Neuro-Symbolic Models: A Scalable, Explainable Framework for Strategy Discovery from Big Edu-Data

Deepak Venugopal
University of Memphis
dvngopal@memphis.edu

Vasile Rus
University of Memphis
vrus@memphis.edu

Anup Shakya
University of Memphis
ashakya@memphis.edu

ABSTRACT

Predicting student problem-solving strategies is a complex problem but one that can significantly impact automated instruction systems since they can adapt or personalize the system to suit the learner. While for small datasets, learning experts may be able to manually analyze data to infer student strategies, for large datasets, this approach is infeasible. While Deep Neural Network (DNN) based methods such as LSTMs can be applied for this task, they have drawbacks such as long convergence times for big datasets, and like DNN-based methods in general, have the inherent problem of overfitting the data. To address these issues, we propose a general Neuro-symbolic framework for strategy prediction, where we combine the strengths of symbolic AI (that can encode domain knowledge) with DNNs. We outline several possible benefits of this framework and demonstrate its potential in scalable learning from large educational datasets.

Keywords

Intelligent Tutoring Systems, Learning Strategies, Neuro-Symbolic AI, Markov Logic Networks, LSTMs

1. INTRODUCTION

Intelligent Tutoring Systems (ITSs) [19] and more broadly adaptive instructional systems (AISs)[1] help a diverse population of students by adapting instruction to each learner thus accounting for different learning abilities, learning styles and education goals. However, in order to build effective ITSs, it is important to understand how students learn and what learning and instructional strategies are most effective for whom and under what conditions. Specifically, students can follow several different *strategies* to learn the same content. Depending upon the way a student thinks, one strategy could be easier or harder to grasp compared to the other. Thus, understanding the various ways in which students approach an instructional task will not only further our understanding of how learners learn but will help in adapting ITSs for a personalized learning experience.

[1]The main difference between Intelligent Tutoring Systems and Adaptive Instructional Systems at least in our view is that the former offer full-adaptivity, i.e., both micro- and macro-adaptivity, whereas the latter can offer any type, e.g., just macro-adaptivity.

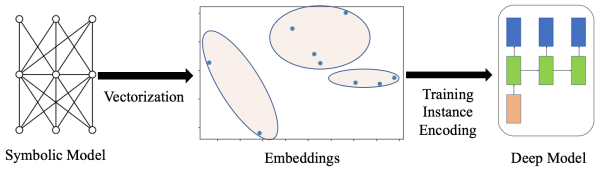


Figure 1: Schematic illustration of a general Neuro-symbolic architecture for strategy discovery. The symbolic model (defined in an appropriate language, e.g. Markov Logic) defines relationships in the data which can be represented as a graphical structure. From the symbolic model, we learn embeddings for nodes in the graph such that the embeddings encode local structures in the graph. We then construct training examples based on the embedding-space for a deep model that predicts strategies.

A student's choice of strategy is a complex function dependent on many factors such as experience with similar problems, general expertise in the topic, other cognitive abilities, etc. Big data and access to advanced reasoning and large computing infrastructure offer new possibilities to discover personalized strategies. In particular, a paradigm that is gaining significant attention in the AI/Machine Learning research community is Neuro-symbolic AI [5] where we augment DNNs with symbolic models to regularize the DNN. This helps in improving both scalability and generalization by allowing DNNs to learn from smaller datasets with higher accuracy. In this paper, we provide an overview of a possible framework for strategy discovery using Neuro-Symbolic learning and outline its potential benefits.

