Incorporating rich features into Deep knowledge tracing

1st Author Name
Affiliation
City, Country
e-mail address

2nd Author Name
Affiliation
City, Country
e-mail address

3rd Author Name
Affiliation
City, Country
e-mail address

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 Performance Factors Analysis (PFA), Bayesian Knowledge Tracing (BKT), 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.

Keywords
Knowledge Tracing, Deep Learning, Deep Knowledge Tracing (DKT), Recurrent Neural Networks (RNN), Auto Encoders.

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) [15] and performance factor analysis (PFA) [11] 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, for example, 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. The concept to treat each skill individually is perhaps a leading factor in the success of these models, as they understand that students will exhibit different learning behaviors depending on content.

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[18]. 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) [1]. 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 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 by Piech et al. [2], 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, $\{(x_{st1,1,1}, y_{st1,1}), (x_{st1,1,2}, y_{st1,2}) \ \dots \ (x_{st1,T}, y_{st1,T})\}$, while the output layer is $\{y_{st1,2}, y_{st1,3} \ \dots y_{st1,T+1}\}$. The term $x_{st1,1}$ refers to the feature combination of question (or skill) and correctness of student1 on a problem of skill 1. $y_{st1,1}$ refers to the correctness of a problem from skill 1 for student 1. 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 recurrent neural network 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. A popular variant of RNN, also used in the DKT model, is that of long short-term memory (LSTM) networks. The key difference of LSTMs to traditional RNNs is the internal node structure [14], that acts like a conveyor belt in determining how to modify information within each recurrent node. The LSTM variant uses three gates to remove or add information to the cell states, determining how much information to remember from previous timesteps and also how to combine that memory with information from the current timestep.

The recurrent hidden nodes are trained to identify and retain the relevant aspects of the input history as it pertains to student performance. The appearance of DKT drew attention by the educational data mining community due to the claimed dramatic improvement over BKT, claiming about 25% gain in predictive performance using the ASSISTments 2009 benchmark dataset. At the 2016 Educational Data Mining Conference, three papers [9, 13, and 17] were published to compare DKT with traditional probabilistic and statistical models. They argue that traditional models and variants still perform as well as this new method with better interpretability and explanatory power.

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 DKT model, but the network can easily consider more features. In this paper, we explore the inclusion of more features to improve the accuracy of prediction. However, the incorporation of new features can quickly increase the input layer dimensionality, requiring careful consideration to avoid model over fitting and also to ensure the feasibility of training such a model within reasonable hardware requirements.

While a simple feed-forward neural network can be trained relatively quickly depending on the number of nodes and size of the dataset, RNNs are considerably more computationally expensive due to the comparatively larger number of parameters. In such models fitting procedures often take hours or days to run on large data sets. For example, training a LSTM DKT model with 50 skills and 200 hidden nodes needs to learn 250,850 parameters. In our environment, the training of DKT models on the ASSISTments 2009 benchmark dataset takes 3.5 minutes per epoch, equating to more than 14 hours when using a 5 fold cross validation run over 50 epochs. In contrast, BKT is able to train on the same dataset within 10 minutes.

In this way, training time and the number of parameters are considered as an important metric of comparison; such models need to provide significant gains to predictive performance to justify their usage over simpler models. To this extent, the network structure of DKT may benefit from reduced dimensionality, particularly if this can be achieved without sacrificing performance. An auto-encoder [7] is one such approach to this problem. Auto-encoders are multilayer neural networks with a small central layer that can convert 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 important information. This technique is an unsupervised learning algorithm that applies backpropagation, much like a traditional feed-forward neural network, observing the input vector as the training output. Using a smaller number of nodes in the hidden layer, therefore, finds a smaller number of values that can reconstruct the input. Once trained, the output layer can be removed, and the hidden layer can connect to another network layer. Auto-encoders 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.

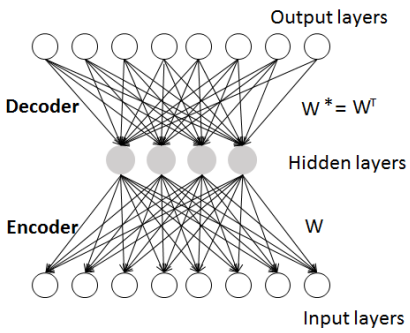

Figure 1. One layer auto-encoder neural network; the weights of the decoder is simply the matrix transpose of the encoders. The hidden layer becomes a dense feature vector representative of the input layer.

