Reinforcement Learning-Based Item Selection for Adaptive Testing and Mastery Testing

Chengzhi Wei, Wenyi Wang
* UCLA, Los Angeles, CA 90095

***Abstract*—Item Response Theory (IRT) is widely used in education measurement to get student ability, item difficulty and Knowledge Component (KC) distribution. These parameters can be utilized in student performance assessment and future item selection. There are two main purposes of item selection. First is to choose the most suitable item in the pool to get the most information about a student's ability, which is call Computational Adaptive Testing (CAT), Second method tries to select a sequence of items each testing one KC and hope to know a student mastery of all the KCs, we call it Mastering Testing (MT). In this project, we evaluate whether reinforcement learning facilitate the process of item selection. We first introduce a gradient based CAT method which is much faster and can optimize for both student and item compared with traditional Fisher information based method. However, the sampling error makes it hard to converge. In the meanwhile, we find that there is lack of research in the area of MT. We then propose a reinforcement learning based MT method to minimize the number of items we need to get a whole picture of a student mastery. Two different frameworks are analyzed and at last we are able to check the mastery of 100 KCs with just 20 items where a random method usually needs more than 70 items.**

***Index Terms*—Optimal Item Selection for Adaptive Testing and Mastery Testing**

I. INTRODUCTION

Item Response Theory (IRT) [1], [2], [3] is the most ubiquitous model to analyze student and item's latent trait. Since this gives us a quantitative description of the item difficulty and student ability, automatic item selection is made possible in modern online test where computer decides the next best problem giving to the student [4], [5]. There are mainly two different criteria in deciding which item is the best. In Computational Adaptive Testing (CAT) [6], [7], [8], [9], the item in pool which can maximize the Fisher Information of student ability is chosen. This is equivalent to minimize the variance of the posterior distribution of student ability. In Mastery Testing (MT), given a graph of the dependence of Knowledge Components (directed graph and an edge means one KC is the pre-requisite of another), the minimum number of items should be selected in sequence to test the student's mastery of all the KCs.

CAT is widely researched and various method is proposed. The main idea is to choose the item which maximizes the Fisher Information of student ability. In [9], Monte Carlo Markov Chain (MCMC) is used to estimate the Fisher information given each item. However, sampling for each item in the pool makes it time consuming. In [10], a modified MCMC method is proposed so that it works efficiently for polynomial number of items. Besides scores, response time is used to assist the item selection in [11]. However, all of them need to search through all items in the pool and find the best match one. In practice, we often have a huge item pool and it is not efficient to do that. In our job, we build a target loss function which we can take gradient over item parameters so that we can use gradient descent to find the best suitable parameters. We then find the item which has the minimum KL divergence with the best parameters. In this method, we don't need to go through all the items in the pool and sample distributions. Also, our loss function can include both ability variance and item parameters' variance so that it can both improve our estimation of ability and optimize our item pool. This enables us to optimize our database simultaneously with estimating student parameters while traditional method tends to use high frequency problem again and again.

Reinforcement Learning gradually becomes popular in Intelligent Tutoring System (ITS). Partially Observable Markov Decision Process is widely applied in Intelligent Tutoring System (ITS) [12], [13]. And in [12], the dependence graph of KCs is utilized to help decide which next instruction should be given to a student. However, in the area of MT, there is little work involving RL [14]. Here, we propose a DQN based framework which maps a current student KC graph to next item. Different features and different networks are compared and Breadth First Embedding with Chebyshev Convolution layers gives us the best solution.

Our contributions are listed as follows.

1) Research on the possibility of applying RL to item selection and analyze why it fails or succeed.
2) Propose a gradient based CAT method which does not need search through all the possible items. It can improve our estimating of student and optimize our item parameter estimation simultaneously to avoid high frequency bias.
3) Propose a DQN based MT method and compares different features and networks. The final Breadth First Embedding with Chebyshev Convolution layers DQN converges fast and works well in different input size.

