Transcending Conventional Binary Labels: Revamping Knowledge Tracing with VAE-Generated Image Representation

Hui Zhao[1], Yanze Wang[1], and Jun Sun[1]✉

[1]Wangxuan Institute of Computer Technology, Peking University, No. 128 Zhongguancun North Street, Haidian District, Beijing, 100871, China
{hui.zhao, 2201111749}@stu.pku.edu.cn; sunjun@pku.edu.cn

Abstract. Knowledge tracking serves as a data mining task aimed at predicting students' learning performance. Nevertheless, without analyzing students' answer contents, it is unjustifiable to represent learning records solely by two states of correct and incorrect answers for the knowledge tracking task. To tackle this issue, we incorporate image data to stand for the answer result labels. Even for the same answer result label, different vectors are employed for representation. Firstly, handwritten digit recognition is introduced as an auxiliary task, and the VAE network is utilized to generate handwritten digit images. Secondly, the handwritten digit images generated by the VAE are used to represent the vectors of answer results. Finally, the auxiliary task is combined with the knowledge tracking task for joint training. Experimental findings suggest the introduced images can better represent the complex states of learning results, thus enhancing the performance of the model. On four knowledge tracking datasets, the average AUC index increases by nearly 2.8%.

Keywords: Knowledge tracing · Binary labels · VAE.

1 Introduction

Knowledge tracking (KT) is a model for predicting students' knowledge mastery status. It predicts the probability of a student answering the next exercise correctly through the historical answer records. The records of students' answer results used in kT only have two labels, correct and incorrect, which enables the model to ignore the answering process and the content of the exercises. Based on the excellent feature representation of deep learning, the model can still achieve outstanding prediction performance at the condition of binary labels in the data sets.

However, when students do exercises, they may accidentally answer the exercises wrongly due to carelessness; or they may guess correctly by chance although they don't actually master the knowledge. Besides, the two binary states of correct and incorrect answers cannot fully represent the students' mastery of knowledge, and it has representational limitations.

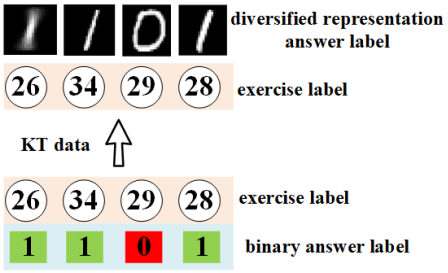

Fig. 1. A schematic diagram of replacing binary labels with the generated image data.

Previous studies have noticed this problem. They alleviated the unreasonableness of this problem to a certain extent by analyzing the content of the exercises, increasing the data of students' answer duration, relying on expert knowledge, etc [15]. However, these approaches increased the difficulty of data collection, consumed more human resources, and did not solve this problem in terms of the data representation of the answer status.

To address this issue, we introduce image data to represent the labels of the answer status. As shown in Figure 1, we have replaced the invariant answer status labels with the image data representations produced by the generative network. In this paper, we propose a new network named Multi-Task Fusion Network (MTFN), which enables a task to learn beneficial patterns from other task's data. In MTFN, there exists a complex primary task alongside a simple auxiliary task. These two tasks are entirely unrelated; however, through a certain data transformation, we represent the data in the primary task using the data from the auxiliary task. This transformation is referred to as data permeation. To ensure data consistency and availability during the data permeation operation, we employed generative networks to reconstruct the data from the auxiliary task. Subsequently, we feed the data from both tasks into networks specifically designed for them. The data from both tasks are linked together, and the enhancement of auxiliary task performance contributes to the improvement of primary task performance. We selected KT as the primary task and handwritten digit recognition as the auxiliary task. Experimental results demonstrate that with the assistance of the auxiliary task, KT achieved an average improvement of 2.8% in the scale of AUC (the area under receiver operating characteristic curve).

2 Related Work

KT uses students' historical learning records to predict how likely a certain student will get the next question right. The probability of prediction is regarded as the certain student's mastery of the corresponding knowledge. In this way, KT task can be dynamically worked through time series prediction [16].

