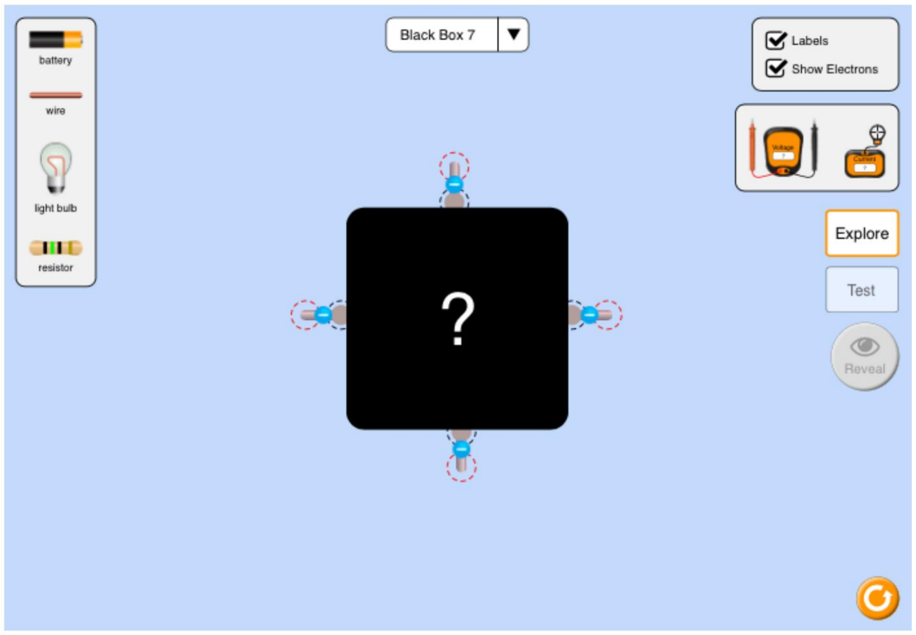

Figure 1: The Black Box Problem Interface

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; 3); 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 recurrent 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 (6).

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 (5) (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. 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

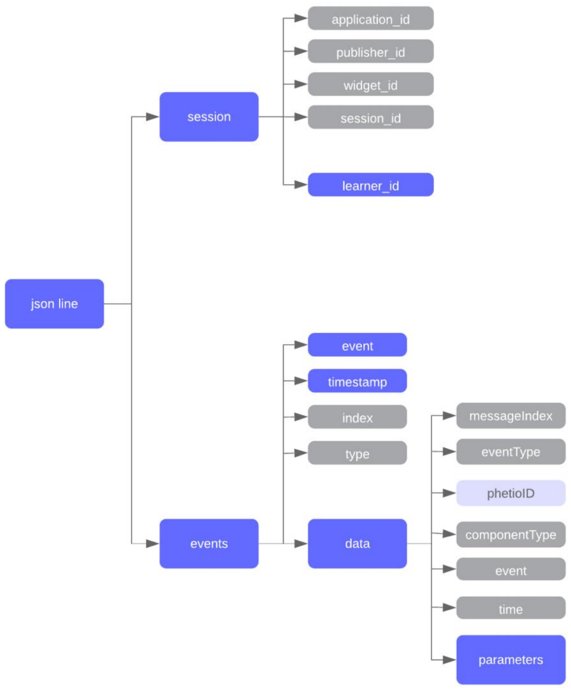

Figure 2: JSON-line hierarchical structure with highlighted portions relevant for feature extraction.

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. Before dealing with sequence-model features, we extracted macroscopic features, bird's-eye-view features from the session as a whole. These features were 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. This processed data was appropriate for use in a vanilla neural network, in which each session is processed individually. Finally, we moved on to extract more microscopic features which ideally unveil the user's decision process. These action-by-action features are suitable for sequence models such as RNNs. To evolve beyond our more primitive, vanilla neural network 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). From this internally-stored representation, we were able to extract features such as the number of edges and nodes, the number of simple cycles, the average node degree, various connectivity metrics, the graph diameter, the clustering coefficient, the number of strongly-connected components, the independent-set size, as well as various connectivity metrics. Because we struggled with too few training examples and exceedingly long sequence lengths, we only ended up incorporating the simple cycles feature into our preliminary sequence vector data. In the end we settled on 12 features for each possible action, represented in a one-hot vector. These actions were adding/deleting a wire, light bulb, resistor, or battery, constructing or deleting a circuit loop, and using the voltmeter or ammeter. With problem solvers' sequence data and their associated solution scores, we train an RNN to predict a particular student's success at solving the black-box problem.