The following sections are organized as follows. In section II, related backgrounds information about IRT, CAT and MT are illustrated. In section III, gradient based CAT is described and why it fails is analyzed. In section IV, RL based MT is introduced and results are shown in section V. Conclusion are made in section VI.

II. Background

Here we will introduce some terminologies we will use later.

A. *Item Response Theory*

IRT is widely used in assessing students' abilities and item difficulties due to its efficiency and readily comprehensible. Given a student i and an item j, we write the result of student i doing item j as $y_{i,j} \in 0,1$. Here, 0 means doing it incorrectly and 1 means doing it correctly. IRT models the probability of correctly given different parameters. The most basic is the one-parameter logistic (1PL) model [15] which is shown in Eq.1,

$$p(y_{ij}|\theta_i, b_j) = \frac{1}{1+e^{-(\theta_i - b_j)}} \tag{1}$$

where θ_i denotes the ability of the i-th student and b_j denotes the difficulty of the j-th item. Some more sophisticated model including 2PL model [16], 3PL model [17] and Multi-dimensional Model.

Given a collection of student test results, both Maximum Likelihood Estimation (MLE) and Maximize A Posterior (MAP) can be used to get all the parameters. In MLE, the likelihood can be written as $L = \prod_i \prod_j p(y_{ij}|\theta_i, b_j)$ and many optimization tools can be used to estimate it. In MAP, thing becomes more complex. The posterior of parameters can be written as $p(\theta_i, b_j|y_{ij}) \propto \prod_i \prod_j p(y_{ij}|\theta_i, b_j) \prod_i p(\theta_i) \prod_j p(b_j)$. The prior distribution is normal distribution but the likelihood is the product of logistic functions. To get a normal posterior distribution, we need to find the KL divergence between normal distribution and logistic function. Many methods has been proposed to solve that including Expectation Maximization [18], MCMC [19], Variational Inference [20].

B. *Computerized Adaptive Testing*

Suppose given a set of data D, we already have a posterior of student ability $P(\theta_i|D) = N(u_i, \sigma_i^2)$ and item difficulty $P(b_j|D) = N(u_j, \sigma_j^2)$. If problem j is selected problem j is given to student i, then the parameters change to Eq.2,

$$p(u_i, u_j, \sigma_i, \sigma_j|D, y_{i,j}) \propto \frac{1}{1+e^{-(u_i - u_j)}} * N(u_i, \sigma_i^2) * N(u_j, \sigma_j^2) \tag{2}$$

We assume all parameters have the normal distribution, so $p(u_i, \sigma_i|D, y_{i,j}) \sim N(\overrightarrow{u_i}, \overrightarrow{\sigma_i})$ and $P(u_j, \sigma_j|D, y_{i,j}) \sim N(\overrightarrow{u_j}, \overrightarrow{\sigma_j})$. Tradition CAT works on finding the item $j*$ shown in Eq.3,

$$j* = argmax_{j \in pool}\ Fisher(\theta_i)\ or\ argmin_{j \in pool} \overrightarrow{\sigma_i} \tag{3}$$

Normally, MCMC is used to sample the Fisher information for each item.

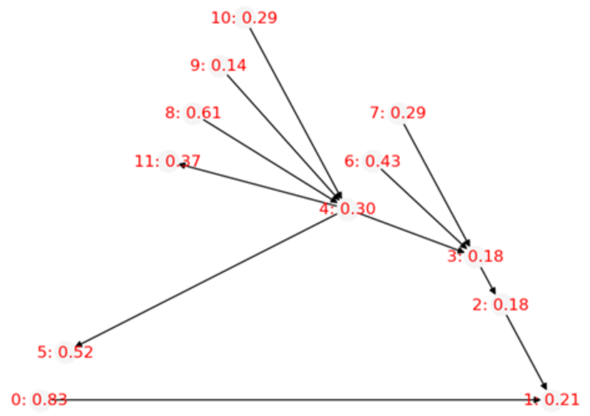


Fig. 1: An example of KC graph, the text in each node is: (KC id: average accuracy of answering this KC)