KT has emerged as a popular research direction in the field of educational data mining and wisdom education. Early KT methods [10] were based on traditional statistical models, such as Bayesian Knowledge Tracing (BKT) [5, 12]. With the advancement of machine learning techniques, researchers have also explored various machine learning-based methods for KT [11]. Deep Knowledge Tracing (DKT) is a RNN based approach that models the sequential dependencies in student responses to predict their knowledge states. On the basis of DKT, network structures and mechanisms of other domains were introduced into the KT network [8]. Dynamic Key-Value Memory Network (DKVMN) [2] and its variants have been proposed to solve the weak explanatory problem. Self-attentive Knowledge Tracing (SAKT) [3] used transformer to capture correlations between practice interactions and achieved a good performance. Other deep learning-based approaches such as Long Short-Term Memory (LSTM), transformer module and attention mechanism have also been used for KT [9].

3 Method

3.1 Formulation for two tasks

In MTFN, we designated the KT task as the primary task. For KT task, we define a student's interactive learning records as $R_n = \{r_1, r_2, ..., r_n\}$, where r_t is the t-th interaction record. For each r_t, $r_t = (k_t, a_t)$, where k_t is the number label of knowledge concept corresponding to a certain exercise and at stands for the answered state (use 1 for correct answer state and 0 for incorrect) of the t-th exercise. Formally, we describe the KT task in terms of conditional probability, i.e. $P = p(a_n = 1|k_n, R_{n-1})$. In order to improve the reliability of the probability by quantity limited interactive data of a certain student, the KT model will take full advantage of the correlation between knowledge concept and use the interactive data of other students as a reference. In this way, the conditional probability can be revised as $P = p(a_n = 1|k_n, \Theta - r_n)$, where $\Theta = \{R_n, Q_m, ...\}$ and Q_m is similar to R_n, standing for another student's learning records.

The handwritten digit recognition task was re-designed to serve as the auxiliary task. The re-designed task involves classifying images of handwritten digits into one of two classes: 0 or 1. The reason for this design choice is that the answer data in the KT task is represented by 0s and 1s. The auxiliary task also consists of data with only 0s and 1s, which can be used to represent the data in the primary task. Through data permeation operation, the data from primary tasks can be associated with data from auxiliary task.

3.2 Data permeation operation

To permeate the handwritten digit recognition data into the KT data, we replaced the labels of 0 and 1 in the KT dataset with image data representing 0 and 1 from the former dataset, respectively. Simply using a single image data representing 0 or 1 would provide limited expressive power, essentially lacking

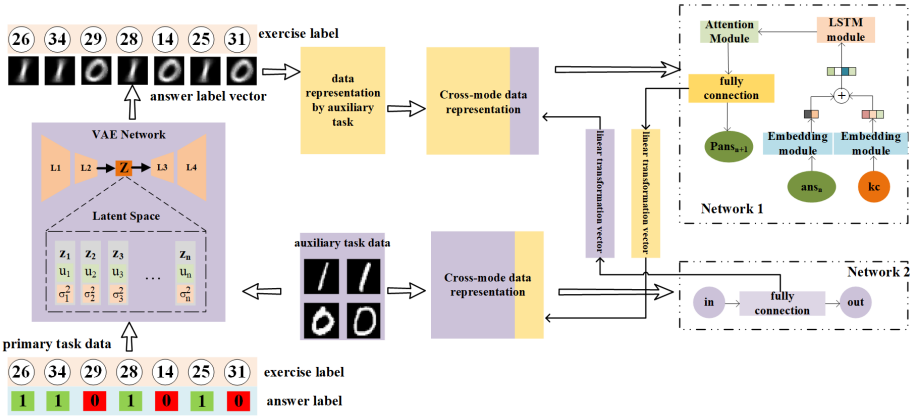

Fig. 2. The process of MTFN. The diagram comprises three main modules: the Data Permeation module, the Task Network module, and the Hybrid Modality module.

distinction from the original 0 or 1 label. To increase diversity and randomness in the representations, we employed a generative VAE network to generate images of 0 and 1.