2. STRATEGY PREDICTION

Student strategies can be defined in different ways. In particular, the definition of what constitutes a strategy also depends upon the type of interaction the student has with the AIS. In our case, we only consider structured interactions with discrete steps. Therefore, we define the student strategy as a function of the sequence of steps in the interaction. Each step is characterized by the central concept

that is utilized to solve that specific step, i.e., the knowledge component [14] (KC) used in that step. Therefore, we define strategy in our case as a sequence of KCs used by a student in a problem solving session. Note that, this formulation of strategy as a sequence of discrete components is similar to the definitions used in prior work [18]. Formally,

DEFINITION 1. *Given a student s and a problem p, we define the strategy as $\bar{\mathbf{x}}_{s,p} = K_{s,p}^{(1)} \dots K_{s,p}^{(n)}$, where $K_{s,p}^{(i)}$ is the knowledge-component that s uses to solve the i-th discrete step in p.*

We can now formulate a learning problem as follows. Given training data, $\{\mathbf{x}_{s_i,p_j}\}_{i=1,j=1}^{n,m_i}$, where n is the number of students and m_i is the number of problems solved by the i-th student, we learn a model $\mathcal{P}$ such that for a student s' and problem p', $\mathcal{P}$ generates a sequence of knowledge components $K_{s',p'}^{(1)} \dots K_{s',p'}^{(k)}$. Note that students sometimes use more than one KC per step, in this case, we just unroll these multiple KCs by repeating the step with each of the KCs. Therefore, for the rest of this paper, we treat both multiple KCs in a step and single KC steps without distinguishing them. Also, to keep notation simpler, instead of adding the subscripts s_i, p_j each time, we denote the training input and output pairs as $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$.

3. NEURO-SYMBOLIC FRAMEWORK

A possible Neuro-symbolic framework for predicting strategies is shown in Fig. 1. This consists of three components,

1. **Symbolic Model.** A generic symbolic model that represents relationships in the data. There are several languages that have been developed to encode symbolic knowledge such as Markov Logic [6], Probabilistic Soft Logic [2], Knowledge graphs [3], Probabilistic graphical models [15], arithmetic circuits [4], etc. We can typically represent the symbolic relationships in the data encoded by these languages in some form of a graphical structure. For example, a student can be connected to other students in the same class, a problem can be connected to another problem from the same topic, etc. Thus, instead of considering multiple instances of the data, we have a single very large instance where the instances are interconnected through relationships.

2. **Embeddings.** To encode symbolic knowledge for neural networks, we need to vectorize (or create embeddings for) the nodes in the graphical structure. Generating node embeddings from graphs is also a popular area of research and there have been several models that learn embeddings to preserve graph structure (at least locally). Some example models include Graph Convolutional Neural Nets [13], Node2Vec [7], tensor factorization methods [22], Obj2Vec [10], etc. Projecting the nodes in the graph into an embedding space allows us to define distances/similarities between nodes. For example, we can identify similarities between students, problems, etc. based on vector distances in the embedding.

3. **Deep Model.** The deep model learns a representation that combines one or more embeddings to predict a strategy for a new instance in the data. A typical architecture that can be applied here is recurrent neural networks (RNNs). Specifically, since we are predicting sequential outputs, RNNs and other related architectures such as Long Short Term Memory (LSTMs) [8] use a latent representation that summarizes previous predictions to determine the next possible step in the strategy. LSTMs in particular are well-known to encode long-range dependencies into the latent representation. This means that for long strategies, there may be dependence spread across steps that are several steps apart and such dependencies can be encoded using the LSTM.

3.1 Potential Benefits

While neural networks by themselves are extremely powerful models, they have certain limitations. In general, a well-recognized problem in several deep learning architectures is that, due to their expressive power, they tend to overfit the data. Further, training a deep network can be computationally very expensive and require large amounts of labeled data. For instance, LSTMs are known to converge very slowly for large datasets [23]. In the domain of education, labeled data is hard to obtain compared to other domains such as image processing (where deep models have achieved greatest gains). Collecting educational data requires significant time investment and expertise. Neuro symbolic learning can regularize deep learning and thus allow it to learn to generalize better with smaller data. More generally, we outline some potential benefits/applications of our proposed Neuro-symbolic framework below.

