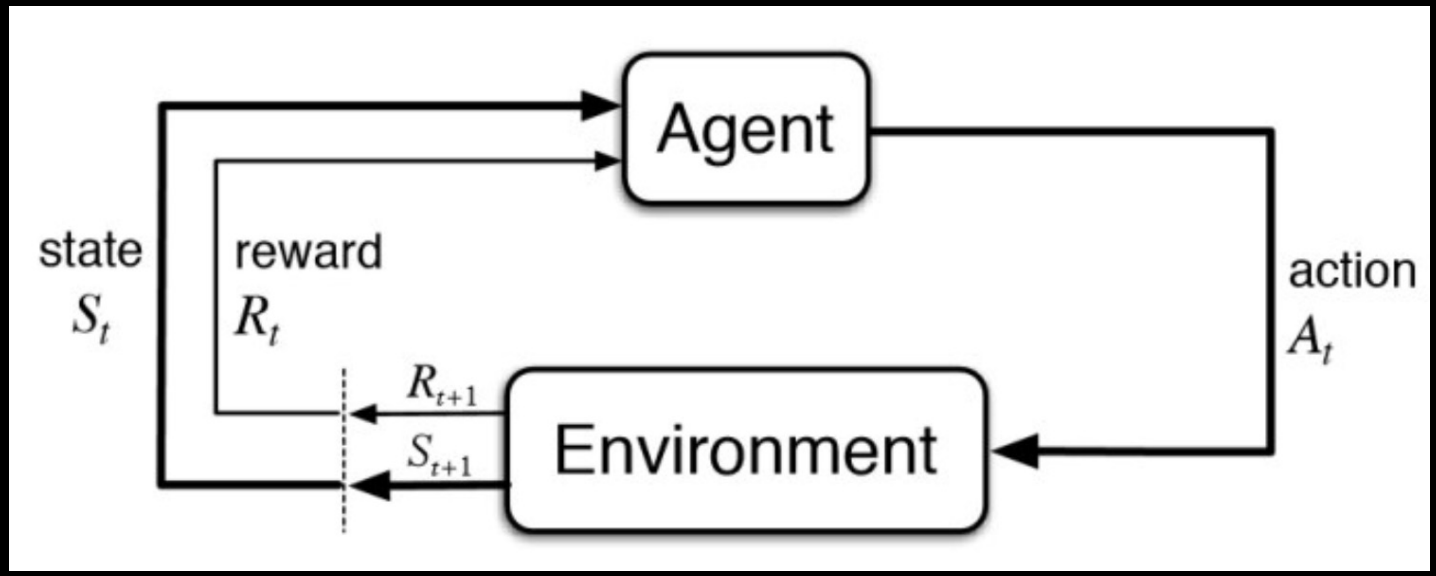
VIP-ITS: ITS-RL

INTELLIGENT DECISION MAKING

REINFORCEMENT LEARNING

- Reinforcement Learning is a machine learning technique used for decision making. This is done by modeling an intelligent environment where one of more agents can take actions and earn rewards, this feedback from the actions is used to learn the optimal sequency of decisions that maximize the cumulative reward.



Q-LEARNING ALGORITHM

- Q-learning algorithm is an off-policy temporal difference (TD) learning algorithm used for reinforcement learning
  - Off-policy: The agent learns using a greedy policy, but then chooses actions or behavior using other policies like epsilon-greedy selection.
  - Temporal difference: A class of model-free reinforcement learning methods which learn by bootstrapping from the current estimate of a value function. The agent interacts with the environment, and repeatedly updates the estimates based on rewards till the value converges
- The value update rule for Q- learning is given by:

$$Q_t(s,a)=Q_{t-1}(s,a)+\alpha(r+\gamma \max(Q_{t-1}(s', \text{all a})-Q_{t-1}(s,a))$$

α = Learning rate; γ = Reward discount factor; r = reward, s = state, a = action, s' = next state

LEARNING PROCESS

Set learning rate(α), epsilon (ε), gamma(γ), decay(d)

Initialize Q(S,A) = 0

For each episode:

Reset environment and observe state s

Set done = false

Loop till done

Select action a from state s with ε-greedy policy in Q

Take action a, observe s', r, done

Set $delta = \alpha[r + \gamma(\max(Q(s':)) - Q(s,a)]$

Update Q(s, a) = Q(s, a) + delta

Set current decay, $d = d * d$

If Epsilon decay, $\varepsilon = \varepsilon * d$;