C. *Mastery Testing*

Mastery Testing aims to generate a sequence of items to test whether a student masters all the KCs. We assume KC graph is obtained through professional analyze and denote it as $G = (N, E)$. G is a directed graph without loop while each node is a KC and each edge from i to j means KC i is a pre-requisite of KC j. At each time step, one item about one KC is selected and given to a student. If it is solved correctly, all the pre-requisites KCs are tested recursively and reversely if it solved wrongly, all the following KCs are tested. Notice that the accuracy of answering each problem also influences the selection strategy and it cannot be directly observed. In Fig.1, an example of KC graph is shown. If a student answer KC 4 incorrectly, we know that KC 4, 11, 5, 3, 2, 1 are not mastered.

In our scenario, given a KC graph without probability, an agent should be able to suggest the next item given to the student that can minimize the total number of items used.

III. Gradient Descent Based CAT

As we proposed in section II.B, MCMC is used to estimate the Fisher information of each item. This means we need to sample hundreds of points for each item in a pool. Here, we propose another approach shown in Eq.4 and Eq.5,

$$u*_j, \sigma*_j = argmin_{u_j, \sigma_j} \overrightarrow{\sigma_i} + \overrightarrow{\sigma_j} \tag{4}$$

$$j* = argmin_{j \in pool} KL(N(u*_j, \sigma*_j), N(u_j, \sigma_j)) \tag{5}$$

Eq.5 can be easily solved since there is arithmetic formula for KL divergence between two normal distributions. To solve Eq.4, gradient descent is used so that we need to calculate $\Delta_{u_j, \sigma_j} \overrightarrow{\sigma_i} + \overrightarrow{\sigma_j}$. However, though MCMC can generate a sample of $\overrightarrow{\sigma_i}$ and $\overrightarrow{\sigma_j}$, we cannot calculate its gradient because a point depends on the previous points. To deal with it, rejection sampling and reparametrization tracks are used. The full procedures are illustrated as follows.

Algorithm 1 Gradient Descent Based CAT

```
1. random initialize $u_i, u_j, \sigma_i, \sigma_j$.
2. calculate $p = \frac{1}{1+e^{-u_i+u_j}}$
3. generate N points $r_i$ and N points $r_j$ from$N(0,1)$.
4. calculate $\theta = u_i + r_i * \sigma_i$ and $b = u_j + r_j * \sigma_j$
5. generate N random points $r_1$ and $r_2$ from $U(0,1)$.
6. reject $\theta, b$ when $\frac{1}{1+e^{-\theta+b}} > r1$ and get their variances $\sigma_{pi}, \sigma_{pj}$
7. reject $\theta, b$ when $\frac{e^{-\theta+b}}{1+e^{-\theta+b}} > r2$ and get their variances $\sigma_{ni}, \sigma_{nj}$
8. get $loss = p * (\sigma_{pi} + \sigma_{pj}) + (1-p) * (\sigma_{ni} + \sigma_{nj})$
9. do gradient descent and return to step 2.
```

However, this method suffers from two problems. First, we only have one more test score and the new posterior does not differ much from the prior. The sampling randomness often overwhelms the real trend and we need a larger N to converge. Second, such a loss function leads to consistent convergence to $\sigma*_j = 0$. If we use the decrease in variance as loss, it will consistently converge to $\sigma*_j = \infty$. This method is highly efficient however, we fail to come up with an efficient loss function that can both optimize the student variance and item variance.

IV. DQN Based MT

A. DQN

Deep Q-learning (DQN) [21] is the most famous reinforcement learning framework. It uses a global approximator to approximate the state-action function and then uses it to make decision. To deal with the unstable convergence of DQN, Double DQN [22] have been proposed. It uses two DQNs and iteratively update their parameters.

