Incorporating Rich Features into Deep Knowledge Tracing

Liang Zhang
Worcester Polytechnic Institute
Worcester, MA 01609 USA
lzhang6@wpi.edu

Xiaolu Xiong
Worcester Polytechnic Institute
Worcester, MA 01609 USA
xxiong@wpi.edu

Siyuan Zhao
Worcester Polytechnic Institute
Worcester, MA 01609 USA
szhao@wpi.edu

Anthony Botelho
Worcester Polytechnic Institute
Worcester, MA 01609 USA
abotelho@wpi.edu

Neil T. Heffernan
Worcester Polytechnic Institute
Worcester, MA 01609 USA
nth@wpi.edu

ABSTRACT

The desire to follow student learning within intelligent tutoring systems in near real time has led to the development of several models anticipating the correctness of the next item as students work through an assignment. Such models have included Bayesian Knowledge Tracing (BKT), Performance Factors Analysis (PFA), and more recently with developments in deep learning, Deep Knowledge Tracing (DKT). This DKT model, based on the use of a recurrent neural network, exhibited promising results. Thus far, however, the model has only considered the knowledge components of the problems and correctness as input, neglecting the breadth of other features collected by computer-based learning platforms. This work seeks to improve upon the DKT model by incorporating more features at the problem-level. With this higher dimensional input, an adaption to the original DKT model structure is also proposed, incorporating an auto-encoder network layer to convert the input into a low dimensional feature vector to reduce both the resource requirement and time needed to train. Experiment results show that our adapted DKT model, observing more combinations of features, can effectively improve accuracy.

ACM Classification Keywords

H.5.m Information interfaces and presentation: Miscellaneous

Author Keywords

Knowledge Tracing, Deep Learning, Deep Knowledge Tracing (DKT), Recurrent Neural Networks (RNN), Auto Encoder

1. INTRODUCTION

Models that attempt to follow the progression of student learning often represent student knowledge as a latent variable. As students work on new problems, these models update their estimates of student knowledge based on the correctness of responses. The problem emerges to be time series prediction, as student performance on previous items is indicative of future performance. Models then use the series of questions a student has attempted previously and the correctness of each question to predict the student's performance on a new problem.

Two well-known models, Bayesian Knowledge tracing (BKT) [2] and performance factor analysis (PFA) [5] have been widely explored due to their ability to capture this progression of knowledge with reliable accuracy. Both of these models, exhibiting success in terms of predictive accuracy, use differing algorithms to estimate student knowledge. BKT uses a Bayesian network to learn four parameters per knowledge component, or skill, while the PFA model uses a logistic regression over aggregated performance to determine performance for each skill.

Deep learning is an emerging approach which has proved to yield promising results in a range of areas including pattern recognition, natural language processing and image classification. The *deep* aspect of deep learning refers to the multiple levels of transformation that occur between input nodes and output nodes; these levels are usually referred to as layers, with each layer consisting of numerous nodes. The hidden nodes are used to extract high level features from previous layers and pass that information on to the next layer. However, the features extracted by deep learning are largely uninterpretable due to the complexity. This complexity makes it infeasible to explain the meaning behind every parameter learned by the model, unlike BKT and PFA which attempt to incorporate interpretability with its estimates.

Many deep learning algorithms like Recurrent Neural Network (RNN) and Convolutional Neural Networks (CNN) have been proposed in recent years to benefit machine learning systems with complex, yet more accurate representative models. Such an attempt in the field of learning analytics is that of Deep Knowledge Tracing (DKT) [6]. Building from the promising results of that model, this work seeks to make better use of the complex nature of deep learning models to incorporate more features to improve predictive accuracy. We also explore

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

L@S 2017 April 20-21, 2017, Cambridge, MA

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 123-4567-24-567/08/06…$15.00

DOI: http://dx.doi.org/10.475/123_4

how other deep learning structures can help reduce these high dimensional inputs into smaller representative feature vectors.

2. DEEP LEARNING IN EDUCATION

Deep knowledge tracing (DKT), introduced in paper [6], applies a RNN for this educational data mining task of following the progression of student knowledge. Similar to BKT, this adaptation observes knowledge at both the skill level, observing which knowledge component is involved in the task, and the problem level, observing correctness of each problem. The input layer of the DKT model is described as an exercise-performance pair of a student. In other words, the skill and correctness of each item is used to predict the correctness of the next item, given that problem's skill.

