GRAPH-BASED KNOWLEDGE TRACING: MODELING STUDENT PROFICIENCY USING GRAPH NEURAL NETWORK

Hiromi Nakagawa, Yusuke Iwasawa & Yutaka Matsuo
Department of Engineering, The University of Tokyo
{nakagawa,iwasawa,matsuo}@weblab.t.u-tokyo.ac.jp

ABSTRACT

We apply graph neural network (GNN) to a new area, *knowledge tracing*. Knowledge tracing predicts student performance on coursework exercises over time. From the viewpoint of data structure, coursework can be potentially structured as a graph. Incorporating such a graph-structured nature to the knowledge tracing model as a relational inductive bias can improve performance; however, previous methods, such as Deep Knowledge Tracing (DKT), do not consider such a latent graph structure. Inspired by the recent successes of GNN, we propose a GNN-based knowledge tracing method, graph-based knowledge tracing (GKT). Casting the knowledge structure as a graph, we reformulate the knowledge tracing task as a time series node-level classification problem in GNN. Since the knowledge graph structure is not explicitly given in many cases, we propose various implementations of the graph structure. Empirical validations on two open datasets showed that our method outperforms past methods in predicting student performance. Moreover, the model provides better interpretable predictions than the previous methods.

1 INTRODUCTION

We apply graph neural network (GNN) to a new area, *knowledge tracing* (Corbett & Anderson, 1994), which entails predicting student performance on coursework exercises over time. Accurate prediction helps students to identify contents suitable for their individual knowledge level and facilitates more efficient learning. From the viewpoint of data structure, coursework can be potentially structured as a graph $G = (V, E, \mathbf{A})$; the requirements for mastering the coursework are decomposed into N knowledge concepts, known as nodes $V = \{v_1, \cdots, v_N\}$, and these concepts share dependency relationships, known as edges $E \subseteq V \times V$, where the degree of dependency is defined by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. In addition, we assume a student having a temporal knowledge state $h^t = \{h^t_{i \in V}\}$ for each concept independently at time step t, and the knowledge state is updated over time as follows: When the student answers an exercise associated with concept v_i, the student ' s knowledge state for not only the answered concept itself h^t_i, but also for its neighboring concepts $h^t_{j \in \mathcal{N}_i}$ is updated. Here, $\mathcal{N}_i$ denotes a set of nodes neighboring v_i.

Incorporating such a graph-structured nature of knowledge into a knowledge tracing model as a relational inductive bias can improve the model's performance and interpretability (Battaglia et al., 2018); however, previous methods do not consider such a nature. Knowledge tracing predicts $\mathbf{y}_t \simeq \mathbf{x}_{t+1}$ based on the student ' s time-series exercise logs $\mathbf{x}_1, \cdots, \mathbf{x}_t$, where $\mathbf{x}_t = \{q_t, r_t\}$ is a tuple that considers an answered exercise q_t as well as whether the exercise was answered correctly r_t. Deep Knowledge Tracing (DKT) (Piech et al., 2015) was the first deep learning-based method that demonstrated remarkable performance compared to the traditional methods such as Bayesian Knowledge Tracing (Corbett & Anderson, 1994). However, DKT adopted a simple recurrent neural network (RNN) architecture and this caused two shortcomings: 1) Representing the student temporal knowledge state using a single hidden vector complicated the modeling of the knowledge state for each concept separately. 2) Simple model architecture that embeds an input vector and propagates it to a recurrent layer made it difficult to model or reflect complex relationships between concepts. Zhang et al. (2016) proposed Dynamic Key-Value Memory Network (DKVMN), which utilizes two