In this project, DQN is trained to do Mastery Testing. The state is a KC graph structure without hidden probability. Potential actions are all the KCs. If a KC is chosen, we will give an item on that KC. Then a function approximator is used to mapping a given state to Q values of all actions. Based on that, ϵ-greedy will be used to select an action. Removing a selected KC from the graph will returns an reward(-5 if remove an none-existing KC, -1 if remove a existing KC and 10 if remove the last KC of the graph) and give us the next KC graph. Thus, our function approximator should be able to learn the hidden probability and make the optimal decision to use the least number question to cover all KCs.

Two different function approximators will be discussed in the following sections.

B. CNN Based Method

Convolutional neural network[23] is class of artificial neural network which is based on the shared-weight architecture of the convolution kernels that slide along input features and provide translation equivariant responses.

The first thing is to build an input based on the KC graph. A three layer NxN matrix is introduced as the selected feature for CNN network. The first layer is a Laplacian matrix calculated by $L = A - D$ where A is the adjacent matrix and D is the degree matrix represent the degree of each node. This layer provide detailed information between each KCs but the network can not discriminate independent node (KC) and node (KC) we already remove. To solve this problem, we introduce the second layer, a matrix with $A_{ii} = 1$ for each existing KC i to let the network learn the difference between independent nodes and removed nodes. The third layer is our self-designed matrix. For each node, we find the number of out going nodes it can reach in m steps where $m = N/2$ as well as number of in coming nodes it can reach in m steps using dfs. So for each node we have a $2 * m$ which is N length row. We combine the row for each node to get the third N x N matrix. We hope this matrix will provide more information about the influence of each node(KC) to the whole graph when it's been removed. This will help to facilitate the network learning process.

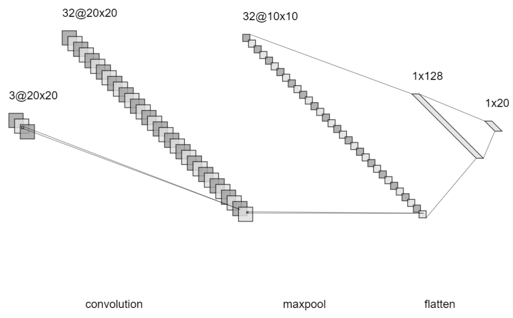


Fig. 2: Structure of CNN network for 20 nodes graph

A traditional CNN with 3*N*N feature matrix as input is used as the function approximator. An example Structure of CNN for 20 Nodes KC graph is shown in fig.2. The first layer is a 2D convolution layer of kernel size 3x3, output channel 32, padding 1 and stride 1 with Relu as activation function. Following by a maxpooling layer of size 2x2 and stride 2. Then we flatten it and import into a fully connected layer of size 3200x128 and another fully connected layer of size 128x20 and the output will represent which action should be chosen.

C. GCN Based Method

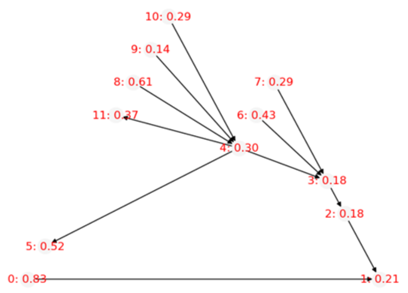
Graph Convolution Network is a type of Graph Neural Network used for processing data represented by graph data structures. In this project, we use the embedded feature matrix (third layer of CNN input) as the feature input. Then two Chebyshev spectral graph convolutional layers are used. The first one have output dimension 3*N and the second one is 5*n. Next, two fully connected layers with output dimensions 2*N and 1 are used. Graph neural network utilizes the graph structure information and can perform better learning in graph problem. Also, Chebyshev layer is used because it can keep the zeros in the input features so that the removal information is transmitted through layers. With the help of graph structure and the better information flow, GCN is supposed to work better than normal CNN.