The DKT algorithm uses a RNN to represent latent knowledge state, along with its temporal dynamics. As a student progresses through an assignment, it attempts to utilize information from previous timesteps, or problems, to make better inferences regarding future performance. Specifically, the DKT model implements a popular variant of RNN, Long Short-Term Memory (LSTM), that employs cell states and three *gates* to determine how much information to remember from previous timesteps and also how to combine that memory with information from the current timestep.

Due to the recency of the DKT model, it is not as deeply researched as other established methods. We believe that DKT is a promising approach due to its comparable performance, and with the emergence of new neural network optimization algorithms, the structure has space for improvement. Thus far, only question (or skill) and correctness are considered as input to the model, but the neural network can easily consider more features. In this paper, we explore the inclusion of more features to improve the predictive accuracy of the model. In addition to these added features, we explore the usage of an Undercomplete Auto Encoder that incorporates a small central layer to convert high dimensional data to low dimensional representative encodings in order to increase the feasibility of implementing feature vectors of larger dimensionalities.

3. IMPROVING DKT WITH MORE FEATURES

Intelligent tutoring systems (ITS) often collect numerous features as students work, including information pertaining to problems, instructional aid, and time spent on individual tasks. Models and algorithms that make use of these additional information have been proposed. For example,students response time, hint request and number of attempts are added to make better student model [3]. In this paper, we do something similar using these extra features. In our experiment, students responce time, attempt number, and first action are selected for consideration as these are recorded by almost all such learning platforms. All input information is converted into a sequence of fixed-length input vectors in the RNN model, representing problem-level covariates while working through a possible multitude of assignments.

3.1 Feature process

Feature engineering plays a vital role in representing features effectively. The goal of this process, as it pertains to this work and coincides with how input is represented in the DKT model, is to convert the features to categorical data to simplify the input without losing much information. This process is described briefly for each considered feature as follows:

- *Exercise tag* exhibits differing representations described by either a numeric skill id or the name of the knowledge component in different dataset. Regardless of representation in the data, this is strictly categorical and is handled as such.
- *Correctness* is represented as a binary value where 1 indicates correctness and 0 represents an incorrect response.
- *First Response Time* is z-scored within skill and discretized based on its relationship with correctness as shown in Figure 1.

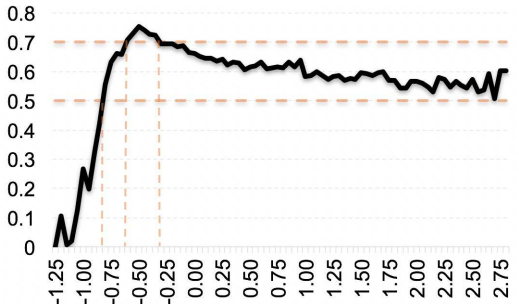

Figure 1. ASSISTments 2009 dataset z-scored time feature (*X axis*) and problem correctness (*Y axis*). Two reference lines of correctness, 50% and 70%, are selected to form the discretized boundaries.

- *Attempt count* is the number of attempts needed to correctly answer each item and are discretized as $[0, 1, other]$ in ASSISTments and $[== 1, 1 < and <= 5, > 5, other]$ in Open Learning Initiative as described further in a later section.
- *First action* is strictly categorical, representing if a student makes an attempt or requests help within the system as a first action.

After converting to categorical data, features are represented as a sparse vector by a one-hot encoding. These form cross features, where, for example, correctness is expressed as two values (correct and incorrect) for each skill. The combination of some features into cross features can improve model accuracy. The cross features of exercise and correctness, as well as time and correctness are selected in our model. All the ecodeed features are concatenated to construct the input vector in Figure 2.

Using cross features leads to a rapid increase of the dimensionality of the input vector. RNNs are considerably more computationally expensive due to the comparatively larger number of parameters. For example, training a LSTM DKT model with 50 skills and 200 hidden nodes, which needs to learn 250,850 parameters, takes 3.5 minutes per epoch, equating to more than 14 hours when using a 5 fold cross validation run over 50 epochs. To this extent, the network structure of DKT may benefit from reduced dimensionality, particularly if this can be achieved without sacrificing performance. Auto Encoder[4] is one such approach to this problem. It is a multi-layer neural network with a small central layer that can convert

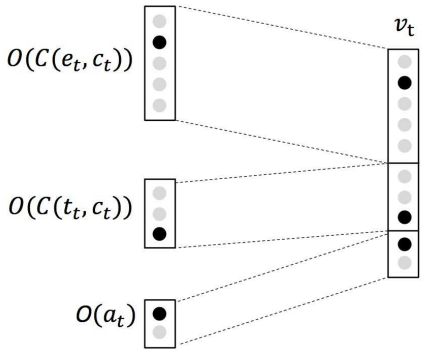

Figure 2. Concatenated encoded features to input layer

high dimensional data to low dimensional representative encodings that can be used to reconstruct the high dimensional input vectors; in this way dimensionality is reduced without the loss of too much important information.3 Once trained, the output layer can be removed, and the hidden layer can connect to another network layer. Auto Encoder may be stacked in this way but each layer must be trained one at a time. Like other neural network, the gradient descent method is used to train the weight values of the parameters. In our experiment, the dimension is reduced to a half of the input size.