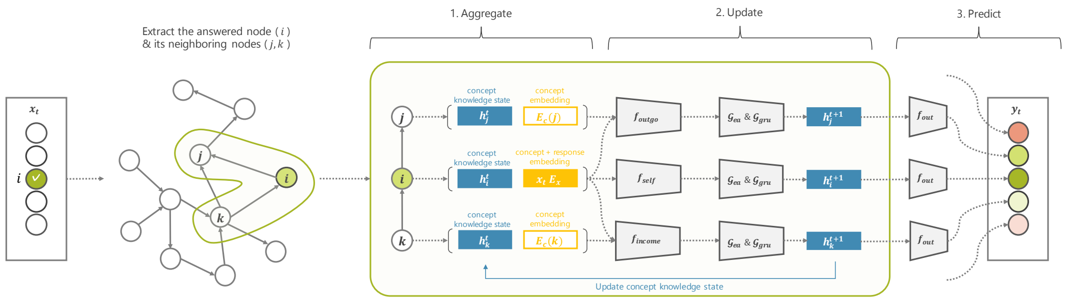


Figure 1: Architecture of GKT. When a student answers a concept, GKT first aggregates node features related to the answered concept, then updates the student's knowledge states for only related concepts, and finally predicts the probability of the student answering each concept correctly at the next time step.

memory matrices and the simple attention mechanism to overcome these shortcomings; nonetheless, the model could not address complex and multiple relationships. Recent successes of GNN inspired us to reformulate knowledge tracing as a GNN task and develop a new model to address these shortcomings.

In this paper, we propose a GNN-based knowledge tracing method, graph-based knowledge tracing (GKT). Casting the knowledge structure as a graph, where nodes correspond to concepts and edges correspond to the relationships between them, we reformulate the knowledge tracing task as a time series node-level classification problem in GNN. Empirical validations on two open datasets showed that our method can predict student performance to a better degree, and the model shows more interpretable predictions than the previous methods. Our contributions are as follows: 1) We show that knowledge tracing can be reformulated as an application of GNN. 2) To implement the graph structure that needs to be fed into the model, which is not explicitly given in many cases, we propose various methods and compared them using empirical validations. 3) We show that the proposed method results in more accurate and interpretable predictions than the previous deep learning-based methods as it incorporates the graph-structured nature of knowledge.

2 Graph-based Knowledge Tracing

2.1 Proposed Method

GKT applies GNN to the knowledge tracing task and leverages the graph-structured nature of knowledge. We present the architecture of GKT in Figure 1. The following paragraphs explain the processes in detail.

Aggregate First, the model aggregates hidden states and embeddings for the answered concept i and its neighboring concepts $j \in \mathcal{N}_i$:

$$\mathbf{h}'^{t}_{k} = \begin{cases} [\mathbf{h}^t_k, \mathbf{x}^t\, \mathbf{E_x}] & (k = i) \\ [\mathbf{h}^t_k, \mathbf{E}_c(k)] & (k \neq i) \end{cases}$$

where $\mathbf{x}_t \in \{0,1\}^{2N}$ is an input vector that represents which exercise was answered correctly and incorrectly at time step t, $\mathbf{E_x} \in \mathbb{R}^{2N \times e}$ is a matrix embedding the concept index and response of answers, $\mathbf{E_c} \in \mathbb{R}^{N \times e}$ is a matrix embedding the concept index, $\mathbf{E_c}(k)$ represents the k-th row of $\mathbf{E_c}$, and e is the embedding size.

Update Next, the model updates the hidden states based on the aggregated features and the knowledge graph structure:

$$\begin{aligned} \mathbf{m}^{t+1}_k &= \begin{cases} f_{\text{self}}(\mathbf{h}'^{t}_{k}) & (k = i) \\ f_{\text{neighbor}}(\mathbf{h}'^{t}_{i}, \mathbf{h}'^{t}_{k}) & (k \neq i) \end{cases} \\ \tilde{\mathbf{m}}^{t+1}_k &= \mathcal{G}_{ea}(\mathbf{m}^{t+1}_k) \\ \mathbf{h}^{t+1}_k &= \mathcal{G}_{gru}(\tilde{\mathbf{m}}^{t+1}_k, \mathbf{h}^t_k) \end{aligned} \quad (1)$$

where f_{self} is multilayer perceptron (MLP), $\mathcal{G}_{ea}$ is an erase-add gate used in Zhang et al. (2016), and $\mathcal{G}_{gru}$ is a gated recurrent unit (GRU) gate (Cho et al., 2014). f_{neighbor} is an arbitrary function that defines information propagation to neighboring nodes based on some knowledge graph structure, and we propose various implementations of f_{neighbor} in section 2.2.

