Oversea Internship Final Report

SIDA GAO
Supervised by Prof. Emma Brunskill
Reinforcement Learning and Education Lab, CMU CSD
August 28, 2016

Abstract

This is the final report of my summer internship in CMU. It is *NOT* a formal academic paper, since all of the projects pictured in this report are still works in progress. Therefore, some parts of the write-up won't be very formal, especially the reference part, which would only appear as footnotes (and works that are not that relevant would not be listed at all). This report assumes the readers have a general understanding of traditional AI, machine learning, deep learning and decision making, but will provide some backgrounds in reinforcement learning (RL), and educational data mining (EDM).

The main project for this summer is Deep Batch RL, in which I work with a master student, aiming to compare the generalization ability of model-based and model-free RL methods which both adopt deep learning models. Our original hypothesis believes that model-free methods should outperform the model-based ones, however, our emperical results suggest the opposite. The side project, in which I take the lead, focuses on extracting prerequisite relationships in educational data. This project also adopts deep learning models, which hold an advantage of dicovering complicated latent relations in the data over traditional methods. The last project I take part in is Adaptive Fraction, which aims to find robust tutoring policy with offline educational data collected from Fraction, an online tutor system. A robust policy is one we can expect it to be at least as good as our evaluations with simulated students. This project is currently at a rather preliminary phase and we are still working on designing and implementing the experiments.

1 Introduction

Artificial intelligents could be roughly divided into two classes: those that are trying to understand the world (classification, clustering, etc.) and those that are trying to improve the world with actions (decision making, planning, etc.). Reinforcement learning aims to do both, since it's trying to make optimal decisions in an unknown world. By doing some episodes of trials in the world, an RL agent collects trajectories of states (i.e. observations), rewards and actions. With these trajectories, the RL agent can pick the best action in any observed state, which would gain him the most reward. RL differs from decision making methods like Marcov decision processes, because MDPs have a perfect understanding and observation of the world, while RL agents know nothing about the transition model of states or the reward model (i.e., the rewards related to each state). In real life situations, the actual states are often not observable, and the agents would also need to learn to infer the states with observations.

Education, or to be more specific, intelligent tutor, is a perfect case where RL methods can be applied. We can easily draw some parallels between tutor systems and RL agents. The actions are the various tutoring materials that could be presented to students, while the rewards could be students' performance in tests. The world is also unknown: the transition model of students' knowledge states

(i.e. how students would digest the materials) is unknown, and the reward model is also unknown (i.e. how student with a certain knowledge state would perform in a test). In education, we are also dealing with a partially observable environments, since we won't be able to directly see the knowledge state for students, and we can only have a belief state infered from observations of the students.

Education also poses an even larger challenge than other RL tasks. In conventional RL tasks like robotics, the robot can run trials in the real environments as much as we want, since in most cases the worst we can expect is merely a broken mechanical arm resulted from tripping over a stone. In education, it won't be ethical to collect online data on real students with an agent that's still learning from trials and errors. A crappy work-in-progress would do irreversible harms to real kids in this practice, and the things that could be broken would be the kids' future. Therefore, doing offline policy evaluation with trajectories collected from real world tutor systems (in which runs policies either designed by expert humans or well-tested AI agents) is an important track of research in educational RL.

Prof. Emma Brunskill's group, in which I interned for the summer, lies its interest in RL and education. We have shown that these two areas are intrinstically related. Our group covers a wide spectrum of research areas, from RL theories (machine learning) and traditional AI, all the way to more application oriented tracks like data mining and human computer interaction (in education). Given the special, yet not unique challenge of education described previously, our research projects are mostly focused on offline policy evaluation or planning. Educational data mining, which focuses on data collected from intelligent tutoring systems, is one of the may side interests of our group. Several distinctive characteristics of EDM include dealing with sequential data, and a special focus on discovering the latent hierarchical structure of data items (i.e. prerequisite structure of knowledge components). Deep learning, while has been widely adopted in various areas like computer vision, is still an emerging methods in RL and EDM, where only only a handful of previous works could be found. It is also a new area for the group.

