A Meta-learning Framework for Personalized Question Selection

Aritra Ghosh
University of Massachusetts Amherst
Amherst, MA 01002
`arighosh@cs.umass.edu`

Andrew S. Lan
University of Massachusetts Amherst
Amherst, MA 01002
`andrewlan@cs.umass.edu`

Abstract

This paper details our winning solution to Task 4 of the NeurIPS 2020 Education Challenge.[1] We use a meta-learning framework for training and testing, a fully-connected neural network model for response correctness prediction, and an active learning algorithm for question selection.

1 Introduction

In this paper, we detail our approach for Task 4 of the NeurIPS 2020 Education Challenge. Note that since the data for tasks 3&4 is different from that for tasks 1&2 (especially that it is much smaller), we use a different model (one that fits better to smaller data) to drive adaptive question selection than what we used for tasks 1&2. Please see our other paper for details on our model for those tasks.

Deep learning models require large amounts of training data to excel. However, in many real-world application domains, e.g., education, geoscience, and economics, there are often contain limited amount of data available; labeling is often a labor-intensive process. Therefore, there has been a recent trend to learn effective deep learning models in data-scarce settings, including few-shot learning and meta-learning [3, 13]. In particular, meta-learning based models are highly successful in cases where learning needs to be done from a few data points per category of interest [3, 12].

In educational applications, we need to track student learning progress over time and predict their future performance. In the knowledge tracing literature, many models have been proposed to learn student's evolving knowledge state [2] from their past responses to assessment questions. However, state-of-the-art models are highly complex and require a large amount of training data [4]. Therefore, estimating a student's knowledge levels and predicting their future performance with *limited* data for *complex* models is crucial in educational applications. Prior work in computerized adaptive testing [1, 6] primarily leverage simple item response theory models [7, 10] that are not flexible enough to leverage large amounts of training data.

The main idea we follow in this paper is that we need to ensure *training and testing procedure must match as much as possible* [13]. We first formulate the problem of learning from limited data points as a meta-learning problem. We use a training framework that matches exactly how we evaluate our method in the testing framework. Furthermore, we detail how we use a policy (either fixed or learnable) to select questions for assessment to improve predictive performance on future student performance.

1st NeurIPS Education Challenge (NeurIPS 2020), Vancouver, Canada.

[1]Our code is available at `https://github.com/arghosh/NeurIPSEducation2020`.

2 Methodology

2.1 Problem Formulation

A student's record consists of their responses to various questions and we do not have timing information about these responses. For student i, for question $j \in \{1, T_i\}$ (where T_i is the number of questions the student answered), we denote the combination of the question that they responded to, their binary-valued response correctness, and the option they chose on this question as a tuple, (x_j^i, y_j^i, a_j^i), where $x_j^i \in \mathbb{N}^+$ is the question index, $y_j^i \in \{0, 1\}$ is the response correctness (1 corresponds to a correct response, which means the student selected the correct option of the question), and $a_j^i \in \{A, B, C, D\}$ is the option the student selected for this question. We use j to index the questions due to the adaptive question selection process; they do not correspond to timestamps or the actual order in which the student responded to questions. Each observation also contains the subjects tagged with the question and the correct option; to remove excessive notational clutter, we do not discuss them in the modeling descriptions and provide some initial observation for these features in Section 3.

For a total of N students, we use N^{train} students for training and the other $N^{\text{test}} = N - N^{\text{train}}$ students for testing purposes. For student i (in either the training or the test set), we split the questions they responded to that are observed into two parts; the first part contains T_{tr}^i training (support) questions, i.e., $j \in \mathcal{D}_{\text{tr}}^i$, and second part is the set of a total of T_{meta}^i meta (query) questions $\mathcal{D}_{\text{meta}}^i$. We would like to select n $(n \ll |\mathcal{D}_{\text{tr}}^i|)$ questions sequentially from the training set $\mathcal{D}_{\text{tr}}^i$, observe their responses and predict their responses for questions in the meta set $\mathcal{D}_{\text{meta}}^i$. We detail our procedure for splitting questions into training and meta sets for each student in Section 3.