Predict Finally, the model outputs the predicted probability of a student answering each concept correctly at the next time step: $\mathbf{y}_k^t = \sigma(\mathbf{W}_{\text{out}}\mathbf{h}_k^{t+1} + \mathbf{b}_k)$, where $\mathbf{W}_{\text{out}}$ is a weight matrix common for all nodes, $\mathbf{b}_k$ is a bias term for node k, and σ is a sigmoid function. The model is trained to minimize the negative log-likelihood (NLL) of the observations.

2.2 Implementation of Latent Graph Structure and f_{neighbor}

GKT can leverage a graph-structured nature of knowledge for knowledge tracing; however, the graph structure itself is not explicitly given in many cases. To implement the latent graph structure and f_{neighbor} in equation 1, we introduce two approaches.

Statistics-based Approach The statistics-based approach implements the adjacency matrix $\mathbf{A}$ based on some statistics and applies it to f_{neighbor} as follows:

$$f_{\text{neighbor}}(\mathbf{h'}_i^t, \mathbf{h'}_j^t) = \mathbf{A}_{i,j} f_{\text{outgo}}([\mathbf{h'}_i^t, \mathbf{h'}_j^t]) + \mathbf{A}_{j,i} f_{\text{income}}([\mathbf{h'}_i^t, \mathbf{h'}_j^t]) \tag{2}$$

where f_{outgo} and f_{income} are MLP. Here, we introduce three types of graphs.

Dense Graph is a simple dense graph, where $\mathbf{A}_{i,j}$ is $\frac{1}{|V|-1}$ if $i \neq j$; else, it is 0.

Transition Graph is a transition probability matrix, where $\mathbf{A}_{i,j}$ is $\frac{n_{i,j}}{\sum_k n_{i,k}}$ if $i \neq j$; else, it is 0. Here, $n_{i,j}$ represents how many times concept j was answered just after concept i was answered.

DKT Graph is a graph generated based on the conditional prediction probability of the trained DKT model, which was proposed by Piech et al. (2015).

Learning-based Approach The learning-based approach learns the graph structure in parallel with the optimization of the performance prediction in an end-to-end manner. Here, we introduce three methods to learn the graph structure.

Parametric Adjacency Matrix (PAM) simply parameterizes the adjacency matrix $\mathbf{A}$ and optimizes it with other parameters under some constraints so that $\mathbf{A}$ satisfies the property of an adjacency matrix. f_{neighbor} is defined in the same manner as equation 2.

Multi-head Attention (MHA) leverages the multi-head attention mechanism (Vaswani et al., 2017) to infer the edge weights between two nodes based on node features. f_{neighbor} is defined as follows:

$$f_{\text{neighbor}}(\mathbf{h'}_i^t, \mathbf{h'}_j^t) = \frac{1}{K} \sum_{k \in K} \alpha_{ij}^k f_k(\mathbf{h'}_i^t, \mathbf{h'}_j^t)$$

where k is the head index among a total of K heads, α_{ij}^k is the k-th head's attention weight from v_i to v_j, and f_k is a neural network for the k-th head.

Variational Autoencoder (VAE) assumes the discrete latent variable that represents the types of edges and infers them based on node features. f_{neighbor} is defined as follows:

$$f_{\text{neighbor}}(\mathbf{h'}_i^t, \mathbf{h'}_j^t) = \sum_{k \in K} z_{ij}^k f_k(\mathbf{h'}_i^t, \mathbf{h'}_j^t)$$

where k is the edge type among a total of K types, z_{ij}^k is a latent variable sampled from the Gumbel-Softmax distribution (Maddison et al., 2016), and f_k is a neural network for the k-th edge type. VAE minimizes not only NLL but also the Kullback-Leibler divergence between the encoded distribution $q(\mathbf{z}|\mathbf{x})$ and prior distribution $p(\mathbf{z})$. Using one of the K edge types to represent the "non-edge" class, which means that no messages are passed along this edge type, and setting high probability on the "non-edge" label encourages generation of a sparse graph.