3. IMPROVING DKT WITH MORE FEATURES

Intelligent tutoring systems often collect additional features about the interaction of students including information on problems, instructional aids, and time spent on individual tasks. Models and algorithms that make use of this additional information have been proposed. For example, hint usage and the number of attempts need to find the problem answer are adopted to predict the performance in the sequence of actions (SOA) model [8]; partial credit history acquired based on the number of hints used and the number of attempts are used to predict the probability of that students getting the next question correct [7].

As previously described, it is easy to incorporate useful information such as this into the input layer of a neural network. However, the key consideration is how feature engineering is performed on these features. Feature engineering played a vital role for the NTU team [6] who won the KDD competition in 2010. They incorporated a large number of features and cross-features into a vector-space model and then trained a traditional classifier. They also identified some useful feature combinations to improve the performance. Cross features were used in the original DKT work as well, utilizing a one-hot encoding to represent an correct and incorrect response for each skill separately as a vector of 2 times the number of skills; alternatively, such information could be represented separately, with a one-hot encoding representing skills, and just one binary metric to indicate correctness equating to a vector of the number of skills plus 1. In wide-and-deep learning proposed by Google [16], sparse features and cross features are selected for wide part, while the continuous columns and the embedding dimension for each categorical column are selected for deep part. These exemplary models use the engineering of features to improve model accuracy helping to motivate the methodology of this work.

3.1 Feature process

In order to train the RNN model on student-tutor interaction data, the information must be converted into a sequence of fixed-length input vectors. Several features are selected for our modeling experiment they are exercise (skill) tag, correctness, time (the time in seconds before the student's first response), hint usage (total number of hints requested by the student), attempt count (the number of attempts made to answer correctly on this problem), and problem view (total number of times the student encountered the problem so far). The exercise tag feature is used to identify the content of a problem, acting as the skill-level tag. In different data sets, the skill level tag can exhibit differing representations, described by either a numeric skill id or the name of the knowledge component.

Numerical features like time, hint usage, attempt count and problem views can be bucketed into categorical features which can be used to construct cross features in order to reduce the complexity of the model. This process simplifies the input without losing much information, as small difference in numeric values is often less important than large differences. For example, if a student finishes exercise *a* within 10 seconds while the other student is 300 seconds in the same exercise, the time difference represents their different mastery in exercise. Meanwhile, comparing a student who finishes in 10 seconds to a student who finishes in 11 seconds demonstrates similar, if not arguably the same level of understanding. Bucketing still captures this information while significantly reducing model complexity. The numeric features in this paper are bucketed across all skills and the result is represented by a one-hot encoding.

Cross features such as the tuple of exercise and correctness, are represented as one integer represented by a one-hot encoder. The advantage of using cross features has been shown to improve model performance [6] while models representing features separately exhibit degraded performance [2]. However, the disadvantage of using cross features is the rapid increase of the dimensionality of the input vector. As the dimensionality increases, it is hard for the model to converge to the global optimal. At the same time, computational resources may become exhausted due to the large number of parameters. Dimensionality reduction, and the extraction of key features, is critical to guarantee the running of such models. Here, an auto-encoder is used to accomplish this task. In our experiment, the dimension is successfully reduced to a quarter of the

input size. We train the initial weights using an auto-encoder, and hold them constant while training the remainder of the model.

3.2 Model

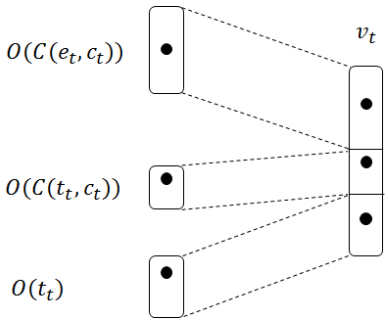

Figure 2 Feature concatenation

The input vector of our model 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. The term e_t refers to the exercise tag, while c_t refers to correctness, and t_t represents time before the first response. Concatenation is described in the formulas below.

$$v_t = O(C(e_t, c_t)) +' O(C(e_t, c_t)) +' O(t_t) \quad (1)$$

$$C(e_t, c_t) = e_t + (\max(e) + 1) * c_t \quad (2)$$

In these, $O()$ is the one-hot encoder format, $C(\ )$ is the cross feature, and the $+'$ operator is used to denote concatenation, not addition in (1). In (2), 1 is added in the expression due to the unincluded exercise.