We use a neural network $f(\cdot)$ to predict the response a student makes to a question (in the meta set) and refer to it as the classifier network. The classifier network is parameterized by Θ and takes as input question index q_t^i, and a context variable $\mathbf{w}^i(\Theta)$ (also a function of Θ) as input and make a binary prediction for the question q_t^i. The context variable $\mathbf{w}^i(\Theta)$ encodes i) the estimated student knowledge state based on their responses to a small portion of the questions in the training set and ii) the student's learning context, i.e., which questions have they responded to, $\mathcal{D}_{\text{tr}}^i$; we detail the particular choice for parameterizing $\mathbf{w}^i(\Theta)$ later in this section. We can view each student as equivalent to a new task in the meta-learning framework; in many black-box few-shots learning models, each task is often encoded as a context variable [8, 11]. The loss function $\ell\Big(y_t^i, f(q_t^i, \mathbf{w}^*(\Theta), \Theta)\Big)$ is usually set to the binary cross-entropy loss between observed response and the predicted response probabilities. The context variable $\mathbf{w}^*(\Theta)$ is computed using the function $g(\cdot)$ that takes as input i) the questions that they responded to, $\mathcal{D}_{\text{tr}}^i$, and ii) their responses to a small subset of questions, $\Phi(\mathcal{D}_{\text{tr}}^i)$, which is selected by a personalized question selection policy. We solve the following optimization problem:

$$\min_{\Theta} \mathcal{L}^{\text{meta}}(\Theta) \triangleq \sum_{i=1}^{N^{\text{train}}} \frac{1}{|\mathcal{D}_{\text{meta}}^i|} \sum_{t \in \mathcal{D}_{\text{meta}}^i} \ell\Big(y_t^i, f(q_t^i, \mathbf{w}^i(\Theta), \Theta)\Big) \quad (1)$$
$$\text{s.t. } \mathbf{w}^i(\Theta) = g(\mathcal{D}_{\text{tr}}^i, \{y_t, a_t\}_{t \in \Phi(\mathcal{D}_{\text{tr}}^i)}; \Theta),\ \forall i \in [1, N^{\text{train}}].$$

We can also optionally jointly optimize the policy (e.g., an reinforcement learning policy) Φ as,

$$\min_{\Theta, \Gamma} \mathcal{L}^{\text{meta}}(\Theta) \text{ s.t. } \mathbf{w}^i(\Theta) = g(\mathcal{D}_{\text{tr}}^i, \{y_t, a_t\}_{t \in \Phi_\Gamma(\mathcal{D}_{\text{tr}}^i)}; \Theta),\ \forall i \in [1, N^{\text{train}}],$$

where the question selection policy is parameterized by Γ. We experimented with both optimizing the policy (using a actor-critic framework) and a fixed active learning policy (uncertainty sampling). Our initial results were similar for both of these approaches; but we chose to experiment more with the latter (fixed policy) for rapid experimentation. We discuss briefly the choice of the policy Φ later in this section.

2.2 Model

Some existing meta-learning methods use recurrent neural networks or memory network-based architectures. However, we do not have explicit ordering among the questions since their exact timestamps are unknown; thus, we chose to use a feed-forward neural network that is invariant to the order in which the responses are observed. We use two embedding modules; the embedding module $\mathcal{E}_l(y)$ embeds the binary-valued response $y \in \{0, 1\}$ into a real-valued vector in $\mathbb{R}^d$, and the embedding module $\mathcal{E}_a(a)$ embeds a selected option $a \in \{A, B, C, D\}$ into a real-valued vector in $\mathbb{R}^d$. For a total of Q questions, we represent the student inputs as,

$$\begin{aligned}\mathbf{x}^i &= [(\mathcal{E}_l(y_1^i) \odot f_1^i) \oplus (\mathcal{E}_a(a_1^i) \odot f_1^i) \oplus \cdots \oplus (\mathcal{E}_l(y_Q^i) \odot f_Q^i) \oplus (\mathcal{E}_a(a_Q^i) \odot f_Q^i)] \in \mathbb{R}^{2dQ}, \\ \mathbf{z}^i &= [z_1^i, z_2^i, \ldots, z_Q^i]^T \in \mathbb{R}^Q,\end{aligned} \tag{2}$$

where f_j^i is 1 if we observed the answer and response for the jth question of the ith student and 0 otherwise, z_j^i is 1 if the jth question is responded to by the ith student and 0 otherwise, y_j^i is the response (correct/incorrect) of the jth question of the ith student (if observed, otherwise simply 0), a_j^i is the answer (A/B/C/D) of the jth question of the ith student (if observed), $\odot$ is element-wise multiplication, and $\oplus$ is the concatenation operator. Therefore, at the start of the question selection algorithm, $f_j^i = 0,\ \forall j$, and after selection of $m \leq n$ questions (and observing the responses), $f_j^i = 1$ for those m selected questions and only the features of these questions are active in $\mathbf{x}^i$. However, at any stage, $z_j^i = 1,\ \forall j \in \{t : q_t \in \mathcal{D}_{\text{tr}}^i\}$ and 0 otherwise.