The learning-based approach is close to the idea of edge feature learning (Battaglia et al., 2016; Gilmer et al., 2017; Chen et al., 2018), and MHA and VAE were motivated by graph attention network (GAT) (Velickovic et al., 2017) and neural relational inference (NRI) (Kipf et al., 2018), respectively; however, we modify them with regard to two points. First, we calculate the edge weights

Table 1: Comparison of prediction performance.

Method		AUC	
		ASSISTments	KDDCup
Baseline	DKT	0.709	0.751
	DKVMN	0.710	0.753
Statistics-based	Dense Graph	0.722	0.762
	Transition Graph	0.721	**0.769**
	DKT Graph	**0.723**	0.764
Learning-based	PAM	0.719	0.762
	MHA	**0.723**	0.766
	VAE	0.722	**0.769**

based on static features such as embeddings of concepts and responses instead of dynamic features. This enables us to learn a knowledge graph structure invariant to students and time steps, which is more natural for knowledge tracing settings. Second, for VAE, we limit the edge-type inference for nodes related to the answers at each time step. This fits the knowledge tracing situation where students answer only a small subset of concepts at each time step, and it also helps to reduce the computational cost from $\mathcal{O}(KN^2)$ of the original NRI to $\mathcal{O}(KN)$.

3 Experiments

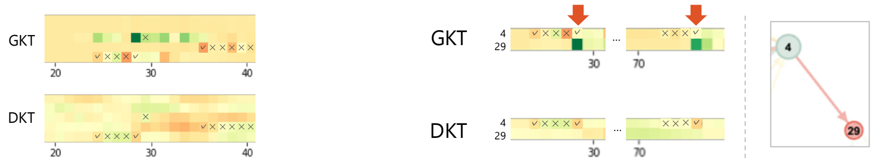
Datasets For the experiments, we use two open datasets of student math exercise logs: ASSISTments 2009-2010 "skill-builder" provided by the online educational service ASSISTments[1] (hereinafter called "ASSISTments") and Bridge to Algebra 2006-2007 (Stamper et al., 2010) used in the Educational Data Mining Challenge of KDDCup (hereinafter called "KDDCup"). We show preprocessing procedures and the dataset statistics in Appendix A.

Prediction Performance First, we evaluate the prediction performance of GKT. We set DKT and DKVMN as baselines and compare the Area Under the Curve (AUC) score of GKT with them. We show the implementation details in Appendix B. We present the results in Table 1. We show the highest scores in bold in each dataset. In both datasets, GKT showed the highest AUC score. This suggests that GKT can trace student knowledge state better than the previous methods, which do not consider the knowledge graph structure. In the statistics-based approaches, the Transition Graph or DKT Graph outperformed the simple Dense Graph. This indicates that representing sparse relationships between nodes in some way enables GKT to perform better. In the learning-based approaches, MHA or VAE, which estimate edge information using the neural network based on node features, performed better than the PAM, which simply optimizes the adjacency matrix. However, the best performance of the statistics-based and learning-based approaches was almost similar, and the methods that performed the best differed between the two datasets. Thus, further experiments over various datasets are needed.

We present network analysis of the learned graph structure extracted from the trained GKT model in Appendix C.

Interpretability of the Prediction Next, we visualize how the model predicted the student knowledge state change over time and evaluate the interpretability of the prediction. This visualization helps students and teachers to recognize the students' knowledge state efficiently and intuitively, and thus, its interpretability is important. Here, we evaluate interpretability based on the following two points: 1) Whether the model updates only the related concepts to the answered concept at each time step. 2) Whether the update is reasonable with the given graph structure. We randomly sample a student and depict the student knowledge state change to a subset of concepts in Figures 2a and 2b. We show the depiction procedure in detail in Appendix D. The *x*-axis and the *y*-axis show the time steps and concept indices respectively, and the cell color shows the extent to which the mastery level changed at the time step. Green denotes an increase and red denotes a decrease. We fill the elements answered correctly and incorrectly with "✓" and "$\times$" respectively. From Figure 2a, we can see that GKT clearly updates the knowledge state of related concepts only, although DKT updates

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



(a) Subset of frequently answered concepts. (b) Subset of concepts neighboring on the graph.