1. **Big Data Training.** The symbolic model specifies symmetries in the graph. Exchangeable nodes are those that can be exchanged in the graph to yield an isomorphic graph. Though, identifying exchangeable nodes is a hard problem, there are efficient techniques to compute approximately exchangeable nodes. We can use exchangeability to scale up to large datasets since instances chosen from an exchangeable group are more likely to have similar strategies. Thus, instead of training over big data which is computationally expensive for deep models, we can reduce the data to contain diverse instances in the data, where each instance effectively represents a group of exchangeable instances.

2. **Joint Modeling.** In joint models, the idea is that the predictions are related to each other. For example, suppose we want to predict mastery and strategy, conditioned on the mastery of the student the type of strategy the deep model may predict may be constrained. Similarly, conditioned on the strategy followed by a student, we may be able to predict mastery more accurately. To predict both jointly, we can develop Expectation Maximization-like algorithms that optimize one assuming the other and alternate the optimizations in each step.

3. **Transparency.** A key difficulty with DNNs is that it is hard to understand or trace their predictions down to explainable features. This is a huge disadvantage

in EDM since for practical use (e.g. in schools), it is essential that strategies predicted by the model are transparent in the sense that they are accompanied by a trace of why a strategy is appropriate for a student (or group of students). To do this, one possible approach in our framework is to *perturb* the embeddings by modifying the symbolic model and observing changes to the deep model predictions. Thus, we create a mapping between the predictions and the explainable symbolic model. This is similar to explanation models such as LIME [17] and SHAP [16] that rank features based on the effects their perturbation has on the predictions.

4. **Fairness.** A major consideration in education is to ensure algorithmic fairness, i.e., the algorithms must not be tuned to favor a "majority" group to achieve high accuracy in terms of performance metrics [9]. The Neuro-symbolic framework can be utilized to encode fairness as constraints within the symbolic model. Thus, the type of models learned by the DNN are constrained to follow constraints set by the designers. Implementation of fairness is likely to require a human-in-the-loop where the transparent model generates a log that can be audited by a human to verify if the learned model adheres to fairness principles.

5. **Uncertainty Estimation.** Often, it is as important for the model to say "I am unsure of a prediction" rather than simply present a prediction. Thus, in the case of predicting strategies, we want the model to present a probability of a predicted strategy. To do this, we can place a joint distribution over the instances and perform probabilistic inference. Importantly, since the instances are related, we can quantify uncertainty of a prediction by considering the predictions over related instances.

3.2 Illustrative Example: Scalable Learning

To illustrate the potential benefits of our Neuro-symbolic framework, we present an approach to learn strategy prediction in a scalable manner. Details for this approach are available here [20]. Specifically, we use the language of Markov Logic to construct a symbolic representation of the relationships in the data. However, since this graph can be very large, using graph-based methods to identify symmetries is infeasible. Therefore, we use an approach called Obj2vec [10] that is designed to construct embeddings for symbols in a first-order knowledge base. Given the embeddings, we sample training instances selectively to train an LSTM model for strategy prediction.

Specifically, let our input instances be $\mathbf{x}_1 \ldots \mathbf{x}_N$, where each $\mathbf{x}_i$ consists of embeddings for a specific student s solving problem p, and the outputs are $\mathbf{y}_1 \ldots \mathbf{y}_N$, where $\mathbf{y}_i$ is a sequence of KCs used by the student s to solve the problem p. The LSTM training objective is given by,

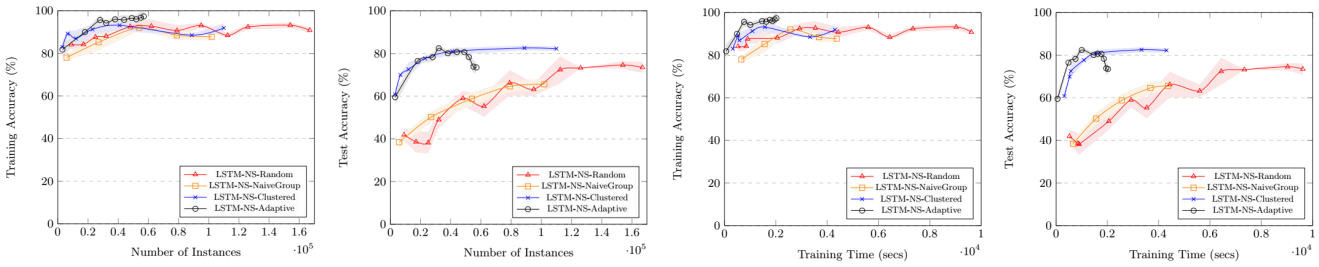
$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\psi(\mathbf{x}_i, \theta), \mathbf{y}_i) \quad (1)$$

