Deep Factorization Machines for Knowledge Tracing

Jill-Jênn Vie
RIKEN Center for Advanced Intelligence Project
Nihonbashi 1-4-1, Mitsui Building 15F
Chuo-ku, 103-0027 Tokyo, Japan
vie@jill-jenn.net

Abstract

This paper introduces our solution to the 2018 Duolingo Shared Task on Second Language Acquisition Modeling (SLAM). We used deep factorization machines, a wide and deep learning model of pairwise relationships between users, items, skills, and other entities considered. Our solution (AUC 0.815) hopefully managed to beat the logistic regression baseline (AUC 0.774) but not the top performing model (AUC 0.861) and reveals interesting strategies to build upon item response theory models.

1 Introduction

Given the massive amount of data collected by online platforms, it is natural to wonder how to use it to personalize learning. Students should receive, based on their estimated knowledge, tailored exercises and lessons, so they can be guided through databases of potentially millions of exercises.

With this objective in mind, numerous models have been designed for student modeling (Desmarais and Baker, 2012). Based on the outcomes of students, one can infer the parameters of these so-called student models, measure knowledge, and tailor instruction accordingly.

In the 2018 Duolingo Shared Task on Second Language Acquisition Modeling (Settles et al., 2018), we had access to attempts of thousands of students over sentences (composed of thousands of possible words, each of these being labeled as correct or incorrect), and we had to predict whether a student would write correctly or not the words of a new sentence. Sentences were annotated with precious side information such as lexical, morphological, or syntactic features. This problem is coined as knowledge tracing (Corbett and Anderson, 1994) or predicting student performance (Minaei-Bidgoli et al., 2003) in the literature. In this particular challenge, it is done at the word level.

In this paper, we explain the motivations that led us to our solution, and show how our models handle typical models in educational data mining as special cases. In Section 2, we show related work. In Section 3, we present the existing model of DeepFM and clarify how it can be applied for knowledge tracing, notably the SLAM task. In Section 4, we detail the data preparation, in order to apply DeepFM. Finally, we expose our results in Section 5 and further work in Section 6.

2 Related Work

Item Response Theory (IRT) models (Hambleton et al., 1991) have been extensively studied and deployed in many real-world applications such as standardized tests (GMAT). They model the ability (level information) of students, and diverse parameters of items (such as difficulty), and involve many criteria for the selection of items to measure the ability of examinees.

Related work in knowledge tracing consists in predicting the sequence of outcomes for a given learner. Historically, Bayesian Knowledge Tracing (BKT) modeled the learner as a Hidden Markov model (Corbett and Anderson, 1994), but with the advent of deep learning, a Deep Knowledge Tracing (DKT) model has been proposed (Piech et al., 2015), relying on long short-term memory (Hochreiter and Schmidhuber, 1997). However, Wilson et al. (2016) have shown that a simple variant of IRT could outperform DKT models.

All of these IRT, BKT or DKT models do not consider side information, such as knowledge components, which is why new models naturally rose. Vie and Kashima (2018) have used Bayesian factorization machines for knowledge tracing, and recovered most student models as special cases.

Wide and deep learning models have been proposed by Google (Cheng et al., 2016) to learn lower-order and higher-order features. Guo et al. (2017) have proposed a variant where they replace the wide linear model by a factorization machine, and this is the best model we got for the Shared Task challenge.

3 DeepFM for knowledge tracing

We now introduce some vocabulary. We assume that our observed instances can be described by C *categories* of discrete or continuous features (such as `user_id`, `item_id` or `country`, but also `time`). *Entities* denote couples of categories and discrete values (such as `user=2`, `country=FR` or again `time` if the category is continuous). We denote by N the number of possible entities, number them from 1 to N. The DeepFM model we are describing will learn an embedding for each of those entities[1].

Each instance can be encoded as a sparse vector $\boldsymbol{x}$ of size N: each component will be set at a certain value (for example, 1 if the category of the corresponding entity is discrete, the value itself if it is continuous, and 0 if the entity is not present in the observation). For each instance, our model will output a probability $p(\boldsymbol{x}) = \psi(y_{FM} + y_{DNN})$, where ψ is a link function such as the sigmoid σ or the cumulative distribution function (CDF) Φ of the standard normal distribution.