We compute the context variable $\mathbf{w}^i(\Theta) = g(\mathcal{D}_{\text{tr}}^i, \{y_t, a_t\}_{t \in \Phi(\mathcal{D}_{\text{tr}}^i)}; \Theta)$ for student i as,

$$\begin{aligned}\mathbf{w}^i &= \text{NN}_2\Big([\mathbf{x}^i \oplus \text{NN}_1(\mathbf{z}^i)]\Big) := \text{NN}_2\Big(\mathbf{h}^i\Big), \\ \text{NN}_1(\mathbf{z}^i) &= \text{Dropout}(\text{ReLU}(\mathbf{W}_2(\text{Dropout}(\text{ReLU}(\mathbf{W}_1\mathbf{z}^i))))), \\ \text{NN}_2(\mathbf{h}^i) &= \text{Dropout}(\text{ReLU}(\mathbf{W}_4(\text{Dropout}(\text{ReLU}(\mathbf{W}_3\mathbf{h}^i))))),\end{aligned} \tag{3}$$

where Θ consists of $\mathbf{W}_{1,2,3,4}$, a set of weight matrices and bias that are omitted for simplicity. We compute the final output states $\hat{\mathbf{y}}^i \in \mathbb{R}^Q$ using another linear layer,

$$\hat{\mathbf{y}}^i = \mathbf{W}_5\mathbf{w}^i = [\hat{y}_1^i, \ldots, \hat{y}_Q^i] \in \mathbb{R}^Q,$$

and $\sigma\big(\hat{y}_j^i\big)$ (where σ is the Sigmoid function) represents the probability of answering question j correctly. The final meta loss function for student i is computed on the meta question set

$$\frac{1}{|\mathcal{D}_{\text{meta}}^i|} \sum_{j \in \mathcal{D}_{\text{meta}}^i} \ell\Big(y_j^i, \hat{\mathbf{y}}_j^i\Big).$$

2.3 Sample Selection Policy

We need to use a policy Φ to select n questions (in sequence) for observing the responses and answers. Our final solution uses a fixed policy using uncertainty-based active learning algorithm. We select question j_t^i at step t using the following equation,

$$j_t^i \in \underset{j \in \mathcal{D}_{\text{tr}}^i, f_j^i = 0, z_j^i = 1}{\arg\max} \quad \min\{\sigma(\hat{\mathbf{y}}_j^i), 1 - \sigma(\hat{\mathbf{y}}_j^i)\}.$$

Thus, at each step we select a question j_t^i that has not been selected yet ($f_j^i = 0$), is part of the support set ($z_j^i = 1$), and have the maximum uncertainty in its predicted output. Note that, after the selection of each question, the context-variable is updated using Eqs. 2 and 3. After observing the response and answer for question j_t^i, we set $f_j^i = 1$ and recompute $\hat{\mathbf{y}}^i$ using the updated input $\mathbf{x}^i$ and $\mathbf{z}^i$.

2.4 Optimization

We solve the problem in Eq. 1 using stochastic gradient descent (SGD) algorithm. Moreover, we also augment the objective with a loss term on the student's responses to questions that are selected and observed. The final objective becomes

$$\begin{aligned}\min_{\Theta} \sum_{i=1}^{N^{\text{train}}} \Big(\frac{1}{|\mathcal{D}_{\text{meta}}^i|} \sum_{t \in \mathcal{D}_{\text{meta}}^i} \ell\big(y_t^i, f(q_t^i, \mathbf{w}^i(\Theta), \Theta)\big) + \frac{\alpha}{m} \sum_{j \in \mathcal{D}_{\text{tr}}^i, f_j^i = 1} \ell\big(y_j^i, f(q_j^i, \mathbf{w}^i(\Theta), \Theta)\big)\Big) \\ \text{s.t. } \mathbf{w}^i(\Theta) = g(\mathcal{D}_{\text{tr}}^i, \{y_t, a_t\}_{t \in \Phi(\mathcal{D}_{\text{tr}}^i)}; \Theta),\ \forall i \in [1, N^{\text{train}}],\end{aligned} \tag{4}$$

where α (set as 0.1) is a small tunable hyper-parameter and m is the number observed responses so far (in our case less than or equal to $n = 10$).