where θ^* and θ represent the parameters of the LSTM, $\mathcal{L}$ is a loss function and $\psi(\mathbf{x}_i, \theta)$ is the sequence of KCs output by the LSTM parameterized by θ for input $\mathbf{x}_i$. In general, a stochastic gradient descent (SGD) procedure can be used to minimize the objective in Eq. (1). In SGD, we sample the training instances to approximate the gradient. Typically, SGD assumes that all training instances are equally important, and therefore samples them uniformly. That is, the probability of sampling a specific instance in the training data is equal to $p = \frac{1}{N}$. However, this approach is expensive particularly if we repeatedly choose training instances that are similar to each other. For example, suppose all the training instances that we sample are likely to encode similar strategies, then our model may take a long time to understand diverse strategies. Therefore, we force the model to learn from instances with diverse relationships by imposing an importance distribution over the training data. Specifically, training instances with larger importance are more likely to be chosen.

In general, to focus the training on more important data instances, we can modify the sampling distribution such that each instance is sampled with a non-uniform probability. This approach has been explored in prior work, where we scale up training by replacing the uniform distribution over the training instances with an importance distribution that quantifies how important a specific example is for the training process [12]. Previous work such as [12, 1, 11], have focused mainly on approximating importance as a function of the gradient norm which is hard to compute exactly. In [12], therefore, the authors propose an approximation to the gradient norm and use this to target important training examples. The focus in these approaches is to target the training examples that are likely to induce changes when updating the model parameters during backpropagation which can be shown to translate to a reduced variance in the gradient estimates. However, in our case, we have more information apriori in the embeddings to identify importance of a training example in terms of their relationships. Specifically, recall that the embeddings are based on symmetries specified in the knowledge base. Our model focuses on instances with diverse relationships using embedding-similarities. Specifically, we cluster the embeddings and sample from the clusters to construct a sub-sampled training dataset. However, since the embedding is approximate, there is uncertainty about whether clustered instances are truly similar. Therefore, by adapting the importance distribution as training progresses, we place more importance on clusters where it is harder to predict strategies.

3.2.1 Experiments

We show some scalability results on the publicly available KDD EDM challenge dataset `Bridge to Algebra 2008 - 2009` [21], which consists of data collected from the Mathia platform. Each instance consists of several discrete steps and each step is mapped to a knowledge component which is used to solve that step. All results shown were based on experiments performed on a 64GB memory machine with a Nvidia GPU and an Intel Core-I9 processor. For computing accuracy, in each input instance, we compute the percentage of total steps where the true KC matches with the predicted KC. The overall accuracy is computed as the average accuracy across all instances. To measure variance of our estimates, for each of the results shown, we run the experiments 10 times and compute the mean accuracy and the standard deviation of the accuracy.



(a) Train (b) Test (c) Train (d) Test

Figure 2: Accuracy results for `Bridge to Algebra 2008 - 2009`, the shaded portions show the standard deviation and the mean accuracy is plotted in the graphs.