In this section, we have briefly described the backgrounds and motivations of the projects. The following sections are organised as: section 2 described a deep learning model which my work in all 3 projects are based on; section **??**, section **??** and section **??** discusses the big pictures of the three projects, where we are and what part I play in them. At last, section **??** jots down some other aspects of the internship basides work, and section **??** summarizes the 10-week internship.

2 Deep Knowledge Tracing

2.1 Knowledge Tracing Going Deep

Knowledge tracing is a common task for educational data mining, with a goal towards accurately model the students' knowledge acquisition process. The conventional method, Bayesian Knowledge Tracing (BKT), keeps a binary state for each skill (i.e. knowledge component, or concept), either known or unknown. When a skill is not learned, by doing exercises on the skill, the student may have some probability to transition to the learned state. When answering a question on a certain skill, the student could get it wrong even when he already knows the skill (i.e. there exists a slip rate). And reversely, a student can get a question correct with luck, without knowing the skill, which corresponds to a guess rate. There have been various extentions to BKT, like modeling the forgetting process, and collapsing similar skills into one (since BKT tracks each skill separately, this kind of practice is trying to address the relations between skills).

There has been a quite influential work published in 2015, *Deep Knowledge Tracing* [1] (DKT), which adopts an LSTM to model students' knowledge acquisition. Retrospectively speaking, LSTM is basically an upgraded version of BKT. Instead of having one binary state representation for each skill, LSTM uses a long vector of real numbers (which is normally more than the number of skills) to represent the knowledge state. By doing this, LSTM could capture complicated (and unfortunately, not very interpretable) relationships between different skills. The four gate layers in LSTM also closely resemble the key parameters in a BKT. The output gate, which reads out the hidden state vector to get an output (i.e. belief state on each skill), is a counterpart of slip and guess rate. The update gate, which scales the update value from the input, is like the acquisition rate. And the forget gate is literally modeling the forgetting of a knowledge component. In a nutshell, LSTM just upgrades the model of these processes from a single real number, to a layer of fully connected neural network. Therefore, we wouldn't be surprised to see DKT vastly outperforms BKT.

DKT can be simplified as a function approximator which takes a student's practice history (sequence of exercises he did, and whether he got each exercise correct or not) as input, and outputs the predictions (belief states) of the probability of getting each skill correct if presented as the next question.

2.2 Implementing DKT

DKT is a good modeling method for the student, i.e., a learned transition model of hidden knowledge states. This could be useful in various parts in our group's projects. Therefore, the first thing to do is implementing the model.

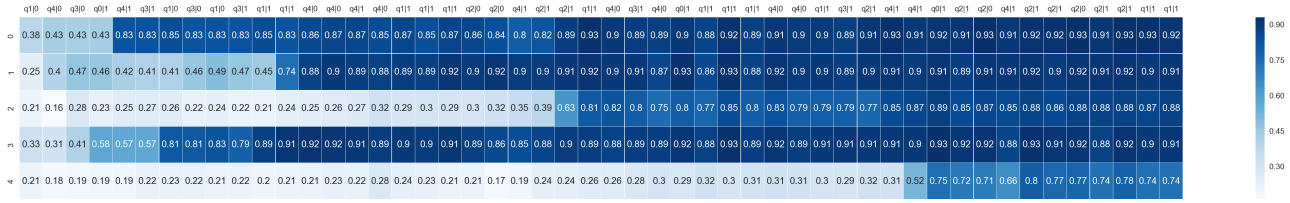
Although there has already been several published DKT implementations, besides being a great opportunity to sort out all the subtlety like sequence padding and dropouts in LSTM, there are several other advantages of having our own implementation. First of all, our implementation is in Python, under the Tensorflow framework. Python is a friendly language for machine learning and data mining practices with its various powerful packages, and Tensorflow is a more flexible deep learning frame work, both of which provide convenience not presented in previous implementations. More importantly, our ultimate goal of DKT doesn't stop at getting a prediction metric on the test dataset, but to adopt it to do planning and data mining, thus calling an off-the-shelf LSTM cell from the framework won't work for us, since we won't be having access to the inner state of the LSTM that way. We need to implement the LSTM from scratch to better accustom to our goals. At last, since we have the full access to the model's structure, we made a small step forward of DKT by connecting the readout layer to the trainable initial state vector, which enables our DKT to predict the first answer without any previous input (which is a feature not presented in the original DKT implementation).

