Visualizing Online Learner Patterns

Christopher Vu Le CHRISVLE@BERKELEY.EDU
University of California, Berkeley, USA
Electrical Engineering and Computer Science

Zachary A. Pardos ZP@BERKELEY.EDU
University of California, Berkeley, USA
School of Information & Graduate School of Education

Abstract

In this work in progress paper, we apply recurrent neural networks and t-distributed stochastic neighbor embedding (t-sne) towards discovery of latent modes of behavior among a sample of learners using the online learning platform Khan Academy. We explore how the temporal sequence of correctness of problem attempts on various exercises in the platform is abstracted by the hidden layer and how the students' hidden layer state transitions over time. We compare visualizations of several off the shelf variants of t-SNE and provide an animation of the hidden state transition over time.

Keywords

Education, RNN, LSTM, Neural Networks, T-SNE, Visualization

1. INTRODUCTION

The richness and scale of learner event action logs from digital learning environments like Khan Academy[1] provides an opportunity to discover patterns of behavior that can inform instruction and shed light on previously unrecognized cognitive knowledge states and transitions. Our goal in this paper was to train a predictive model of student performance and then visually interrogate the model for its representation of the student. The predictive model utilizes recurrent neural networks (RNN), a special class of artificial neural networks particularly well suited for temporal data. The training data were the sequential problem logs of student, correctness, exercise tuples from Khan Academy users. We visualized the model's hidden state representation at each time slice for each student by reducing its dimensionality down to two using t-Distributed Stochastic Neighbor Embedding (t-SNE), and then creating an animation (with interpolation) of the first 100 time slices for each learner. We provide still and animated visualizations of our pilot model and point to areas for improvement.

2. Methodology

2.1 Dataset

Our original dataset was a sample of learner problem logs from Khan Academy and consisted of 1,044,929 student attempts to answer a problem in the tutoring system, mostly consisting of dichotomously scoreable multiple-choice or short answer questions on arithmetic, geometry, or algebra. The events in the dataset were grouped by learner and ordered chronologically within the group and included the following features per event: anonymized learner id, timestamp, time taken, exercise name, hints used, and Boolean correctness of the answer attempt.

2.1.1 Pre-processing

Since we were interested in looking at our model's hidden state representation of learners over tim,e we choose to filter out users who had logged fewer than 100 problem attempts. For the training of the model we used only the first 100 problem attempts. While many features exist for extended modeling of student behavior, for this pilot analysis we kept only information about the learner ID, chronology, correctness, and exercise name associated with the problem attempt.

The final post-processed training dataset contained a feature vector for each of the 100 time slices (problem attempts) for each of the 1,516 unique students. Each vector had a one-hot encoding of the correctness of the previous problem attempt and a one-hot encoding of the current exercise name, of which there were 750 in our post-processed training set.

2.2 LSTM Model and Experiments

We chose to use the Long Short-Term Memory variant of Recurrent Neural Networks because of its strength in addressing the vanishing gradient. In our data, a learner's performance on an exercise at the beginning of their sequence may have bearing on a particular exercise seen much later in their sequence. LSTMs are superior to RNNs in retaining relevant information from far back in a sequence of input data. The input to the model at each time slice was our feature vector of previous correctness and exercise. The output of the model at each time slice was the correctness of the current attempt and the next exercise. This representation has the model trying to predict both performance but also what exercise the learner will choose to interact with next. Future work will be to adopt the knowledge tracing representation used in [4].

We ran a limited hyper parameter search with 1 or 2 layers LSTM models with a 10 or 100 batch size, and 50, 128 or 256 hidden nodes. All of these experiments were conducted using Keras' implementation of LSTMs and trained to 10 epochs. The best combination in terms of the 10% validation set prediction accuracy was the 1 layer LSTM with batch size of 100 and 128 hidden nodes, in which 54.39% of the predictions accurately predicted both correctness and next exercise.

[1] http://www.khanacademy.org

2.3 T-SNE Dimensionality Reduction

We chose to use the t-SNE framework for our dimensionality reduction because of its recent success in the Merck Visualization Challenge[2] and its common paired with high dimensionality datasets and deep learning. We first extracted the hidden states (100) from our LSTM model for each learner in our post-processed dataset We then used these hidden states to run three different T-SNE models. Each hidden state was treated as an independent instance but were later re-grouped by student when rendering the animation. The three implementations of t-SNE utilized were (1) Barnes-Hut (2) standard python implementation from Laurens van der Maaten, and (3) Barnes-Hut for Scikit-Learn. Barnes-Hut is a variant of t-SNE which is significantly faster [1]. The three implementation completed training in 20h14m, 37m, and 28m respectively.