V. Results

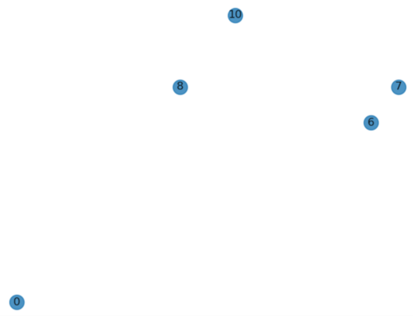
A. CNN Based Method

CNN Based method can learn the Mastery testing process well when the graph size is small. A complete KC selection process for 12 KCs graph is shown in Fig.3. The original KC graph with hidden probability is shown in Fig.3 (a). DQN with CNN function approximator first choose KC 9 as first action. From the probability graph, we can see that the probability of student to answer it correctly is 0.14 which can split it into two scenarios. If his answer is wrong, that will remove all the KCs depend on KC 9 and lead to Fig.3 (b) and have five independent KCs left which needs five more steps and return 6 steps as total steps. In second scenario, if student answer question correctly with probability 0.14, it will lead to Fig.3 (c). The next step it choose is to select KC 3 instead of KC 4, which obviously depended by more existing KCs. This move first show that CNN based DQN is able to learn the hidden probability well as we can see from Fig.3, KC 4 is less likely to be wrong and remove depended nodes. Also, this action shows the advantage of using DQN over greedy algorithm as it give up higher potential current reward in trade of higher future reward by reducing future independent KCs.

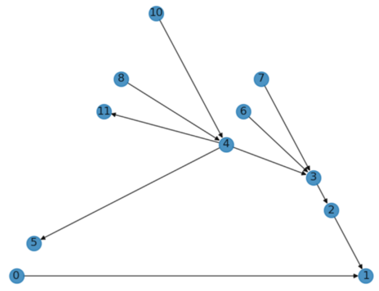
To better understand the influence of each feature layers on CNN function approximation, we test on three group of inputs. First input is the three layer feature defined before. Second input is a three layers N*N matrix contain diagonal matrix and Laplacian matrix. The third one is a single layer matrix of Laplacian matrix only. The training detail is shown in fig.4. From graph, we can see the input with Laplacian only does not converge which follows our expectation as it can not discriminate independent KCs and KCs already removed. So the network can not converge as it keep selecting removed KCs. The input with diagonal and Laplacian matrix contains enough information for CNN to learn the features as it converge. But adding the self defined layer can increase the convergence speed as it provide more information.



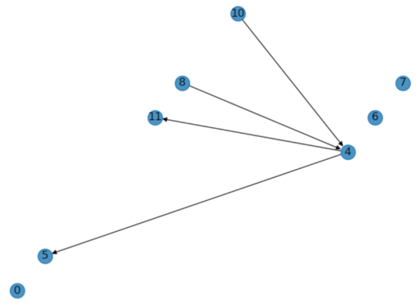
(a) KC graph of node 12 with inner probability



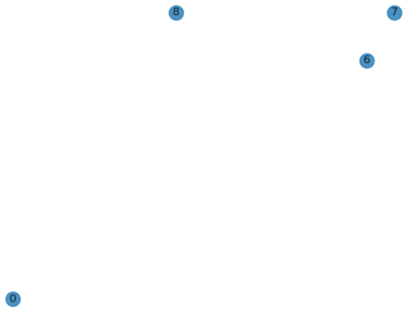
(b) KC graph of node 12 when node 9 removed and incorrect