For our first baseline, we randomly sampled instances to train our model given a timeout window. We refer to the trained models using random sampling as `LSTM-NS-Random`. We further implemented a stratified-sampling/group based training on students and problems. For sampling by student, we selected N students from the student pool and for each selected student, we sampled M problems solved by that student. For sampling by problems, we selected N problems from the problem pool and sampled M students who have solved those problems. By increasing values of M and N, we progressively increased the instances as we show later in the results section. We refer to this as `LSTM-NS-NaiveGroup`. `LSTM-NS-clustered` samples from clusters but does not adapt the importance weights. `LSTM-NS-Adaptive` denotes the adaptive importance sampling approach. Fig. 2 compares the accuracy for different approaches. Fig. 2 (a), (b) shows the training and test accuracy respectively as we vary the number of instances used in training the models for the `Bridge to Algebra 2008 - 2009` dataset (we have not shown the results for `Algebra 2008 - 2009` due to space constraints). As we can see from these plots, `LSTM-NS-Adaptive` obtains the highest accuracy in the shortest training time compared to the other methods. In particular `LSTM-NS-Adaptive` achieves a larger or comparable accuracy in training time that is less than 10% of the time required to train `LSTM-NS-Random` and `LSTM-NS-NaiveGroup`. This illustrates that using symmetries from the symbolic model to sample the instances can have a significant impact on scalability of the Neuro-symbolic model.

4. CONCLUSION

Predicting student strategies in problem solving can make AISs more robust since the system can adapt itself to suit the student's strategy. In this paper, we outlined a Neuro-symbolic framework for this problem. We discussed potential benefits of this approach and provided an illustrative case to demonstrate how this framework can improve scalability in training the model.

In the future, we will try to work on one or more of the challenging potential applications of this framework as mentioned in section 3.

5. ACKNOWLEDGEMENTS

This research was sponsored by the National Science Foundation under the awards The Learner Data Institute (award #1934745) and NSF IIS award #2008812. The opinions, findings, and results are solely the authors' and do not reflect those of the funding agencies.

References

[1] G. Alain, A. Lamb, C. Sankar, A. C. Courville, and Y. Bengio. Variance reduction in SGD by distributed importance sampling. *CoRR*, abs/1511.06481, 2015.

[2] S. H. Bach, M. Broecheler, B. Huang, and L. Getoor. Hinge-loss markov random fields and probabilistic soft logic. *J. Mach. Learn. Res.*, 18:109:1–109:67, 2017.

[3] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[4] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[5] A. S. d'Avila Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP*, 6(4):611–632, 2019.

[6] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA, 2009.

[7] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 855–864, 2016.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] A. M. Howard, C. Zhang, and E. Horvitz. Addressing bias in machine learning algorithms: A pilot study on emotion recognition for intelligent systems. In *ARSO*, pages 1–7. IEEE, 2017.

[10] M. M. Islam, S. Sarkhel, and D. Venugopal. On lifted inference using neural embeddings. In *AAAI Conference on Artificial Intelligence*, pages 7916–7923, 2019.

[11] T. B. Johnson and C. Guestrin. Training deep models faster with robust, approximate importance sampling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 7276–7286, 2018.

[12] A. Katharopoulos and F. Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2525–2534, 2018.

[13] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[14] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cogn. Sci.*, 36(5):757–798, 2012.

[15] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[16] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining (KDD)*, 2016.

[18] S. Ritter, R. Baker, V. Rus, and G. Biswas. Identifying strategies in student problem solving. *Design Recommendations for Intelligent Tutoring Systems*, 7:59–70, 2019.

[19] V. Rus, S. K. D'Mello, X. Hu, and A. C. Graesser. Recent advances in conversational intelligent tutoring systems. *AI Magazine*, 34(3):42–54, 2013.

[20] A. Shakya, V. Rus, and D. Venugopal. Student strategy prediction using a neuro-symbolic approach. In *EDM*, 2021.

[21] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Algebra I 2008-2009. Challenge data set from KDD Cup 2010 Educational Data Mining Challenge. Technical report, 2010.

[22] T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard. Knowledge graph completion via complex tensor factorization. *J. Mach. Learn. Res.*, 18:130:1–130:38, 2017.

[23] Y. You, J. Hseu, C. Ying, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large-batch training for lstm and beyond. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019.