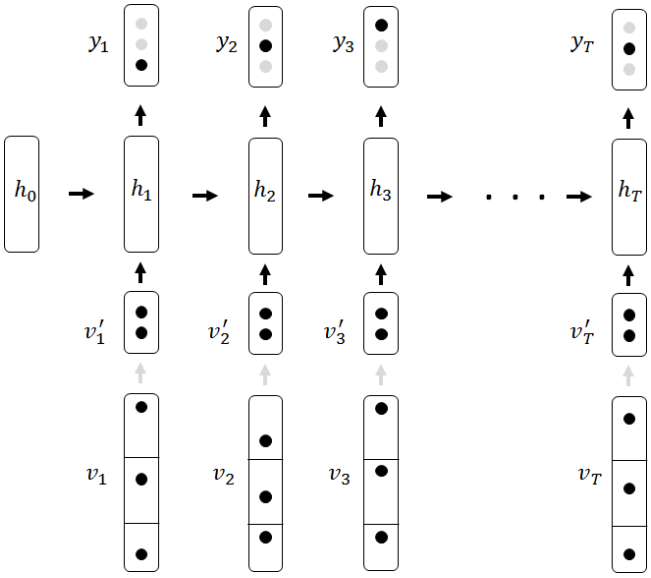

Figure 3 A representation of the Deep Knowledge Tracing model with more features

Figure 3 depicts the resulting model representation utilizing an auto-encoder layer to support the added features. In figure 3, v_1' represents the feature vector extracted from v_1 by auto-encoder; after training, this is simply the output of the hidden layer for each input vector. The gray arrows mean that weights between the two layers are held constant, so the auto-encoder is trained separately in advance. From our experiments, we noticed that the fine tuning of auto-encoder weights, if trained with the RNN together, would lead to over fitting due to the increase of parameters. Therefore, the pre-trained weights in encoder are fixed to prevent over fitting.

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

$$h_t = \sigma(W_{hx} v_t' + W_{hh} h_{t-1} + b_h) \quad (4)$$

$$y_t = \sigma(W_{hy} h_t + b_y) \quad (5)$$

The model predicts performance in every exercise but just one prediction is selected at each time step because just one label exists at that time. The loss function was defined to use cross-entropy, as is common in other RNN models.

4. DATASETS AND ENVIRONMENT

Three educational datasets are tested in this paper. Each of these datasets comes from a system in which students interact with an intelligent tutor system for math content. Area under the curve (AUC) and r-squared metrics are measured for each. The original DKT model with inputs that include only exercise tag and correctness is used as a model for comparison. Since it is a time-series algorithm, students whose records are less than 2 are not considered.[1]

4.1 ASSISTments 2009-2010 Data Sets

ASSISTments is a computer-based learning system that simultaneously teaches and assesses students. This dataset was gathered from ASSISTments skill builder problem sets [1], which are assignments in which a student works on similar questions until he/she can correctly answer n consecutive problems correctly (where n is usually 3). After completion, students do not commonly rework the same skill. Xiong et al [17] discovered three issues that have unintentionally inflated the performance of DKT in the original version, so the updated version of this dataset is adopted here.

Unlike other datasets, the records of a student may not be consecutive. That is why some previous works [2] report 15,391 students while others [17] report 4,217. In our model, all records that belong to one student are concatenated. The exercise tag is defined as the skill id.

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

In order to simplify the model and use cross features between time and others, the time is bucketed according to boundaries [-1, 60, 300, 1200, 3600, INF]. The boundary of hint count is defined as [-1, 0, 2, 4, INF], and [-1, 1, 20, 100, INF] for attempt count.

After preprocessing, this dataset consists of 4,217 students, 124 exercise tags and 338,000 records in total.

4.2 ASSISTments 2014-2015 Data Sets

In addition to the 2009-2010 skill builder set we felt it appropriate to include a more recent representation of student data within the ASSISTments platform.. We also used another dataset from ASSISTments that covers student response records from the 2014-2015 school years.