2.3 Sanity Checks for DKT

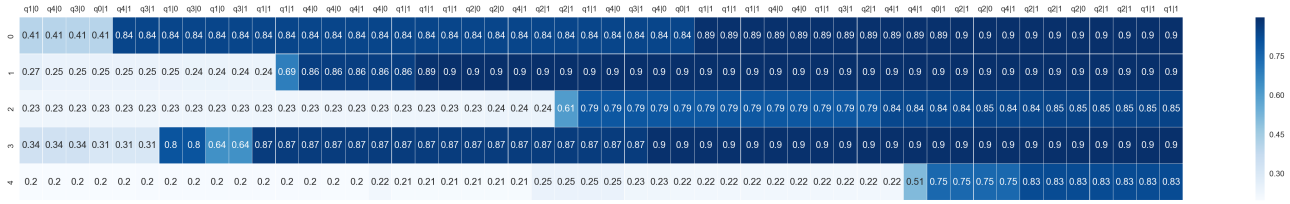
The biggest challenge of implementing a deep learning model is verifying the model. It requires little tuning (which is a great characteristic of using Adam optimizer in stochastic gradient descent) to get an AUC prediction performance comparable with previous works on the test set, however, we need more than a single AUC number to convince us that the model can be trusted.

Besides conventional checks like printing out or plotting several crucial variables/metrics, we did a more convincing sanity check. The goal is to test if DKT can fully recover a BKT model. First, we did a BKT simulation with 5 skills, each has its unique set of BKT parameters. As an extention to vanilla

[1] *Deep Knowledge Tracing*, Chris Piech et al, NIPS 2015.



(a) DKT result on student 9999.



(b) Oracle BKT result on student 9999.

Figure 1: An example of the knowledge tracing visualization of DKT and BKT on one of the simualated student trajectory. The row labels are student ids. The column labels, $qi|j$, are student actions in each time step: the question is on skill i, $j = 1$ when the student answers correctly, otherwise $j = 0$.

BKT, we add several prerequisite relations into the simulation: skill 0 and 1 are prerequisites for skill 2, while skill 2 and 3 are prerequisites for skill 4. We simulate such relations in a rather simplified manner: the student's acquisition rate of a skill would stay 0 until he learns all the prerequisites.

BKT model is actually a twi-state HMM model, with acquisition rate corresponds to transition model, slip and guess rate corresponds to sensor model. Therefore, we can easily construct an oracle BKT inference model, which knows all the hidden parameters of BKT, and can infer the student's hidden knowledge state with the simulated trajectories as observations. DKT can achieve the same AUC, 0.81, with this oracle BKT inference model. Again, only a AUC number won't be enough (there are millions of ways to accidentally get a high AUC). We computed the MSE (mean squared error) of the step-by-step belief states predicted by both models on the test set and got 0.003, which is a small number, but we still don't know if it's small enough. Therefore, we plot the knowledge tracing process given by the two models and compare the trends. From **??**

In addition, we did a more novel sanity check, in which we reorder the train or test set in various ways and observe the consequenct AUC performance. This yields some interesting results and end up being a new project, which is ellaborated in section **??**.

Accurate BKT.

follow ups of DKT, traditional outperform

DKT no domain knowledge is required

Data quality issues