As a generative probabilistic model that learns the latent representation of original data, following previous works [13, 14], our VAE model consists of encoder module and decoder module. As shown is Figure 2, the encoder part includes two linear layers that map the original data to the latent space, capturing its statistical properties. The decoder part of the VAE also includes two linear layers that symmetric with encoder, which mapping the latent representations back to the data space, generating reconstructions of the original data. Given the original learning interaction data O, we expect the encoder network learn low-dimensional features of the data as Z, which can be reconstructed to vectors as G that similar to O by decoder. Z is composed of multidimensional feature z_i, where i represents the i-th feature. The image data generated by the VAE follows a certain probability distribution, which enables the permeated data in the KT dataset to be represented more effectively. This process can be likened to a constrained dynamic embedding operation for the 0 and 1 labels. The superiority of this representation is reflected in subsequent training processes.

3.3 Hybrid modality network

In Figure 2, we adopt a cross approach to dynamically update the input data for both networks. Specifically, the representations of data from the primary task network and the auxiliary task network are extracted, combined with the original data from the other task, and then re-input into their respective networks. This operation is performed for each mini-batch of data, allowing the parameters in both networks to be updated. The updated parameters improve the effectiveness

of the data representations in the primary task network, consequently enhancing the performance of the primary task.

This cross-representation operation can be understood as a form of hybrid modality. The vectors resulting from cross-representation originate from the network of the other task. Since the data in the primary task is represented by the auxiliary task, the data from the auxiliary task can be regarded as another modality of the primary task data. This modality effectively compensates for the deficiencies of the data of primary task network, resulting in improved performance. The hybrid modality is dynamically updated, and it evolves along with the enhancement of both tasks, thereby altering the representation space.

In Figure 2, the additional modality information we designed comes from the fully connected layers of both networks. At the end of the fully connected layers, the output vectors undergo a linear transformation, and then they are concatenated with the input data from the other task.

3.4 Backbone network of the tasks

The auxiliary task is relatively simple, and we only designed a fully connected layer network to easily achieve the task's effectiveness. The following of this part mainly introduces the backbone network for KT. As shown in Figure 2, we design the KT backbone network which is composed of embedding module, attention module, long short-term memory (LSTM) module and full connection layers.

Input data operation. We first project the two types of sequences into the form of vector embedding. We define the embedding exercise sequences as $KC \in R^{L \times d_e}$ and the embedding answer sequences as $ANS \in R^{2 \times d_a}$, where L is the length of interactive data and d_e, d_a are embedding dimensions of the exercise sequences and answer sequences respectively. A certain student's interaction of question sequences and answer state sequences can be described as $s_n \in R^{1 \times d_e}$ and $ans_n \in R^{1 \times d_a}$, where s_n and ans_n could be looked up in S and ANS and n is the index. $Pans_{n+1}$represents the final output of the network, representing the predicted probability values by inputs.

Data processing procedure. In Figure 2, the two vectors after embedding are first concatenated and then input into the LSTM module, followed by the attention module. Finally, the vector output from the attention module undergoes a fully connected network, ultimately yielding the prediction result vector. We introduce the attention mechanism [7] due to the association of knowledge concept. The operation of *cumsum* denotes the cumulative sum of all items, and the operation of *cat* means simple concatenation of two vectors. Finally, it is worth mentioning that during the backpropagation operation, we sum the Binary Cross-Entropy (BCE) losses produced by the two networks, but assign different weights to the losses of different tasks. It can be formulated as follows:

$$\mathcal{L}_{total} = A \cdot \mathcal{L}_{primary} + B \cdot \mathcal{L}_{auxiliary}, \tag{1}$$

in which A$\gg$B.

3.5 Explaination and mathematical statement

We assume the primary task as PT, the auxiliary task as AT, and their datasets as D_{PT} and D_{AT}, respectively. The deep learning network corresponding to the primary task is denoted as W, which represents a complex function $W(D_{PT})$ that is difficult to describe. The ability of this network to solve PT is defined as the probability distribution $P(W(D_{PT}))$. Similarly, the network for the auxiliary task is denoted as Y, and its ability to solve AT is defined as the probability distribution $P(Y(D_{AT}))$. Since we use D_{AT} to describe D_{PT}, there exist a transformation T and $D_{PT} = T(D_{AT})$.