The process of feature processing with the exception of handling skill ids in this datasets is same as in the previous dataset. Unlike the ASSISTments 2009 dataset, some assignments have no mapped skill id so use the sequence id to represent skill id directly. Since the sequence level is finer than skill level, this process would introduce the noise to the dataset. The new skill id is mapped to the same pattern.

After pre-processing, the dataset consists of 19,103 students, 85 exercise tags, and 707,866 records.

4.3 KDD Cup 2010 Data Sets

KDD Cup 2010 is an education data mining competition organized by an ACM Special Interest Group on Knowledge Discovery and Data Mining (KDD) to predict student algebraic problem performance given information regarding past performance. The dataset came from Carnegie Learning's Cognitive Tutor in Algebra from years 2005-2009.

Unlike the ASSISTments platform, the Cognitive Algebra Tutor is part of an integrated curriculum and has more support for the learner during the problem-solving process. It provides a much finer representation of the concepts assessed by an individual item. Each step a student takes to answer problem is counted as a separate interaction, with each step potentially assessing different knowledge components (KCs). We use each interaction (step) as the finest problem for prediction, over 438 knowledge components representing skill. The exercise tag is a numerated knowledge component derived from the text description. A skill composed of several sub-components is considered as a separate knowledge component. Time is bucketed according to boundaries [-1, 10, 60, 150, 300, INF]. Hint usage is bounded by [-1, 2, 5, 10, INF]. As there is no attempt count field in this dataset, problem view is instead used and bucketed according the boundaries [-1, 2, 5, 10, INF].

After processing, the data set consist of 574 students, 438 exercise tags and 809,684 records.

4.4 Environment

For replicability, the running environment is reported as the following: Ubuntu 14.04, i5600 processor, 16G RAM, GTX 1070 (8G) graphics. The models were written with Tensorflow 0.10 using Python 3.4.

From our experiment, we find that LSTM has better performance than a traditional RNN and another variant, GRU, so only LSTM is reported for comparison. Similar to the DKT model, 200 hidden nodes are used. In order to prevent over fitting, dropout is applied to v_1, when computing v_1', and when computing y_1, but not in the computation of h_{t+1}. The dropout probability is set as 0.6. Binary cross entropy is the training objective which is trained by stochastic gradient descent on mini batches. The batch size for ASSISTments 2009 and 2014 is 30, while KDD is 5 because of the fewer number of students.

5. RESULT

The prediction is evaluated in terms of Area under curve (AUC) and the square of Pearson correlation (r2). Experiment undergoes 5-fold student level cross validation. There are a lot of possible feature and cross feature selection methods, but here we just explore few of them. AUC and r2 provide robust metrics for evaluation predictions where the value being predicted is either a 0 or 1 also represents different information on modeling performance. An AUC of 0.50 always represents the scored achievable by random chance. A higher AUC score represents higher accuracy. r2 is the square of Pearson correlation coefficient between the observed and predicted values of dependent variable.

Table1. AUC results

Model	2009	2014	KDD
DKT: exercise/correct	0.829	0.714	0.799
DKT +' time/correct	0.857	0.725	0.806
AE(DKT +' time/correct)	0.855	0.721	0.803
DKT +' time/correct +' time +' hint +' attempt	0.859	0.728	**0.808**
AE(DKT +' time/correct +' time +' hint +' attempt)	0.857	0.716	0.794
AE(DKT +' time/correct +' exercise/time +' time +' hint +' attempt)	**0.863**	**0.731**	**0.808**

Table1. The results of each of the explored models. The +' operator denotes concatenation. The attempt feature in KDD data refers to the problem view feature.

Table2. R2 results

Model	2009	2014	KDD
DKT: exercise/correct	0.323	0.115	0.234
DKT +' time/correct	0.387	0.129	0.245
AE(DKT +' time/correct)	0.387	0.124	0.239
DKT +' time/correct +' time + hint +' attempt	0.388	0.133	**0.250**
AE(DKT +' time/correct +' time +' hint +' attempt)	0.393	0.119	0.221
AE(DKT +' time/correct +' exercise/time +' time +' hint +' attempt)	**0.403**	**0.135**	**0.250**

On all three datasets, models with incorporated features outperform the original DKT model. In the ASSISTments 2009 dataset, AUC value is improved to 0.857 from 0.829 after adding the cross feature of exercise and time. However, the AUC value just increases 0.2% when adding more features such as time, hint usage and attempt count into the input vectors. Even adding a cross feature of exercise and time shows no further improvement.

