Efficient Human Inputs in Learning From Example

Mohammad Nasirifar
Department of Computer Science
University of Toronto
farnasirim@cs.toronto.edu

Abstract—I tackle the problem of teaching specific expert level skills to learners. I leverage expert inputs for creating the learning material. The main contribution of this project is formulating this problem as a contextual bandit problem where different arms correspond to different hints/solutions for a particular learning task. Each task corresponds to a state and learner's previous skills define the context. Furthermore I provide a solution to a simplified case formulated in this way using multi-armed bandits and Thompson sampling.

I. Introduction

Learning based on examples has been proved to be an effective way of improving one's abilities in a particular skill. This way of learning/teaching has been employed in chess education for a very long time: learners receive puzzles of real games played by grand-masters and are asked to come up with or rationalize the original move of the grand-master.
This is widely employed in chess since the chess games are labeled with the correct approach by definition and the rationale behind a move can be figured out by the learner or demonstrated to her by a teacher.
There are however education scenarios in which the actions, labels, and/or explanations are not this clear. Take teaching programming as an example: there are lots of ways to accomplish a certain task, many of which are sub-optimal. In this project I have tried to create a framework to allow us to employ the aforementioned "puzzle-based learning" in environments where obtaining puzzle explanations is more "complicated" than the chess teaching scenario.

II. Related Work

The main motivation for this work comes from "learnersourcing" problem explanations idea described in AXIS[1], where the authors leveraged reflection and explanation by learners' about their own answers to both improve their learning performance, and furthermore, to generate and curate hints that can be used to help other learners understand a certain subject.
The difference between this project and AXIS comes from the fact that in AXIS it is assumed that some students are capable of generating expert-grade hints for a subject they are studying. This can be true in more controlled environments such as academic ones, but can fall short in scenarios where more expertise is needed to create valuable hints. I believe that in these scenarios expert knowledge can be leveraged effectively to yield better results. This idea originates from the fact that we can imagine an extreme scenario in which we have access to one expert (professor) per learner (student), who would do the hint generation step instead of the student, as a consequence of which the hints will be of more quality. The obvious shortcoming in this approach however is that our resource of domain experts is quite sparse and we must be quite strict about the exact workload that we demand from them.

This brings us to the next important background for this project that targets efficient human intervention in human in the loop systems [2]. They model the human in the loop systems as Markov Decision Processes. The human (expert) resource constraint is resolved by making the human intervention process on-demand: a human will inform the system that she is willing to provide a new action for the MDP (create a new hint for some problem). That is, the platform moderator is free to use whatever amount of resource that she decides to improves the system at any time. The system responds with a state that is expected to yield the highest return if the human were to add the new action there. Under certain assumptions, a strategy is formulated to maximize the Expected Improvement (EI) over time.
Unfortunately important hypotheses from this work depend upon the uncertainty in state changes imposed by the MDP, which is something that does not naturally fit the scenario of providing hints (actions) in different states (problems) since the core element of this problem is the uncertainty in the action rewards and the emphasis on the uncertainty in state changes is not very useful here. This is expected, as a general multi-armed bandit problem is equivalent to a single-state MDP.

III. Motivation

The main motivation for this project has been improving the process of training new programmers who have recently joined a company/project. This is a recurring problem in software engineering which has forced companies to allocate resources to developing their own training material. In this task the newcomers can hardly generate hints/explanations that is useful for others, as deep knowledge of different part of the project is crucial for this, regardless of the expertise of the new programmer

in the fundamental programming skills. I will try to fit the to formulate this scenario using the model and it can be seen that no semantic dependency is formed between the two and other problems can be mapped to this model conveniently by defining the required model specific parameters and metrics.

IV. Idea and Modeling

We define different entities in the model:

1) Problem: An encapsulated unit that can be passed to a learner that she would try to solve. In the programming scenario, a problem consists of issue texts, tasks, requirement documents, and anything else which can define a unit of work for an already expert member of the team. The learner is then asked to try to carry out the task as the expert would. The answer to the problem is the patch that the learner creates by writing code. These "problem" are easy to generate in this scenario: teams store the history of their projects and the resources resulting in a tasking being done can be extracted from the team's issue tracker, documents, VCS, etc.
2) Answer: An explanation of what the solution to a particular problem must ideally look like. In our scenario these can consist of the actual patch files and the documentation for the design decisions. These are generated by the experts to help the learners understand the underlying subjects of the problems.
3) Learner Context: These are the context variables that define a certain learner. For programmers, this can correspond to their familiarity with different fields in software development.

