# Computing Weights in Knowledge Tree

Subhalingam D and Vishal Turkar

May 31, 2020

## 1 Introduction

The aim was to find a technique to compute the edge-weights in the knowledge graph of each student and update them after each assessment. The ideas presented in this report are based on the paper proposed by Nakagawa et al. [1] and several ideas were taken from this paper. They compare their model with another knowledge tracing method, proposed by Piech et al. [2], who reported a method called deep knowledge tracing (DKT), which leverages recurrent neural networks (RNNs). The writers claim that their model would be useful when there is a need to model knowledge as graph instead of sequential data.

In the paper [1], they proposed a graph neural network (GNN)-based knowledge tracing method, i.e., a graph-based knowledge tracing (GKT). Their model reformulates knowledge tracing as a time-series node-level classification problem in the GNN. This formulation is based on three assumptions: 1) The coursework knowledge is decomposed into a certain number of knowledge concepts. 2) Students have their own temporal knowledge state, which represents their proficiency in the concepts of the coursework. 3) The coursework knowledge is structured as a graph, which affects the updating of student knowledge state: if a student answers a concept, correctly or incorrectly, his/her knowledge state is affected not only with regard to the answered concept but also other related concepts that are represented as neighboring nodes in the graph.

They claim that their model also provides more interpretable predictions, like the process of student proficiency, i.e., the concepts which the students gained understanding of, and the time required by them for the same, compared to the previous ones.

The writers' primary motivation for using GNN is the success of convolutional neural networks (CNNs). However, CNNs can only operate on regular Euclidean data such as images and texts, whereas several applications in the real world generate non- Euclidean data. The GNN, on the other hand, regards these non-Euclidean data structures as graphs and enables the same advantages of the CNN to be reflected on these highly diverse data.

# 2 Proposed Model

## 2.1 Setup

Let $G = (V, E)$ be a knowledge graph, where each skill is represented by nodes $V$ and their relationships are given by $E \subset V \times V$. Let $V = \{v_1, v_2, \cdot, v_N\}$ be $N$ nodes. In addition, we assume a student having a temporal knowledge state for each concept independently at time step $t$, $h^t = \{h_i^t\}$, and this knowledge state is updated over time as follows: when the student solves an exercise associated with concept $v_i$ , then the student's knowledge state for the answered concept itself $h_i^t$ and for its related concepts $h_{j \in N_i}^t$ is updated (here, $N_i$ denotes a set of nodes neighboring $v_i$)

## 2.2 Architecture

The following sections explain the processes in detail.

### 2.2.1 Aggregate

First, the model aggregates the hidden states and embeddings for the answered concept $i$ and its neighboring concepts $j \in N_i$ :

$$h_k'^t = \begin{cases} \left[h_k^t, x^t E_x\right] & (k = i) \\ \left[h_k^t, E_c(k)\right] & (k \neq i) \end{cases}$$

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

### 2.2.2 Update

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

$$m_k^{t+1} = \begin{cases} f_{\text{self}}(h_k'^t) & (k = i) \\ f_{\text{neighbour}}(h_i'^t, h_j'^t) & (k \neq i) \end{cases}$$
$$\tilde{m}_k^{t+1} = \mathcal{G}_{ea}(m_k^{t+1})$$
$$h_k^{t+1} = \mathcal{G}_{gru}(\tilde{m}_k^{t+1}, h_k^t)$$

where $f_{\text{self}}$ is a multilayer perceptron (MLP), $\mathcal{G}_{ea}$ is an erase–add gate used in Zhang et al.[3] and $\mathcal{G}_{gru}$ is a gated recurrent unit (GRU) gate. $f_{\text{neighbour}}$ is an arbitrary function (discussed in 2.3) that defines the information propagation to neighboring nodes based on a knowledge graph structure.

### 2.2.3 Predict

Finally, the model outputs the predicted probability of a student answering each concept correctly at the next time step:

$$y_k^t = \sigma(W_{\text{out}} h_k^{t+1} + b_k)$$

where ($W_{\text{out}}$ is a weight matrix common for all nodes, $b_k$ is a bias term for node $k$, and $\sigma$ is a sigmoid function. The model is trained to minimize the negative log like hood (NLL) of the observations.