Each of the three models have different significant hyper parameters and one parameter, perplexity, which they all share in common. Perplexity is related to the number of nearest neighbors used in manifold learning algorithms. Maaten asserts that the most appropriate number depends on the density of the dataset and should be somewhere between 0 and 50. For our current models, we use a value of 30. Larger datasets typically require larger perplexities [2]. In addition to perplexity, the standard python implementation from Maaten utilizes an initial dimensions parameter that reduces the input dataset to the specified number using PCA. The Barnes-Hut implementation from Maaten utilizes a third key hyper parameter, theta, along with perplexity and initial dimensions. Theta is specific to Barnes-Hut implementations and determines how fine or course the approximations is. Values close to 1.0 will be faster, while values close to 0 will be more exact.

2.4 Visualization and Animation

We visualized our two-dimensional output by plotting each of the 100 time steps of the 1,516 students in a scatterplot using matplotlib. We then created an animation[3] by joining each of these time steps along with interpolation frames to allow transitions to be followed. Students that have large differences in their time steps will travel faster while students with small differences will travel slower. In several time steps, some learners maintain their position and do not move at all, perhaps due to repeated responses to the same exercise. We used Holoviews and BokehJS to create an interactive animation inside Jupyter Notebook that allows for scrubbing and hovering over specific points. The code written for these visualizations is available on github[4].

3. Future Work

We have just scratched the surfgace in terms of utilizing visualization to broaden our understanding of human cognition and behavior. There is room for improvement on the modeling front; additional features can be added to the model, such as time taken to respond and educational video viewing behavior, which can potentially improve the discrimination of the model. Additionally, we can utilize a different representation for the targets such that the model predicts performance on all exercises, as was done in [4], instead of the target inclduing the next exercise. Student behavior can be studied at a macro level, watching students' transitions through all of the material. On a micro level, within exercise transitions can be observed that may be indicative different stages of cognitive mastery. Most immediately, we wish to develop the tools that will allow us to link plotted data points to their raw data so that the semantics of the space and the rationale for transitions can be investigated.

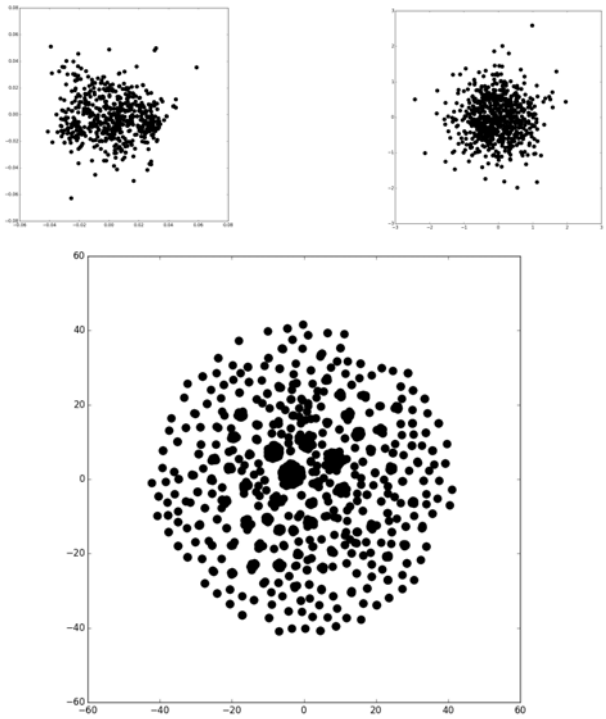

Figure 1. Python Model, Barnes-Hut (SKL), and Barnes-Hut (LM) produced plots from left to right, top to bottom.

4. ACKNOWLEDGMENTS

We would like to thank Khan Academy for their collaboration in providing a sample of de-identified student data to us for research purposes and to NSF for supporting the student author under BIGDATA Award #1547055.

5. REFERENCES

[1] L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms**.** *Journal of Machine Learning Research* 15(Oct):3221-3245, 2014.

[2] L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research* 9(Nov):2579-2605, 2008.

[3] L.J.P. van der Maaten. Learning a Parametric Embedding by Preserving Local Structure. In *Proceedings of the Twelfth International Conference on Artificial Intelligence & Statistics (AI-STATS), JMLR W&CP* 5:384-391, 2009.

[4] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep Knowledge Tracing. *Proceedings of the 29th Conference on Neural Information Processing Systems, Montreal, Canada* (pp. 505-513).

[2] https://www.kaggle.com/c/MerckActivity/prospector#186

[3] https://www.youtube.com/watch?v=eCCJ54rAEFY

[4] https://github.com/CAHLR/khan_visualization