We define mutual information as $I(W(D_{PT}); Y(D_{AT}))$, which indicates the reduction in uncertainty of $W(D_{PT})$ due to the inclusion of $Y(D_{AT})$. If this value is bigger than 0, it means that incorporating $Y(D_{AT})$ into the representation of $W(D_{PT)}$ in the form of data permeation and hybrid modality is beneficial. In this way,

$$\begin{aligned} &I(W(D_{PT}); Y(D_{AT})) = H(W(D_{PT})) - H(W(D_{PT})|Y(D_{AT})) \\ &= -\sum P(W(D_{PT})) \cdot log(P(W(D_{PT})))+ \\ &\sum P(W(D_{PT}), Y(D_{AT})) \cdot log(P(W(D_{PT})|Y(D_{AT}))), \end{aligned} \tag{2}$$

in which AT is a simple task, and $P(Y(D_{AT})) \sim 1$. Therefore, upon substitution, equation 2 can be expressed as:

$$\begin{aligned} &= -\sum P(W(D_{PT})) \cdot log(P(W(D_{PT})))+ \\ &\sum P(W(D_{PT})) \cdot log(P(W(D_{PT})|Y(D_{AT}))) \\ &= \sum P(W(D_{PT})) \cdot log \frac{P(W(D_{PT})|Y(T^{-1}(D_{AT})))}{P(Y(D_{AT})) \cdot P(W(D_{PT}))}, \end{aligned} \tag{3}$$

In equation (3), the numerator of the logarithm represents the probability of using the data space of PT to solve both PT and AT after combining the two networks. The denominator of the logarithm represents solving PT and AT separately using their respective networks. In PT, if the concatenated network can learn the patterns of AT, the performance will be better due to the increased network size. In this case, the expression inside the logarithm will have the numerator bigger than the denominator, so equation (3) will be bigger than 0. Conversely, if PT cannot learn the patterns represented by AT, the numerator will be smaller than the denominator in the expression inside the logarithm, resulting in equation (3) being less than 0. Only when equation (3) is greater than 0 will AT provide benefits to PT.

Table 1. An overview of four benchmark datasets

Dataset	Students	Concepts	Interactions
ASSISTments2009	4151	110	325,637
Statics2011	333	1,223	189,297
ASSISTments2015	19,840	100	683,801
ASSISTments2017	1,709	102	942,816

4 Experiments and analysis

4.1 Datesets

We evaluate our method on four widely used benchmark datasets. As shown in Table 1, these four datasets have three properties, namely number of students, number of knowledge concept, number of interactions by all students with all questions. We divided each dataset into 5 training sets, 5 verification sets and 5 test sets. The average number of sequences of these four datasets in the training set is 2500, 230, 11904, 1180 respectively. These values determine the hyperparameters of our KT network.

4.2 Implementation details

VAE network. Although the amount of data on each dataset is different, we use the same hyperparameters when training VAE network. For the encoder, the input/output dimensions of the two linear layers are (28×28, 100), (100, 20). For the decoder, input/output dimensions are symmetric, i.e. (20, 100), (100, 28×28). During the training, we use the Adam optimization method and set learning rate = 0.001. The balanced weights of $\mathcal{L}_{ELBO}$ are set to $\alpha = \beta = 1$.

KT network. Since the amount of data and the number of interactions between the four datasets differ greatly, we take this situation into consideration when setting the embedding dimensions and hidden dimensions. In all the experiments below, the embeddings of the questions on the four datasets are set to 256, 512, 30 and 256, respectively. The embeddings of the answers on the four datasets are set to 80, 60, 30 and 60, respectively. The default number of hidden dimensions are 160, 96, 80 and 160, respectively. We set the patience strategy value of training as 25 and set learning rate = 0.001. We set the dimension of all attention modules as 120. The values in the other modules can be deduced by the structure of the network. In $\mathcal{L}_{total}$, A is set to 1 and B is set to 0.012.

4.3 Comparison with the State-of-the-Arts

We compare our method with the state-of-the-art methods on the AUC scale. In Table 2, our method outperforms other methods by 5.17%, 5.70%, 3.75%, 9.28%, 6.02% and 2.83% in terms of average. Particularly, our method achieves the best performance on every dataset among all methods, and even improves 6.99% on ASSISTments2015 when compared with ATKT, the best method on AUC performance in previous works.