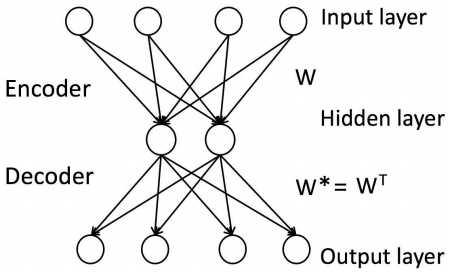

Figure 3. Representation of an Auto Encoder that reconstructs its input layer from a hidden layer of smaller dimensionality.

3.2 Model

[1] The input vector is constructed by concatenating one-hot encodings for separate features as illustrated in Figure 2, where v_t represents the resulting input vector of each student exercise. e_t refers to the exercise tag, while c_t refers to correctness, and t_t represents time.

$$C(e_t, c_t) = e_t + (max(e) + 1) * c_t \quad (1)$$

$$v_t = O(C(e_t, c_t)) \frown O(C(t_t, c_t)) \frown O(t_t) \quad (2)$$

$$v'_t = tanh(W_{ae} * v_t + b_{ae}) \quad (3)$$

$C()$ is the cross feature, $O()$ is the one-hot encoder format, and the $\frown$ operator is used to denote concatenation. In Equation 1, 1 is added to represent the unincluded exercise. Figure 4 depicts the resulting model representation utilizing an Encoder layer to support the added features. v'_t represents the feature vector extracted by Auto Encoder according to Equation 3.

[1]https://github.com/lzhang6/DKT-extension

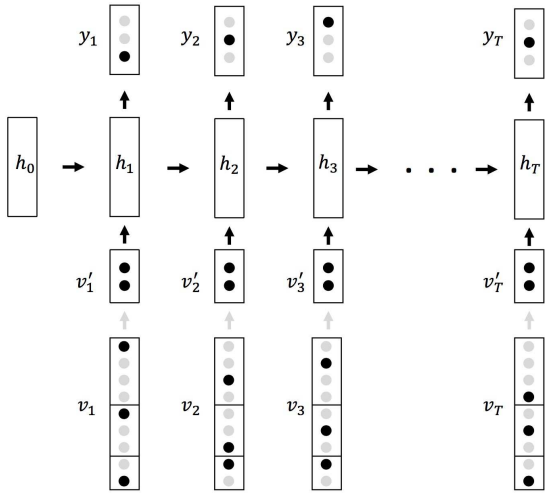

Figure 4. New DKT LSTM Model to incorporate more featuers with fixed Auto Encoder weights

The gray arrows mean that weights between the two layers are held constant, meaning the encoder weights are trained separately in advance. h_t represents the LSTM hidden nodes while y_t represents output layers nodes. The performance of every exercise is predicted but just one is supervised because only one label exists at each time step. We use a loss function of binary cross entropy. Only one LSTM layer with 200 hidden nodes is used and a dropout probability of 0.4 is applied during training.

4. DATASETS

ASSISTments and Open Learning Initiative (OLI) are computer-based learning platforms which embed practice and assessment throughout the learning process. Table 1 show running hyperparamters and information on the two data sets.

	ASSISTments 2009-2010	OLI Statics F2011
Student	3,866	332
Skill	124	82
Record	303k	257k
TimeStep	1,218	1,500
BatchSize	30	10
Epoch	40	40

Table 1. ASSISTments 2009-2010 and Statics 2011 dataset statistics and running hyper parameter setting

ASSISTments 2009-2010[2] dataset was gathered from mastery-based, skill builder problem sets. Three issues discovered [7] had unintentionally inflated the performance of DKT in initial reported results, so the updated version is utilized here.