Figure 2: Visualization of the transition of the predicted knowledge states for concept subsets.

that of all the concepts indistinctly and cannot model the change in related concepts. In addition, Figure 2b shows that although concept 29 is not answered, its knowledge state is clearly updated at $t = 28$. In that time step, concept 4 is answered correctly, and the given graph shows an edge between concepts 4 and 29, as shown on the right-hand side of the figure. This suggests that GKT models the student knowledge state definitively based on the given graph. However, DKT does not display this behavior. These results show that GKT can model the student mastery level for each concept distinctively and reasonably and provide interpretable prediction.

4 Conclusion

In this paper, we proposed a GNN-based knowledge tracing method called GKT. Casting the knowledge structure as a graph, we reformulated the knowledge tracing task as a time-series node-level classification problem in GNN. Empirical validations on two open datasets showed that our method provides the best prediction of student performance and shows more interpretable predictions than the previous methods. These results confirm the potential of our proposed method to enhance performance and showcase the possibility of its application to real educational knowledge tracing environments, which could help improve the learning experience of students in more diverse environments.

References

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. 2018.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-adapted Interaction*, 4(4):253–278, 1994.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

Table 2: Dataset statistics.

Dataset	# students	# skills	# logs
ASSISTments	1,000	101	62,955
KDDCup	1,000	211	98,200

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, pp. 505–513, 2015.

J. Stamper, A. Niculescu-Mizil, S. Ritter, G.J. Gordon, and K.R Koedinger. Bridge to algebra 2006-2007. development data set from kdd cup 2010 educational data mining challenge. http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp, 2010.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 1(2), 2017.

Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory network for knowledge tracing. *arXiv preprint arXiv:1611.08108*, 2016.

A Dataset Preprocessing

We preprocess each dataset using the following conditions.

ASSISTments

1. Combine simultaneous answer logs into one.
2. From 1, extract the logs associated with the named skill tags.
3. From 2, extract the logs associated with the skill tags answered at least 10 times.

KDDCup

1. Consider the combination of *problem* and *step* as one answer.
2. From 1, extract the logs associated with the skill tags that are named and are not dummies.
3. From 2, extract the logs associated with the skill tags answered at least 10 times.

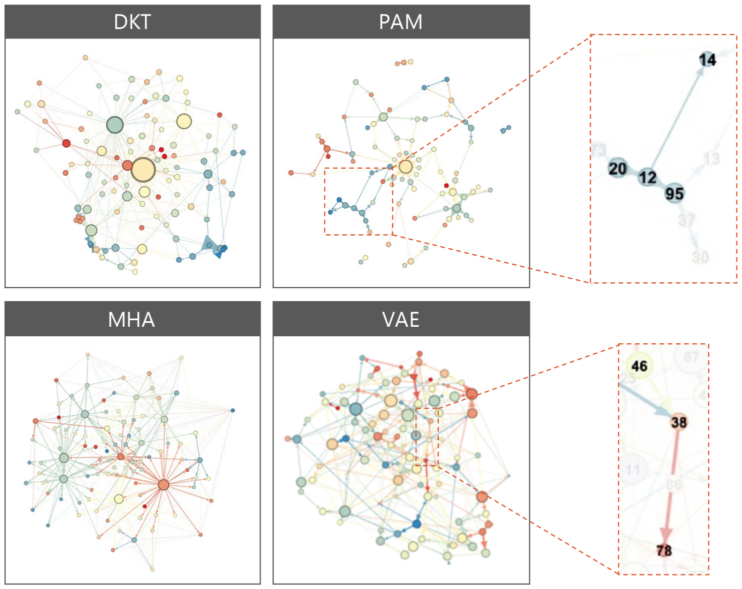
Combining simultaneous answer logs into one set prevents unfairly high prediction performance because of frequently co-occurring tags. Excluding skill tags that are unnamed or dummies removes noise. Thresholding logs with the number of times each skill tag was answered secures a sufficient number of logs in order to remove the noise. The statistics of the datasets preprocessed using the above conditions are shown in Table 2.