Since the model parameters are randomly initialized at the start of the training, we do not introduce the question selection policy at the beginning. Moreover, for a fixed question selection policy, some questions might not be selected enough times for the model to be generalizable to questions in the meta set. Therefore, we use a simple heuristic during training: For the first K epochs, we use a random question selection policy Φ^{random}. After that for each iteration, we use the random selection policy with probability p and use the active learning-based selection policy Φ^{entropy} with probability $1 - p$. At the testing phase, we always use active learning-based selection policy Φ^{al}.

3 Experimental Results

Training and Testing. We do not use a fixed split of training and meta questions. Instead, at each epoch, we randomly partition the set of questions responded to by each students into training and meta sets with 80% questions and 20% questions, respectively; We observed overfitting with fixed training and meta sets. At the same time, we need to learn the best model based on validation/test loss; Random training and meta sets at every epoch usually results in high variance of the validation/test loss. Thus, we used 120 fixed seeds to create random question splits and fix them on the local test set. At testing time, we compute performance across these 120 seeds for each student to get a robust estimate of the model's performance. We use use 95% of the students in the training set for training and the rest of the students for local testing.

Network Architectures and Hyper-parameters. We use the Adam optimizer [5] to optimize the model parameters with learning rate $2e - 5$. We mix uncertainty sampling-based question selection with probability $p = \{75, 66.67\}\%$ with random question selection with $\{25, 33.33\}\%$ probability and set the number of initial exploration epochs to be $K = 50$. We train the models for 500 epochs and save the best five models based on average test accuracy on questions partitions for students in the local test set. We set the output dimension for $\mathbf{W}_3, \mathbf{W}_4$ as 1024 and fix the output dimension for $\mathbf{W}_5$ to the number of questions $Q = 948$. We set the output dimension for $\mathbf{W}_1, \mathbf{W}_2$ as 128 and $\{256, 1024\}$, respectively. We set the dropout rate from $\{25\%, 50\%\}$.

Extension. We also experimented with augmenting tree-structured subject information provided for each question. We decided to use only the most fine-grained (level 3) subjects, i.e., subjects that do not have any descendants in the tree for each question. We augment the subject information in the input state as,

$$\mathbf{x}^i = \Big[\Big(\big(\mathcal{E}_l(r_1^i) \oplus \mathcal{E}_a(a_1^i) \oplus \mathcal{E}_s\{s_1^i\}\big) \odot f_1^i\Big) \oplus \cdots \oplus \Big(\big(\mathcal{E}_l(r_Q^i) \oplus \mathcal{E}_a(a_Q^i) \oplus \mathcal{E}_s\{s_Q^i\}\big) \odot f_Q^i\Big)\Big] \in \mathbb{R}^{3dQ},$$

where we use another embedding module $\mathcal{E}_s$ to embed multiple subjects tagged with a question (for simplicity, we used the summation of individual subject embeddings). Initial local evaluation with subject information provided a marginal improvement of $0.01 - 0.03\%$ on predictive accuracy.

Results and Discussion. One of our key observations is that learning with limited samples results in a high variance in the results. We would want a robust model that performs well across most of the random partitions (120 in our case). Although we selected a model based on average performance across multiple partitions, we observed the best model in one partition may not perform well in other partitions. We believe the low number of students in the test set might be one of the reasons behind this.

Nevertheless, learning a robust model from limited observation for each student (irrespective of the number of students) remains a key challenge in this framework. We observed that the actor-critic algorithm [9] performs similarly to active learning-based methods in our preliminary experiments with a larger test set (20% students). We believe that learning a policy (using reinforcement learning) might improve results than a fixed active learning algorithm with more training and enough hyperparameter tuning.

Acknowledgement

This work is supported by the NSF under grant no. IIS-1917713.

References

[1] H. Chang and Z. Ying. Nonlinear sequential designs for logistic item response theory models with applications to computerized adaptive tests. *The Annals of Statistics*, 37(3):1466–1488, June 2009.

[2] A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-adapted Interaction*, 4(4):253–278, Dec. 1994.

[3] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[4] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339, 2020.

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations*, May 2015.

[6] W. J. Linden, W. J. van der Linden, and C. A. Glas. *Computerized adaptive testing: Theory and practice*. Springer, 2000.

[7] F. Lord. *Applications of Item Response Theory to Practical Testing Problems*. Erlbaum Associates, 1980.

[8] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[9] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[10] G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. MESA Press, 1993.

[11] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[12] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[13] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.