Table 2. The comparison of several methods on the performance of AUC.

Method	Assistments2009	Statics2011	Assistments2015	Assistments2017
BKT+ [5]	≈0.69	≈0.75	-	-
DKT [1]	0.8170±0.0043	0.8233±0.0039	0.7310±0.0018	0.7263±0.0054
AKT [4]	0.8169±0.0045	0.8265±0.0049	0.7828±0.0019	0.7282±0.0037
SAKT [3]	0.7520±0.0040	0.8029±0.0032	0.7212±0.0020	0.6569±0.0027
DKVMN [2]	0.8093±0.0044	0.8195±0.0041	0.7276±0.0017	0.7073±0.0044
MCB [17]	0.8059	0.8130	-	0.7141
ATKT [6]	0.8244±0.0032	0.8325±0.0043	0.8045±0.0097	0.7297±0.0051
MTFN	**0.8480±0.0022**	**0.8384±0.0042**	**0.8744±0.0047**	**0.7435±0.0048**

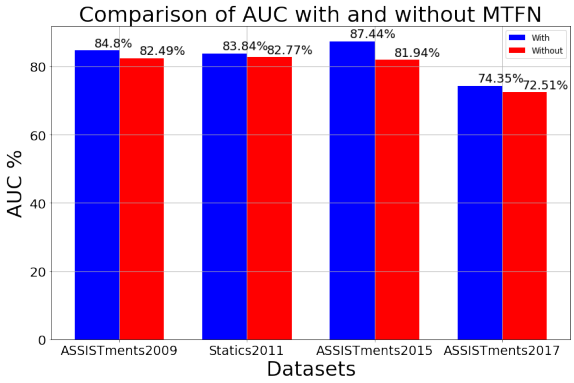

Fig. 3. Comparison of AUC with and without MTFN.

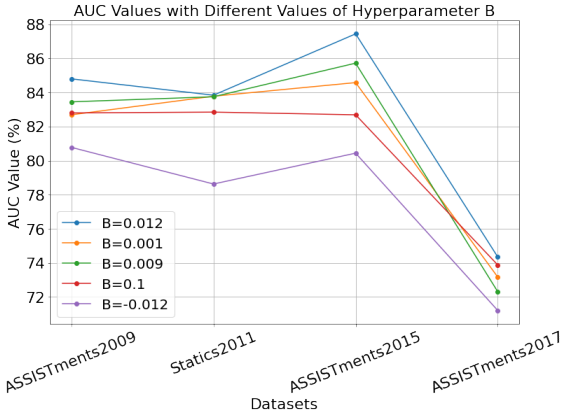

Fig. 4. AUC Values with Different Values of B.

4.4 Ablation study

To verify the effectiveness of MTFN, we removed the auxiliary task and conducted a comparison.As shown in Figure 3, without MTFN, the model's performance significantly declined. The performance decrease was most pronounced on the third dataset, which may be attributed to the larger amount of data on the third dataset and the lack of a well-represented network, resulting in a sharp drop in model performance.

However, the comparison experiment does not entirely demonstrate the effect of the auxiliary task on the representation learning of the primary task. We set A to 1 and the values of B were set to 0.001, 0.009, 0.012, -0.012, and 0.1, with particular emphasis on the negative value. We investigated the performance of the primary task under different values of B, and the specific data is presented in Figure 4. We found that when B was around 0.009, the accuracy of the auxiliary task oscillated between 0.5 (random guessing) and 1. When the value of B was less than 0.009, it became increasingly difficult for $\mathcal{L}_{total}$ to recognize the auxiliary task, rendering the task ineffective. In this scenario, the accuracy of the auxiliary task was close to 0.5. Consequently, the auxiliary task failed to provide effective representation patterns for the primary task. When B was relatively large (e.g., $B = 0.1$), the auxiliary task converged rapidly, with the accuracy approaching 1. Although the auxiliary task could provide decent representation features for the primary task, the network's inertia increased due to the convergence of the auxiliary task, making it less amenable to parameter adjustments. This resulted in suboptimal data representations provided by the auxiliary task. Through a series of experiments, we found that setting B to 0.012 resulted in relatively good performance for the primary task. When B took negative values, the representation of the auxiliary task became chaotic and disorderly and the accuracy of the auxiliary task was around 0.5. As depicted in Figure 4, the performance of the primary task was significantly affected by the chaotic representation of the auxiliary task, resulting in a considerable decrease in performance. These experiments illustrate that, within the MTFN framework, the auxiliary task serves as a supplement to the primary task modality and has a significant impact on the representation of the primary task. When this modality is in a chaotic state, the performance of the primary task will decline.