B Implementation Details

For each dataset, we divide the students into training:validation:test = 8:1:1. We train the model with the training students' data and adjust the hyperparameters with the validation students' data.

DKT We search for the hyperparameters following Piech et al. (2015). The size of the hidden layer is 200 and we use GRU for RNN. We apply dropout from $\mathbf{h}_t$ to $\mathbf{y}_t$ with a drop rate of 0.5. The batch size is 32, and we use Adam (Kingma & Ba, 2014) as an optimizer with a learning rate of 0.001.

DKVMN We search for hyperparameters following Zhang et al. (2016). The size of the memory slot is 20 for the ASSISTments dataset and 50 for the KDDCup dataset. The size of the hidden



12	Area Rectangle
14	Area Triangle
20	Complementary and Supplementary Angles
95	Volume Cylinder

38	Histogram as Table or Graph
46	Mode
78	Scatter Plot

Figure 3: Network visualizations of graphs derived from DKT and GKT.

vector is 32 for the ASSISTments dataset and 128 for the KDDCup dataset. The batch size is 32, and we use Adam as an optimizer with a learning rate of 0.001.

GKT The size of all hidden vectors and embedding matrix is 32. For the MLPs in the model, we apply dropout from the hidden vectors to the output vectors with a drop rate of 0.5, and apply batch normalization (Ioffe & Szegedy, 2015) for the output layers. The batch size is 16, and we use Adam as an optimizer with a learning rate of 0.01. We set $K = 2$ in MHA and VAE to fairly compare the learning-based approaches with the statistics-based approaches, since the statistics-based approaches assume two edge types: incoming edges and outgoing edges.

C Network Analysis

We extract the learned graph structure from the trained GKT model and analyze it. In the learning-based approaches, GKT learns the graph structure that helps to predict student performance. Thus, the graph extracted from the model that showed the high prediction performance can provide insights regarding a good knowledge structure. We show the networks in Figure 3. The color of the nodes is graded from blue to red, where the earlier an exercise is answered, the bluer the shade. The size of the nodes is proportional to their out-degree, which means that larger nodes influence more nodes. First, in the DKT Graph, which is visualized for comparison, similarly colored nodes are connected with each other, constructing clusters. As DKT models hidden states of all concepts with the same single hidden vector, modeling long time dependency between concepts is difficult. Thus, the model tends to learn the dependencies between nodes that are answered in a temporally close order. The graph extracted from PAM had a structure similar to that of the DKT Graph and constructed clusters; from the top right of the figure, we can perceive that some geometrical concepts are connected. The graph extracted from MHA shows many out-coming edges from certain nodes. Although it is probable that the model learned some special dependencies between concepts that differ from those in the other graphs, the model's prediction can be biased. Thus, we need to evaluate the effect on the prediction performance. The graph extracted from VAE differed from the other graphs in that it formed a dense graph, where many nodes were connected with each other. Although many of these connections are difficult for us to interpret, from the bottom right of the figure, we can identify some statistical concepts are connected.

D Knowledge State Change Depiction

Piech et al. (2015) and Zhang et al. (2016) performed analysis called knowledge state depiction to visualize the temporal transition in student knowledge state. Our study extends this analysis as follows to analyze the temporal transition more precisely for each concept at each time step.

1. Randomly sample a student log until time step T.
2. Remove bias vectors from the output layers in each trained model.
3. Input the student answer logs $\mathbf{x}_{t\leq T}$ to the trained model and stack the output $\mathbf{y}_{t\leq T}$.
4. Normalize the output values at each time step from 0 to 1.