The human in the loop part of the system is modular and detachable. Without human inputs, the system functions like a normal system that approximates the solution to a (contextual) multi-armed bandit problem. This phase consists of choosing a hint for a learner visiting a particular problem, asking for her feedback about how helpful she things the hint has been, and updating the underlying parameters of the algorithm. We can operate any bandit algorithm for this part.

The part that is of our interest however, is interacting with the expert to help her generate helpful hints. In this scenario, the expert must be given a state (problem) to generate a new hint for, in such way that maximizes the cumulative gain of the system over time.

We assume that the learners will eventually go through all of the problems and will do so uniformly and that the expert only asks to generate hints only when the state of the bandit algorithm is "stable", meaning different actions have had enough chance to be selected. This is to ensure that no particular problem would falsely overshadow other problems because they haven't had enough visitors, when we are choosing a target state for the expert. The first assumption is justifiable in the programming scenario as new programmers usually go through step by step tutorials which require them to complete a set of tasks one after the other. The second assumption will eventually be satisfied given the first assumption. Furthermore we take into consideration that multiple experts are involved in the hint generating process, each of whom possibly contributes to the solutions of a specific subset of the tasks. We assume the experts to have different hint generating abilities.

We define an expert e's ability to generate hints as follows:

$$Ability_e = E[r \mid F_e]$$

Where r is the reward gained from presenting a hint to a learner, and F is the feedback data that we have received for this expert's hint:

$$F_e = \{feedback \in hint \mid hint \in H_e\}$$

H_e being the hints that e has created. Now based on the above definitions, we define the optimal problem choice for the new hint H_{new} to be:

$$p = \arg\max_p \{Pr(H_{new}\ is\ chosen\ from\ \{H_{new}\} \cup \{H_i \mid H_i \in p\})\}$$

Where H_{new} is dependent on $Ability_e$.

To be more concrete, suppose that the experts' abilities and the hint rewards are distributed normally:

$$r_e \sim \mathrm{N}(\mu_e, \sigma_e^2)$$

for expert e, and

$$r_h \sim \mathrm{N}(\mu_h, \sigma_h^2)$$

for hint h.

Therefore the answer the optimal choice for the problem that expert e is going to generate hint for would be:

$$p = \arg\max_p \{Pr(H_{new} > max\{H_i \mid H_i \in p\})\}$$

We can consider each problem separately and solve for each probability value and get the maximum over all of them. In case of $|\{H_i \mid H_i \in p\}| = 1$ we are left with calculating $Pr(H_{new} > H_{old})$. For two random variables distributed normally we have:

$$X_1 \sim \mathrm{N}(\mu_1, \sigma_1^2), X_2 \sim \mathrm{N}(\mu_2, \sigma_2^2)$$

$$\mathrm{P}(X_1 > X_2) = \mathrm{P}(X_1 - X_2 > 0) = 1 - \mathrm{P}(X_1 - X_2 \leq 0).$$

$$\mu := \mathrm{E}(X_1 - X_2) = \mu_1 - \mu_2$$

$$\sigma^2 := \mathrm{Var}(X_1 - X_2) = \sigma_1^2 + \sigma_2^2.$$

$$\frac{X_1 - X_2 - \mu}{\sigma} \sim \mathrm{N}(0, 1),$$

Therefore the problem is reduced to calculating the probability of one random variable being bigger than a particular value. Also in $Pr(H_{new} > max\{H_i \mid H_i \in p\})$ every H_i can be considered independently from others, allowing us to extend the two variables case to solve the original problem when $|\{H_i \mid H_i \in p\}| > 1$.

V. Implementation

For simplicity, I have only taken into account a boolean positive/negative feedback to a provided hint and therefore have used Beta distribution to model the rewards of the hints and agents. Furthermore I have considered only a single expert in the system. For calculating the optimal problem based on the above policy, I have used the less efficient Markov Chain Monte Carlo instead of the closed form.