(c) KC graph of node 12 when node 9 removed and correct



(d) KC graph of node 12 when node 3 removed



(e) KC graph of node 12 when node 10 removed

Fig. 3: KC graph of node 12 with CNN network

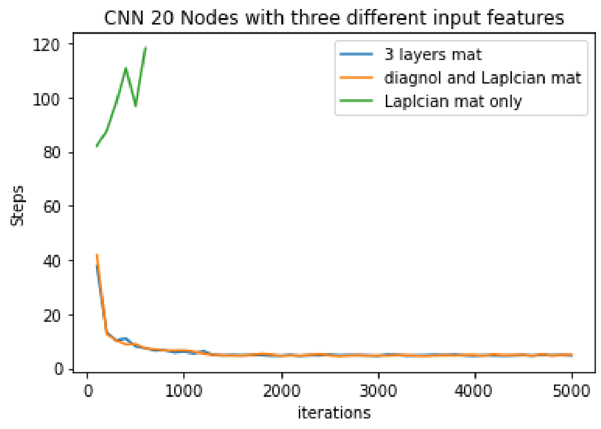


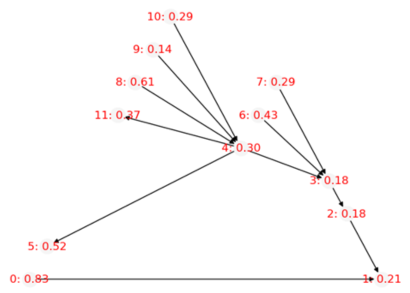
Fig. 4: CNN 20 Nodes 3 inputs Comparison

B. *GCN Based Method*

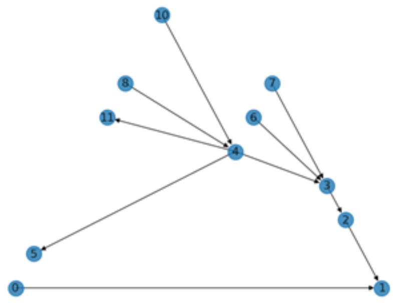
The GCN Based Method can learn the Mastery testing process well without changing network structure for all size of KC graph(KC graph with different number of nodes). Use a 12 nodes KC Graph as an example, in the first 100 iteration it needs about 10 steps(10 questions) to cover all KCs but it converge to 6.1 steps in 5000 iterations. We use this trained network to predict a complete KC removal process of a 12 KCs graph in order to make better evaluations. The original KC graph is shown in fig.5 (a). The system's first move is to remove KC 9, but student has only 14 percent chance to do this question correct. So if student do this wrong, all the KCs depend it like KC 4 will be removed as if student can't do KC 9 correctly, they definitely don't understand knowledge about KC 4. The left over Nodes are independent so we need five more steps to complete the process and give us 6 steps as total step.

In another scenario, student do the question about KC 9 correctly with probability 0.13, then we can not remove all the KCs depend on it and lead to the graph shown in Fig.5 (b). The next step decided by the network is to test on KC 6. In the presenting case, student do this question wrong and remove KCs 1,2,3 as well and gave us graph in Fig.5 (c). Then it choose to test on KC 8 but the student do the question correctly and lead us to graph in Fig.5 (d). By testing on KC 7 and KC 4, the KC graph only have two independent KCs left(shown in Fig.5 (e)) and need two more steps to complete. In this scenario, it need 7 steps.

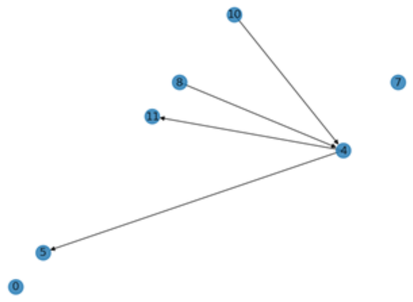
When we look deeper at the correctness probability in Fig.5 (a), we can see test question on node 7 is a better option for second step. It has larger probability to fail and remove same number of nodes if does failed. However, probability information are not given to the network and it has to learn it from each iterations' random probability. The network choose the sub-optimal action shows that our network do learn the probability but may not be able to learn an extremely precise one.