5 Conclusion

In this paper, in order to solve the problem of poor representativeness of the binary labels of the answer status data in the KT data set, we propose the MTFN network. In MTFN, the data from the primary task (KT) are represented from the auxiliary task (handwriting numerals recognition). Through data permeation and cross-training with mixed modality, the performance of KT can be enhanced. Experimental results demonstrate the improvement of KT by 2.8% and results from ablation experiments indicate that optimizing the data representation of the primary task only occurs when it can learn the data patterns from the auxiliary task.

References

1. 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, 28.
2. Zhang, J., Shi, X., King, I. and Yeung, D.Y., 2017, April. Dynamic key-value memory networks for knowledge tracing. In Proceedings of the 26th international conference on World Wide Web (pp. 765-774).
3. Pandey, S. and Karypis, G., 2019. A self-attentive model for knowledge tracing. arXiv preprint arXiv:1907.06837.
4. Ghosh, A., Heffernan, N. and Lan, A.S., 2020, August. Context-aware attentive knowledge tracing. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 2330-2339).
5. Yudelson, M.V., Koedinger, K.R. and Gordon, G.J., 2013. Individualized bayesian knowledge tracing models. In Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings 16 (pp. 171-180). Springer Berlin Heidelberg.
6. Guo, X., Huang, Z., Gao, J., Shang, M., Shu, M. and Sun, J., 2021, October. Enhancing knowledge tracing via adversarial training. In Proceedings of the 29th ACM International Conference on Multimedia (pp. 367-375).
7. Vaswani, A., 2017. Attention is all you need. Advances in Neural Information Processing Systems.
8. Khajah, M., Lindsey, R.V. and Mozer, M.C., 2016. How deep is knowledge tracing?. arXiv preprint arXiv:1604.02416.
9. Pandey, S. and Srivastava, J., 2020, October. RKT: relation-aware self-attention for knowledge tracing. In Proceedings of the 29th ACM international conference on information & knowledge management (pp. 1205-1214).
10. Pardos, Z., Bergner, Y., Seaton, D. and Pritchard, D., 2013, July. Adapting bayesian knowledge tracing to a massive open online course in edx. In Educational Data Mining 2013.
11. Desmarais, M.C. and Baker, R.S.D., 2012. A review of recent advances in learner and skill modeling in intelligent learning environments. User Modeling and User-Adapted Interaction, 22, pp.9-38.
12. Kasurinen, J. and Nikula, U., 2009. Estimating programming knowledge with Bayesian knowledge tracing. ACM SIGCSE Bulletin, 41(3), pp.313-317.
13. Kingma, D.P., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
14. Rezende, D. and Mohamed, S., 2015, June. Variational inference with normalizing flows. In International conference on machine learning (pp. 1530-1538). PMLR.
15. Abdelrahman, G., Wang, Q. and Nunes, B., 2023. Knowledge tracing: A survey. ACM Computing Surveys, 55(11), pp.1-37.
16. Cui, C., Yao, Y., Zhang, C., Ma, H., Ma, Y., Ren, Z., Zhang, C. and Ko, J., 2024. DGEKT: a dual graph ensemble learning method for knowledge tracing. ACM Transactions on Information Systems, 42(3), pp.1-24.
17. Lee, U., Park, Y., Kim, Y., Choi, S. and Kim, H., 2024, June. Monacobert: Monotonic attention based convbert for knowledge tracing. In International Conference on Intelligent Tutoring Systems (pp. 107-123). Cham: Springer Nature Switzerland.