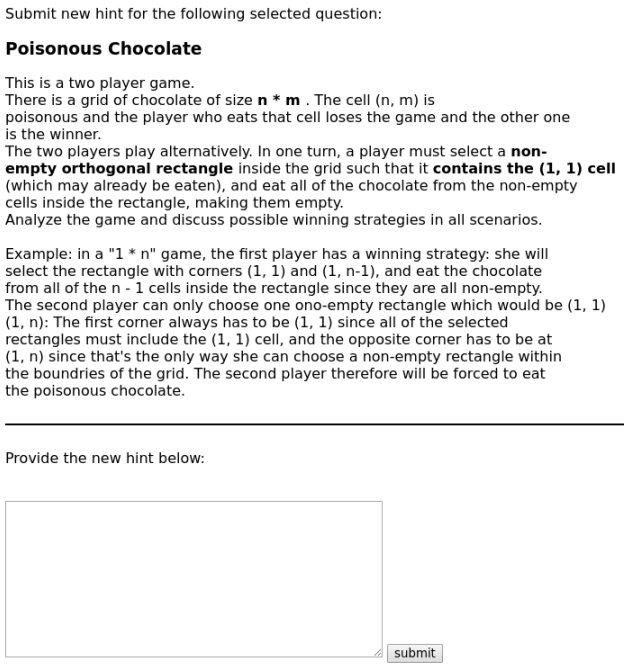
VI. Evaluation

To be able to test the approach on a more accessible audience, I have opted to using algorithmic problems instead of programming tasks. I provide the learner with an an algorithmic problem that I ask them to think about. After they do they would request for a hint. The hint will be selected using Thompson sampling from the current hints in the system. After that they provide a boolean feedback to signify whether or not they think the hint has been useful. The expert can query the system to ask which problem would make the most impactful choice if she were to generate a hint, and after the response, the expert can add the new hint to the problem.

VII. Empirical Results

I implemented a simplified version of the algorithm, added 3 computer science questions to the platform, and asked 7 computer science student to follow through with the flow of the software. The code is at github.com/farnasirim/iai-project and a online hosted version can be visited at http://learn.farnasirim.ir. At the very beginning, after entering the problems, when one queries the system at http://learn.farnasirim.ir/questions/add_hint/ to find out where it would be better to add hints, if there exists a question without a hint, the system will ask for a hint for that problem.

If all of the questions already have a hint, the method that I introduced above is used to choose a question: find where the hint would be most impactful. In this case, the hint to the first question is actually wrong and most of the people noticed that. That has resulted in a number of downvotes for the first one. The second problem is fairly easy and the solution is correct and verbose. Most of the people found that useful. The third problem is a bit complex and people with lack of certain backgrounds may not understand it very easily. There was a mix of positive and negative feedbacks for that one.


Submit new hint for the following selected question:

Poisonous Chocolate

This is a two player game.
There is a grid of chocolate of size **n * m** . The cell (n, m) is poisonous and the player who eats that cell loses the game and the other one is the winner.
The two players play alternatively. In one turn, a player must select a **non-empty orthogonal rectangle** inside the grid such that it **contains the (1, 1) cell** (which may already be eaten), and eat all of the chocolate from the non-empty cells inside the rectangle, making them empty.
Analyze the game and discuss possible winning strategies in all scenarios.

Example: in a "1 * n" game, the first player has a winning strategy: she will select the rectangle with corners (1, 1) and (1, n-1), and eat the chocolate from all of the n - 1 cells inside the rectangle since they are all non-empty.
The second player can only choose one ono-empty rectangle which would be (1, 1) (1, n): The first corner always has to be (1, 1) since all of the selected rectangles must include the (1, 1) cell, and the opposite corner has to be at (1, n) since that's the only way she can choose a non-empty rectangle within the boundries of the grid. The second player therefore will be forced to eat the poisonous chocolate.

Provide the new hint below:

submit

This results in the system to ask for a hint for the first problem (Poisonous Chocolate) almost always at the current state. This will notify the instructor that there is room for improvement in the hint(s) to the first problem, and when she provides a better one, it will receive more positive feedback, which would shift the "most impactful" problem for a hint at the given time to another problem.

VIII. Shortcomings and Future Work

I have not taken into account what happens when we have create new problems in the analysis. Intuitively, the semantics of the model make sense and I would expect everything to work fine in that case too, but I have not gone through any formal analysis.

I think the experts' ability can be captured more strongly. It looks like that there is a lot of "linearization" happening when we combine the whole history of the feedbacks to an experts' hints together and we should be able to do better to "lose track of" less data in the process. One approach that naturally comes to mind in the bandits and Bayesian inference context is modeling the ability function as a probability distribution of probabilities.

References

[1] J. J. Williams, J. Kim, A. Rafferty, S. Maldonado, K. Z. Gajos, W. S. Lasecki, and N. Heffernan, "Axis: Generating explanations at scale with learnersourcing and machine learning," in Proceedings of the Third (2016) ACM Conference on Learning @ Scale, ser. L@S '16. New York, NY, USA: ACM, 2016, pp. 379–388. [Online]. Available: http://doi.acm.org/10.1145/2876034.2876042

[2] T. Mandel, Y.-E. Liu, E. Brunskill, and Z. Popovic, "Where to add actions in human-in-the-loop reinforcement learning," in AAAI, 2017.