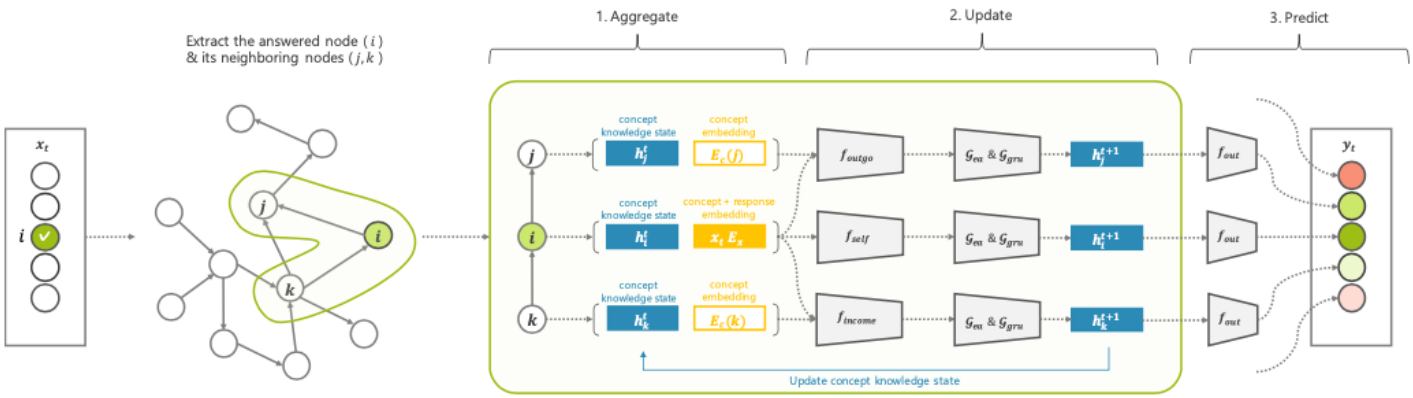


Figure 1: **Architecture of GKT.** When a student answers a concept, GKT first aggregates the node features related to the an- swered concept, subsequently updates the student's knowledge states for only the related concepts, and finally predicts the probability of the student answering each concept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

## 2.3 Implementation of Latent Graph Structure and $f_{\text{neighbor}}$

Two approaches are introduced to implement the latent graph structure when (the structure itself is not explicitly provided) and $f_{\text{neighbor}}$:

### 2.3.1 Statistics-based Approach

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

$$f_{\text{neighbor}}(h_i'^t, h_j'^t) = A_{i,j} f_{\text{out}}([h_i'^t, h_j'^t]) + A_{j,i} f_{\text{in}}([h_i'^t, h_j'^t])$$

where $f_{\text{out}}$ and $f_{\text{in}}$ are MLPs.

Three types of graphs are introduced in the article:

**1. Dense Graph:** Simple densely connected graph, where

$$A_{i,j} = \begin{cases} \frac{1}{|V|-1} & i \neq j \\ 0 & i = j \end{cases}$$

**2. Transition Graph:** Transition probability, matrix where

$$A_{i,j} = \begin{cases} \frac{n_{i,j}}{\sum_k n_{i,k}} & i \neq j \\ 0 & i = j \end{cases}$$

Here, $n_{i,j}$ represents the number of times concept $j$ was answered immediately after concept $i$ was answered.

**3. DKT Graph:** is a graph generated based on the conditional prediction probability of the trained DKT model, which was proposed by Piech et al. [2]

*Later it was found that the Transition Graph would give highest AUC score.*

#### 2.3.2 Learning Based approach

In this approach the graph structure is learned parallel with the optimization of the performance prediction. Three approaches for learning the graph structure are introduced:

**1. Parametric Adjacency Matrix (PAM)** simply parameterizes the adjacency matrix $A$ and optimizes it with the other parameters under certain constraints, such that $A$ satisfies the property of an adjacency matrix. $f_{\text{neighbor}}$ is defined similarly as before.

**2. Multi-Head Attention (MHA)** leverages the multi-head attention mechanism [4] to infer the edge weights between two nodes based on their features. $f_{\text{neighbor}}$ is defined as:

$$f_{\text{neighbor}}(h_i'^t, h_j'^t) = \frac{1}{k} \sum_k \alpha_{ij}^k f_k(h_i'^t, h_j'^t)$$

where $k$ is the head index among a total of $K$ heads, $\alpha_{ij}^k$ 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

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

$$f_{\text{neighbor}}(h_i'^t, h_j'^t) = \sum_k z_{ij}^k f_k(h_i'^t, h_j'^t)$$

where $k$ is the edge type among a total of $K$ types, $z_{i,j}^k$ is a latent variable sampled from Gumbel-Softmax Distribution [5] and $f_k$ is a neural network for the $k$-th edge type.

*Later, during the implementation, it was found that the Variational Autoencoder (VAE) approach performed better than rest of the two approaches.*

# 3 Discussion

It was shown that GKT updates the knowledge state of only the related concepts whereas DKT updates the state of all the concepts indistinctly and cannot model the change in related concepts. In addition, GKT definitively models the student knowledge state based on the given graph, however, DKT does not exhibit this behavior. These results indicate that GKT can model the student proficiency level for each concept distinctively and reasonably, and provide more interpretable predictions.

## 3.1 Differences Between Learning-based Approaches

Three different types of Learning Based Approaches are used **PAM, MHA and VAE**. The difference between PAM and the other two approaches is that in PAM the adjacency matrix is directly optimized and no condition exists to estimate the edge features while in MHA and VAE there exist some condition. The difference between MHA and VAE is in the method of calculating the edge weights. In MHA when a concept i is answered, scores are calculated and normalized across all neighboring concepts whereas in VAE, based on the pairs of answered concepts and each of its neighboring concepts, each edge feature is calculated independently, the Gumbel Softmax Function makes only one of the k edge weights near 1 and other near 0.

# References


[1] Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo, "Graph-based knowledge tracing: Modeling student proficiency using graph neural network," *IEEE/WIC/ACM International Conference on Web Intelligence*, 2019.

[2] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein, "Deep knowledge tracing," *Advances in Neural Information Processing Systems*, 2015.

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

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[5] 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.