The DeepFM model is made of two components, the FM component and the Deep component.

3.1 FM component

Given an embedding size $d \in \mathbf{N}$, the output of a factorization machine is the following:

$$y_{FM} = \sum_{k=1}^{N} w_k x_k + \sum_{1 \leq k < l \leq N} x_k x_l \langle \boldsymbol{v_k}, \boldsymbol{v_l} \rangle$$

The first term shows that a bias $w_k \in \mathbf{R}$ is learned for each entity k. The second term models the pairwise interactions between entities by learning a vector $\boldsymbol{v_k} \in \mathbf{R}^d$ for each entity k.

[1]The original DeepFM paper (Guo et al., 2017) chooses *fields* and *features* in lieu of *categories* or *entities*, but we prefer to use our own formulation (Vie and Kashima, 2018) because we usually agree with ourselves.

3.1.1 Relation to existing student models

If $d = 0$ and ψ is the sigmoid function σ, $p(\boldsymbol{x}) = \sigma(\langle \boldsymbol{w}, \boldsymbol{x} \rangle)$ and the FM component behaves like logistic regression.

In particular, if there are two fields users (of n possible values) and items, then each instance encoding $\boldsymbol{x}_{ij}$ of user i and item j is a concatenation of two one-hot vectors, and $p(\boldsymbol{x}_{ij}) = \sigma(w_i + w_{n+j}) = \sigma(\theta_i - d_j)$ for appropriate values of w, which means the Rasch model is recovered.

As pointed out by Settles et al. (2018), their baseline model is a logistic regression with side information, which makes it similar to an additive factor model. To see more connections between our FM component and existing educational data mining models, see Vie and Kashima (2018).

3.2 Deep component

The deep component is a L-layer feedforward neural network that outputs:

$$y_{DNN} = \text{ReLU}(W^{(L)} a^{(L)} + b^{(L)})$$

where each layer $0 \leq \ell < L$ verifies:

$$a^{(\ell+1)} = \text{ReLU}(W^{(\ell)} a^{(\ell)} + b^{(\ell)})$$

for learned parameters W, a, b for each layer, and the first layer is given by the corresponding $\boldsymbol{v_{i_c}}$ embeddings of the activated entities (the ones for each category $c = 1, \ldots, C$, which correspond to the nonzero entries of $\boldsymbol{x}$):

$$a^0 = (\boldsymbol{v_{i_1}}, \ldots, \boldsymbol{v_{i_C}}).$$

In order to select the hyperparameters, we followed the instructions of (Guo et al., 2017) and the default values of the available implementation on GitHub[2].

3.3 Training

Training is performed by minimizing the log loss of the output probabilities compared to the true outcomes of the students over the tokens. For all models trained, the optimizer was Adam (Kingma and Ba, 2014), with learning rate $\gamma = 10^{-3}$ and minibatches of size 1024.

4 Encoding the Duolingo Dataset

4.1 Fundamental, discrete categories

Fundamental categories (`<fundamental>`) refer to the features that have discrete values, such as

[2]https://github.com/ChenglongChen/tensorflow-DeepFM

user (which refer to the user ID) or countries (which can be in a many-to-many relationship).

- user
- token
- part_of_speech
- dependency_label
- exercise_index
- countries
- client
- session
- format

4.2 Noisy discrete categories

Duolingo was providing the SyntaxNet features (morphosyntactic rules) such as:

- Definite
- Gender
- Number
- fPOS
- Person
- PronType
- Mood
- Tense
- VerbForm

We call them noisy (<noisy> below), because they are the output of another algorithm. Also, not all of them were specified, there were some missing entries.

4.3 Continuous categories

- time for answering the question
- days since when the user subscribed the Duolingo platform.

4.4 Encoding

In the baseline model provided by Duolingo, all fundamental features were encoded as a concatenation of n-hot encoders[3]. Then they used logistic regression and achieved AUC 0.772.

Here are the models we considered.