4 Methods

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. For our baseline neural network, which was used with the earlier described macroscopic, session-level features, we settled on an architecture with a single, 32-unit, hidden layer

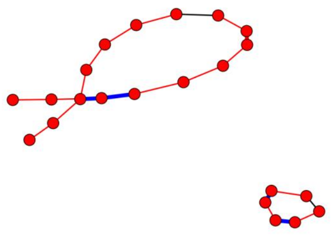

Figure 3: Graphical representation of circuit made with NetworkX package.

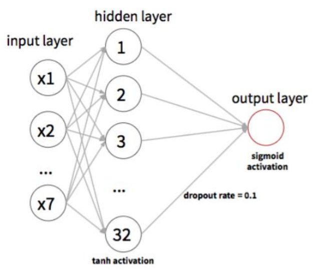

Figure 4: Baseline neural network architecture schematic.

and seven features for our input layer (Figure 4). We used the binary cross-entropy loss function, the Adam optimization algorithm, and dropout regularization with a rate of 0.1. With an 80-20 train-test set split, we trained for 200 epochs.

As for our action-by-action, sequence-model data, we trained first with a unidirectional RNN model with a 16-unit LSTM layer before adding an Attention mechanism. Since the sequence lengths varied from 15 to over 200 actions, we capped the maximum sequence length at 100 and zero-padded all samples when the sequence length was less than 100. The maximum sequence length was chosen after analyzing the distribution of the sequence length and incorporating the information that 80

5 Results/Discussion

The results of our three neural-network-model variations are summarized in Table 1. In addition to overall accuracy, we consider the precision and recall metrics. Precision is the number of samples correctly labeled as positive divided by the total number of samples labeled as positive by the model; recall is the number of samples correctly labeled as positive divided by the total number of positive samples in the ground truth. Since our dataset is class-imbalanced with only 20% positive cases, the precision and recall metrics allow for a better measurement of the predictive performance of the models.

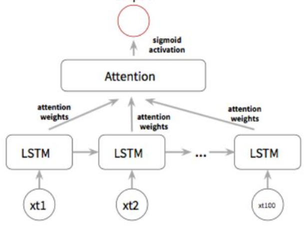

Figure 5: RNN with LSTM cells and Attention architecture schematic.

The large gap between the training accuracy and test accuracy strongly suggests that all three models are overfitting the data. We attempted a variety of regularization techniques and reduced the number of parameters in these architectures, but were unable to further close the gap with our limited data size. Surprisingly, our baseline neural network achieved the highest test accuracy across the three models. This may again be due to our small dataset, but further research should be done to compare the strengths of cumulative, session-level features and of action-by-action, sequence features. Attention appears to have almost always helped our performance according to our chosen assessment metrics; this is likely attributable to the fact that most events in our sequence data, i.e. most student actions, reveal very little about their performance and level of understanding. Our RNN model with LSTM cells and Attention achieved relatively high performance as measured by accuracy, but improvements are needed on the precision and recall metrics for the positive class.

Model	*Baseline Neural Network*	*RNN + LSTM*	*RNN + LSTM + Attention*
Training Accuracy	0.98	0.94	1.0
Test Accuracy	0.78	0.64	0.72
Precision (Class 0/1)	0.83/0.50	0.69/0.25	0.74/0.60
Recall (Class 0/1)	0.89/0.38	0.88/0.09	0.92/0.27
F1 score (Class 0/1)	0.86/0.43	0.77/0.13	0.82/0.37

Table 1: Results and assessment statistics for each model class.

6 Conclusion/Future Work

Deep learning algorithms hold promise to model students' problem-solving skills based on large-scale log data. However, the performance of our algorithms were constrained by the small, dataset size. More work is needed to improve the precision and recall of the algorithms, especially for the prediction of positive (successful) cases. This can be achieved by further data collection, realignment of our task definition, or smarter, human-engineered features. During our research we attempted to incorporate the prior-mentioned graph analytics features, but these led to poorer model performance. Ideally one would provide 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. Human-engineered features of this form that more obviously map onto a student's knowledge state would likely facilitate Deep Knowledge Tracing as well as make the model's "learning" more comprehensible to humans, ideally revealing structure in the student's learning progression.

Additional explorations will include running unsupervised clustering algorithms on this sequence information to uncover patterns reflective of problem-solving performance. We will perform necessary data pre-processing to more easily categorize individual program states. Once pre-processing is complete, we will run classical clustering algorithms, such as X-means and 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 Ana-Maria Istrate, a research associate student in the Wieman Group.

8 Contributions

Data Collection and Raw Data Clean-up: Karen Wang

Data Processing: Max Arseneault

Machine Learning Model Training: Karen Wang and Max Arseneault

Project Write-up: Max Arseneault and Karen Wang

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] 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.

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

[5] 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

[6] 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

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

[8] https://keras.io

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

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

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

[12] https://www.tensorflow.org