[2]https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010

Model	Auto Encoder	ASSISTments		OLI Statics	
		$AUC(\%)$	R^2	$AUC(\%)$	R^2
Baseline: skill/correct	No	83.1 ± 0.6	0.324 ± 0.012	70.6 ± 0.7	0.105 ± 0.009
Baseline⌢time/correct	No	$\mathbf{85.8} \pm 0.7$	$\mathbf{0.391} \pm 0.015$	$\mathbf{73.1} \pm 0.5$	$\mathbf{0.135} \pm 0.010$
Baseline⌢time/correct	Yes	86.7 ± 0.4	0.410 ± 0.008	73.5 ± 0.9	0.142 ± 0.012
Baseline⌢time/correct⌢time⌢action⌢attempt	No	86.1 ± 0.4	0.398 ± 0.011	73.2 ± 0.5	0.140 ± 0.011
Baseline⌢time/correct⌢time⌢action⌢attempt	Yes	86.7 ± 0.2	0.411 ± 0.005	74.0 ± 0.5	0.148 ± 0.009
Baseline⌢time/correct⌢time/skill⌢time⌢action⌢attempt	Yes	$\mathbf{86.7} \pm 0.5$	$\mathbf{0.412} \pm 0.012$	$\mathbf{74.0} \pm 0.9$	$\mathbf{0.147} \pm 0.016$

Table 2. Test Result

OLI Statics F2011[3] dataset was from a college-level engineering statics course in OLI. The exercise tag is a numerated knowledge component derived from the text description.

Since it is a time-series algorithm, students whose records contain less than 2 time steps are not considered.

5. RESULTS

We use a 5-fold student level cross validation and the result is evaluated by AUC and square of Pearson correlation (R^2). Many possible feature combinations exist, but only a select few are explored here.

On both datasets in Table 2, models with incorporated features outperform the original DKT model. In the ASSISTments 2009 dataset, AUC value is improved to 85.8 from 83.1 and R^2 value increases to 0.391 from 0.342 after adding the cross feature of skill and correctness. In the Statics 2011 dataset, the AUC value increases to 73.1 from 70.6 and R^2 value is from 0.105 to 0.135 if only add cross feature of skill and correctness. Actually, if only incorporating cross feature of skill and correctness, the dimension of input layer dimension just increases $8,4(time) * 2(correctness)$ so that it almost has smiliar running efficiency as original DKT model. However, it exhibits only a marginal increase to this upon adding time, first action, and attempt count into the input vectors.

The adoption of Auto Encoder when compared to models using the same features also shows increased performance, supporting its usage for reducing dimensionality. In the ASSISTments 2009 dataset, AUC value is improved to to 86.7 from 85.8 and R^2 value increases from 0.391 to 0.410. While in OLI Statics dataset, AUC value is from 73.1 to 73.5 while R^2 value is from 0.135 to 0.142. In our analyses, the model incorporating all features in the last record in Table 2 was not even feasible without the use of this auto encoder, deeming it necessary to use when given large dimensional inputs.

6. DISCUSSION

Extending this model encompasses several potential directions to pursue. One such direction can explore even more student features like class-level features and school-level features, engineered in different manners, such as tokening the words of knowledge components for different exercise representations. Similarly, a wide and deep approach can be explored in how the features are represented within model training. The numerical data like time and hint usage can also be revisited in future work. Another direction is the usage of differnet dimensionality reduction methods like Principal Component Analysis (PCA) and Locally Linear Embedding (LLE), or different Auto Encoder methods, like adding noise, stacking layers or using RBM for initial weights. Because of flexible structure of deep learning, another research direction is to use similar RNN model structures to multi-task predictions in education data mining such as wheel spinning [1], student dropout, or hint usage.

[3]https://pslcdatashop.web.cmu.edu/Project?id=48

ACKNOWLEDGMENTS

We acknowledge funding from multiple NSF grants (ACI-1440753, DRL-1252297, DRL-1109483, DRL-1316736 & DRL-1031398), the U.S. Department of Education (IES R305A120125 & R305C100024 and GAANN), the ONR, and the Gates Foundation.

REFERENCES

1. J. E. Beck and Y. Gong. 2013. Wheel-spinning: Students who fail to master a skill. In *International Conference on Artificial Intelligence in Education*. Springer, 431–440.
2. A. T. Corbett and J. R. Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* 4, 4 (1994), 253–278.
3. M. Feng, N.T. Heffernan, and K. R. Koedinger. 2009. Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction* 19, 3 (2009), 243–266.
4. G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.
5. P.I. PAVLIK JRa, H. Cen, and K. R. Koedinger. 2009. Performance Factors Analysis–A New Alternative to Knowledge Tracing. *Online Submission* (2009).
6. C. Piech, J. Bassen, J. Huang, M. Sahami S. Ganguli, L. Guibas, and J. Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*. 505–513.
7. X. Xiong, S. Zhao, E.G. VanInwege, and J. E. Beck. 2016. Going deeper with deep knowledge tracing. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016)*. 545–550.