- IRT: user + token, $d = 0$
- Logistic regression baseline: <fundamental>
- Vanilla FM: <fundamental>
- DeepFM: <fundamental>
- DeepFM*: <fundamental> + <noisy> + <continuous>

The implementation of Deep Factorization Machines we used needed a concatenation of one-hot encoders. So we picked the first country among the list of countries for each instance. Also, it could not handle missing entries, so for the noisy partial categories, we used a None entity.

[3]For this reason, the continuous features could not be used for the baseline.

	ACC	AUC	NLL	F1
IRT + attempts	0.833	0.739	0.411	
Basic IRT	0.838	0.752	0.399	
LR baseline	0.838	0.772	0.391	0.284
Vanilla FM	0.824	0.773	0.414	
DeepFM*		0.811		**0.382**
DeepFM		**0.815**		0.329

Table 1: Performance of all tested algorithms on the en_es dataset.

5 Results

We first tried different models on a validation set. All models were trained using 500 epochs for the vanilla FM, or 100 epochs for DeepFM with early stopping, and refit on the validation set.

5.1 On validation set

A vanilla FM was used considering $\psi = \Phi$ the CDF of the standard normal distribution as link function, like in the implementation of[4] (Rendle, 2012). Then, for our experiments, we used the TensorFlow implementation of DeepFM provided by Alibaba on GitHub[5]. Our encoding is available on GitHub[6].

Vanilla FM had comparable performance of the LR baseline. It agrees with the findings of Vie and Kashima (2018) that a bigger dimension may not necessarily help.

5.2 On test set

The DeepFM model managed to improve the baseline by 3 points AUC. We got AUC 0.815, while the top performing solution had AUC 0.861.

Our best performing model was DeepFM: using only the discrete features, train a model of latent embedding size 10 during a fixed number of epochs (50). DeepFM* using all features was slightly worse.

6 Further Work

We could embed the dependency graph provided by Duolingo in the encoding of the vanilla FM.

Ensemble methods such as xgboost (Chen and Guestrin, 2016) could be considered, as typically encountered in challenges.

[4]http://www.libfm.org
[5]https://github.com/ChenglongChen/tensorflow-DeepFM
[6]https://github.com/jilljenn/ktm

Here we want to combine information of the student which is quite poor (almost only their outcomes), compared to the knowledge of tokens (syntactic trees, or word2vec, etc.). This is why we could use extra embeddings, such as a LSTM encoding of the sentence as feature for the token.

The performance of DeepFM* that was using all features was slightly worse than DeepFM that was limited to the fundamental features. We might mitigate this problem by using a field-aware factorization machine (Juan et al., 2016) that learns a parameter per category of feature in order to draw more importance on some category (such as `user`) than others (such as `date`).

7 Conclusion

In this paper, we showed how to use deep factorization machines for knowledge tracing. Our findings show interesting combinations of features, together with embeddings provided by deep neural networks. In some way, it shows how to learn dense embeddings from the sparse features typically encountered in learning platforms.

References

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM.

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

Michel C. Desmarais and Ryan S. J. D. Baker. 2012. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, 22(1-2):9–38.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1725–1731. AAAI Press.

Ronald K. Hambleton, Hariharan Swaminathan, and H. Jane Rogers. 1991. *Fundamentals of item response theory*. Sage.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM.

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

Behrouz Minaei-Bidgoli, Deborah A Kashy, Gerd Kortemeyer, and William F Punch. 2003. Predicting student performance: an application of data mining methods with an educational web-based system. In *Frontiers in education, 2003. FIE 2003 33rd annual*, volume 1, pages T2A–13. IEEE.

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

Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57:1–57:22.

B. Settles, C. Brust, E. Gustafson, M. Hagiwara, and N. Madnani. 2018. Second language acquisition modeling. In *Proceedings of the NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*. ACL.

Jill-Jênn Vie and Hisashi Kashima. 2018. Knowledge tracing machines.

Kevin H. Wilson, Yan Karklin, Bojian Han, and Chaitanya Ekanadham. 2016. Back to the basics: Bayesian extensions of IRT outperform neural networks for proficiency estimation. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM)*, pages 539–544.