The adoption of the auto-encoder when compared to models using the same features shows degraded performance of about 0.2%. In the ASSISTments 2014 dataset, it decreases to 0.716 from 0.728 while 0.808 to 0.794 in KDD data set. However, the auto-encoder is essential if more features are to be considered. For example, the input dimension of the last model, AE(DKT +' time/correct +' exercise/time +' time +' hint +' attempt), in KDD dataset is 3,079, which exhausted the GPU resources in our environment without an auto-encoder even when using small batch sizes. From the above results, the improvement of prediction is mainly contributed by incorporation of cross features.

7. CONCLUSION

The feature transformation and feature combination, when properly selected, can be used to improve the prediction accuracy. Although the parameters are difficult to interpret, such RNN models are adopted due to the performance gains.

The improvement here is attributed to the incorporation of cross features. The auto-encoder allows for the support of larger input vectors, making it possible to explore such combinations represented in one-hot encodings.

The work of extending these models has several potential directions to pursue. One such direction can explore even more features, engineered in different manners, such as tokening the words of knowledge components [6] for different exercise representations. Similarly, a wide and deep approach [16] 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. Bucketed according to the distribution within each exercise rather than across all exercises will likely improve the representation of those features. For example, because skill B is harder than skill A, most students may answer skill A in 20 seconds while the same student requires 300 seconds in skill B.

Because of flexible structure of deep learning, another research direction is to use similar RNN model structures to make other predictions regarding concepts like wheel spinning [4], student dropout, or hint usage.

8. REFERENCE

1. ASSISTments Data. (2015). Retrieved March 07, 2016, from https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010
2. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. In Advances in Neural Information Processing Systems (pp. 505-513).
3. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.
4. Beck, J. E., & Gong, Y. (2013, July). Wheel-spinning: Students who fail to master a skill. In International Conference on Artificial Intelligence in Education (pp. 431-440). Springer Berlin Heidelberg.
5. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation (No. ICS-8506). CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE.
6. Yu, H. F., Lo, H. Y., Hsieh, H. P., Lou, J. K., McKenzie, T. G., Chou, J. W., ... & Weng, J. Y. (2010). Feature engineering and classifier ensemble for KDD cup 2010. In Proceedings of the KDD Cup 2010 Workshop (pp. 1-16).
7. Van Inwegen, E. G., Adjei, S. A., Wang, Y., & Heffernan, N. T. (2015). Using Partial Credit and

Response History to Model User Knowledge. International Educational Data Mining Society.

8. Duong, H., Zhu, L., Wang, Y., & Heffernan, N. (2013, July). A prediction model that uses the sequence of attempts and hints to better predict knowledge:" Better to attempt the problem first, rather than ask for a hint". In Educational Data Mining 2013.
9. Wilson, K. H., Karklin, Y., Han, B., & Ekanadham, C. (2016). Back to the Basics: Bayesian extensions of IRT outperform neural networks for proficiency estimation. arXiv preprint arXiv:1604.02336.
10. Wilson, K. H., Xiong, X., Khajah, M., Lindsey, R. V., Zhao, S., Karklin, Y., ... & Heffernan, N. Estimating student proficiency: Deep learning is not the panacea.
11. Pavlik Jr, P. I., Cen, H., & Koedinger, K. R. (2009). Performance Factors Analysis--A New Alternative to Knowledge Tracing. Online Submission.
12. Stamper, J., Niculescu-Mizil, A., Ritter, S.,G.J Gordon, G., and Koedinger, K. Challenge data sets from KDD Cup 2010.
13. Khajah, M., Lindsey, R. V., & Mozer, M. C. (2016). How deep is knowledge tracing?. arXiv preprint arXiv:1604.02416.
14. Christopher Olah, Understanding LSTM Networks http://colah.github.io/posts/2015-08-Understanding-LSTMs/
15. Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. User modeling and user-adapted interaction, 4(4), 253-278.
16. Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... & Anil, R. (2016, September). Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (pp. 7-10). ACM.
17. Xiong, X., Zhao, S., Van Inwegen, E. G., & Beck, J. E. Going Deeper with Deep Knowledge Tracing. In Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016) (pp. 545-550).
18. Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009, June). Curriculum learning. In Proceedings of the 26th annual international conference on machine learning (pp. 41-48). ACM.