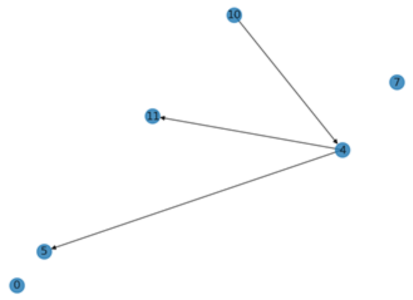
(a) KC graph of node 12 with inner probability



(b) KC graph of node 12 when node 9 removed and correct



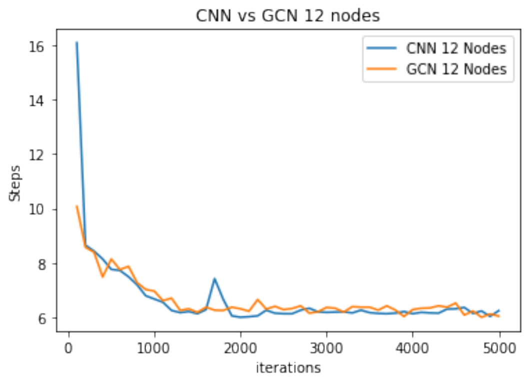
(c) KC graph of node 12 when node 6 removed



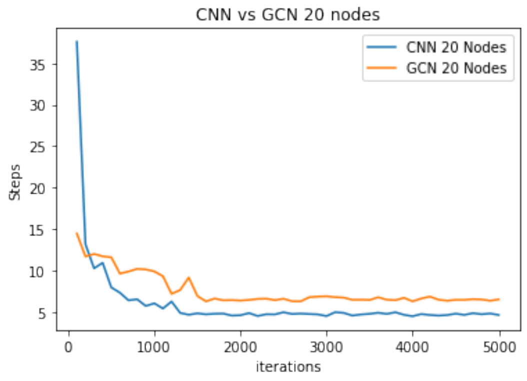
(d) KC graph of node 12 when node 4 removed

(e) KC graph of node 12 when node 4 removed

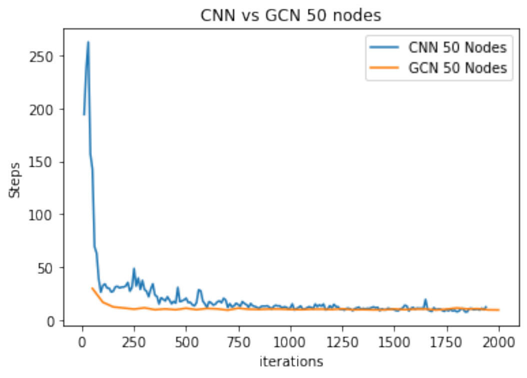
Fig. 5: KC graph of node 12 with GCN network



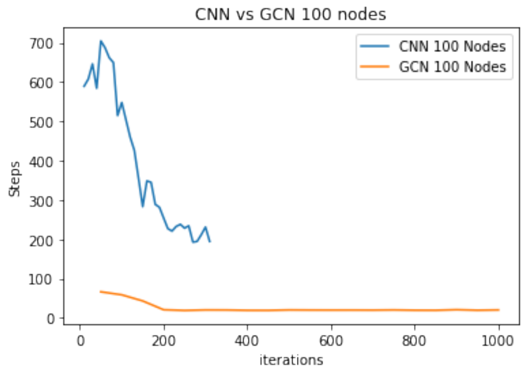
(a) CNN vs GCN 12 KCs



(b) CNN vs GCN 20 KCs



(c) CNN vs GCN 50 KCs



(d) CNN vs GCN 100 KCs

Fig. 6: Comparison of training performance between CNN and GCN

C. *Comparison*

The comparison of training performance between two function approximator (CNN and GCN) for different size KC graph is shown in Fig.6. From Fig.6 we can see that within the first 100 iterations, GCN always converge faster and need less steps than CNN. This phenomenon is due to the advantage of GCN's structure which can distinguish removed KCs and independent KCs. Thus, GCN only needs at most size of KC graph steps to complete a remove. CNN needs to learn which KC can be removed (valid actions) from the feature matrix, so it will keep selecting invalid action in the beginning and took more steps at first.

From Fig.6 (a) and (b), we find that CNN converge to lower steps than GCN. The reason for that is CNN have more parameters and perform better in predicting the hidden probability of each node.

Starting from 50 nodes, the CNN's structure needs to be changed by adding more convolutional layers and increase the fully connected(fc) layer size in order to learn the larger size KC graph. Yet, GCN do not need to change structure. From Fig.6 (c), we can see though CNN can still learn the 50 KCs Mastering testing by changing CNN structure, it's convergence speed is much slower than GCN. And when KCs increased to 100, CNN is not able to converge to a reasonable steps which should be lower than KC graph size.

VI. Conclusion

In conclusion, RL based method can greatly facilitate the process of Item Selection. In cat, our proposed gradient based method is efficient but we didn't come up with a helpful loss. In MT, DQN with GCN and feature selection works really well.

References

[1] Hambleton, R. K. (2018). Handbook of item response theory. W. J. Van der Linden (Ed.). Chapman & Hall/CRC.
[2] Reckase, M. D. (2009). Multidimensional item response theory models. In Multidimensional item response theory (pp. 79-112). Springer, New York, NY.
[3] Orlando, M., & Thissen, D. (2000). Likelihood-based item-fit indices for dichotomous item response theory models. Applied psychological measurement, 24(1), 50-64.
[4] Raubenheimer, J. (2004). An item selection procedure to maximize scale reliability and validity. SA Journal of Industrial Psychology, 30(4), 59-64.
[5] Van der Linden, W. J., & Pashley, P. J. (2009). Item selection and ability estimation in adaptive testing. In Elements of adaptive testing (pp. 3-30). Springer, New York, NY.
[6] van der Linden, W. J. (1998). Bayesian item selection criteria for adaptive testing. Psychometrika, 63(2), 201-216.
[7] Huang, Y. M., Lin, Y. T., & Cheng, S. C. (2009). An adaptive testing system for supporting versatile educational assessment. Computers & Education, 52(1), 53-67.
[8] Mulder, J., & Van der Linden, W. J. (2009). Multidimensional adaptive testing with Kullback–Leibler information item selection. In Elements of adaptive testing (pp. 77-101). Springer, New York, NY.
[9] van der Linden, W. J., & Ren, H. (2020). A fast and simple algorithm for Bayesian adaptive testing. Journal of educational and behavioral statistics, 45(1), 58-85.
[10] Ren, H., Choi, S. W., & van der Linden, W. J. (2020). Bayesian adaptive testing with polytomous items. Behaviormetrika, 47, 427-449.
[11] Choe, E. M., Kern, J. L., & Chang, H. H. (2018). Optimizing the use of response times for item selection in computerized adaptive testing. Journal of Educational and Behavioral Statistics, 43(2), 135-158.

[12] Wang, F. (2014, April). POMDP Framework for Building an Intelligent Tutoring System. In CSEDU (1) (pp. 233-240).
[13] Ramachandran, A., Sebo, S. S., & Scassellati, B. (2019, July). Personalized robot tutoring using the assistive tutor pOMDP (AT-POMDP). In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 8050-8057).
[14] Falmagne, J. C., Albert, D., Doble, C., Eppstein, D., & Hu, X. (Eds.). (2013). Knowledge spaces: Applications in education. Springer Science & Business Media.
[15] Rasch, G. (1960). Studies in mathematical psychology: I. Probabilistic models for some intelligence and attainment tests.
[16] Lord, F. (1952). A theory of test scores. Psychometric monographs.
[17] Hambleton, R. K., & Cook, L. L. (1977). Latent trait models and their use in the analysis of educational test data. Journal of educational measurement, 75-96.
[18] Fox, J. P. (2000). Stochastic EM for Estimating the Parameters of a Multilevel IRT Model. Research Report.
[19] Beguin, A., & Glas, C. A. (1998). MCMC estimation of multidimensional IRT models.
[20] Wu, M., Davis, R. L., Domingue, B. W., Piech, C., & Goodman, N. (2020). Variational item response theory: Fast, accurate, and expressive. arXiv preprint arXiv:2002.00276.
[21] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.
[22] Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 30, No. 1).
[23] LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep Learning. Nature, 521, 436–444. doi: 10.1038/nature14539