If Learning decay: α * d

PARAMETER TUNING

- We have a number of variables need to be tuned depending on the problem
  - Epsilon (ε): If ε is 1, we try to explore the environment by choosing a random action, if ε is 0, we use a greedy approach and select action with maximum Q value. An ε-decay approach allows us to start by exploring a lot and slowly increase exploitation of the learnt information
  - Learning rate (α): This is the amount of Q value updated in each step, At the start of learning we want to larger updates to Q values, so α should be high, you can either keep this constant or slowly decay it to stabilize Q value updates and we will converge to an optimal policy.
  - Discount factor (γ): This is essentially how much the agent values future rewards as opposed to immediate rewards. If γ is 0, future rewards are not valued at all, and if γ is 1, then future rewards are valued as much as immediate rewards. Usually, a value close to 1 is chosen

ITS APPLICATIONS

- Chatbot – The chatbot provides an automated question answer system for both course content as well as logistics. Currently the intent is to find the most relevant context to provides as answers. This can be further extended by further defining the problem as a content recommendation system. Here the states are the questions, actions as top n content recommendations from the prediction model, and rewards would be the user feedback on its relevancy.
- Tutor JS – This systems provides human readable feedback during coding assignments. Similar to the Chatbot, RL can be used to identify most relevant hint (or set of hints). The state can be the coding assignment and the error types, actions can be the top n potential hints and the feedback/rewards can either be specific user feed or the time it takes the user to resolve the error.
- Intelligent Tutoring System – This refers to the overall ITS system which can modelled as a multifaceted environment which connects the different student experiences (piazza questions, chatbot interactions, answer questions, lab assignments etc.). In this case the intent would be to find the optimal policy to navigate the system to gather most knowledge.

INTELLIGENT QUESTION REVIEW

WHY

- In order to test out the validity of using reinforcement learning within any of the described scenarios, a simple setup was used to find the optimal policy for maximizing score when reviewing assignment questions

HYPOTHESEs

- It is possible to model a question review system utilizing existing student performance data to bootstrap a RL environment which can be used to identify an optimal policy for answering questions

ASSUMPTIONS AND SIMPLIFICATIONS

- A sequential progression from the review questions from first to last chapter improves student knowledge
- Students' performance improves as they spent more time reviewing questions.. This makes selecting actions with larger Q-values in latter stages of learning (exploit phase) realistic.
- Compute type questions are harder than matching or multiple-choice questions

MODEL SETUP

- State: Tuple containing chapter and question. Example: (1,32)
- Actions: Combination of score and speed, where scores can be ‘all’, ‘high’, ‘medium’, ‘low’ and ‘none’ and speeds can be ‘fast’, ‘average’ and ‘slow’. Example: “high_fast”. “low_slow”
- Transition probabilities: A model free algorithm is used, so there is no explicit transition probability
- Reward: Based on score; ‘all’: 10, ‘high’:7.5, ‘medium’:5.0, ‘low’:2.5, ‘none’:0. An additional score multiplier of 1.5 is applied for ‘compute’ typed questions

ENVIRONMENT SETUP

- Student data is extracted from the stats_nnnn and scores are translated into categories as 0=None, 1-33=Low, 34-66=Medium, 67- 99=High, 100=All
- This data is then grouped by chapter, question and score range type, and duration is aggregated and counted as well as split into quantiles (25%, 50%, 75%). The quantiles are mapped to speed levels of slow, average, fast
- The count per group is used to determine the action probability for each chapter question and score type, with all three speeds for a score category assigned the same probability
- The action probabilities are used during exploration phase when picking actions. In the second setup the success action probabilities as increased based on question performance to date
- The rewards are assigned based on the score category (None=0, Low = 2.5, Medium = 5.0, High = 7.5, All=10)

ENVIRONMENT RULES & CONSTRAINTS

- Reward shaping is used to incorporate broader and deeper review experience
  - Compute question types gets 1.5 times the rewards points as match or multiple choice questions
  - A transition to the next chapter assesses an additional reward
- Speed (efficiency) and coverage within the question review process is incorporated through constraints such as:
  - A configurable question count limit of 25 is required prior to moving to next chapter
  - Compute question types are double counted vs match or multiple choice questions
  - Limits are placed on total review duration and number of attempted questions

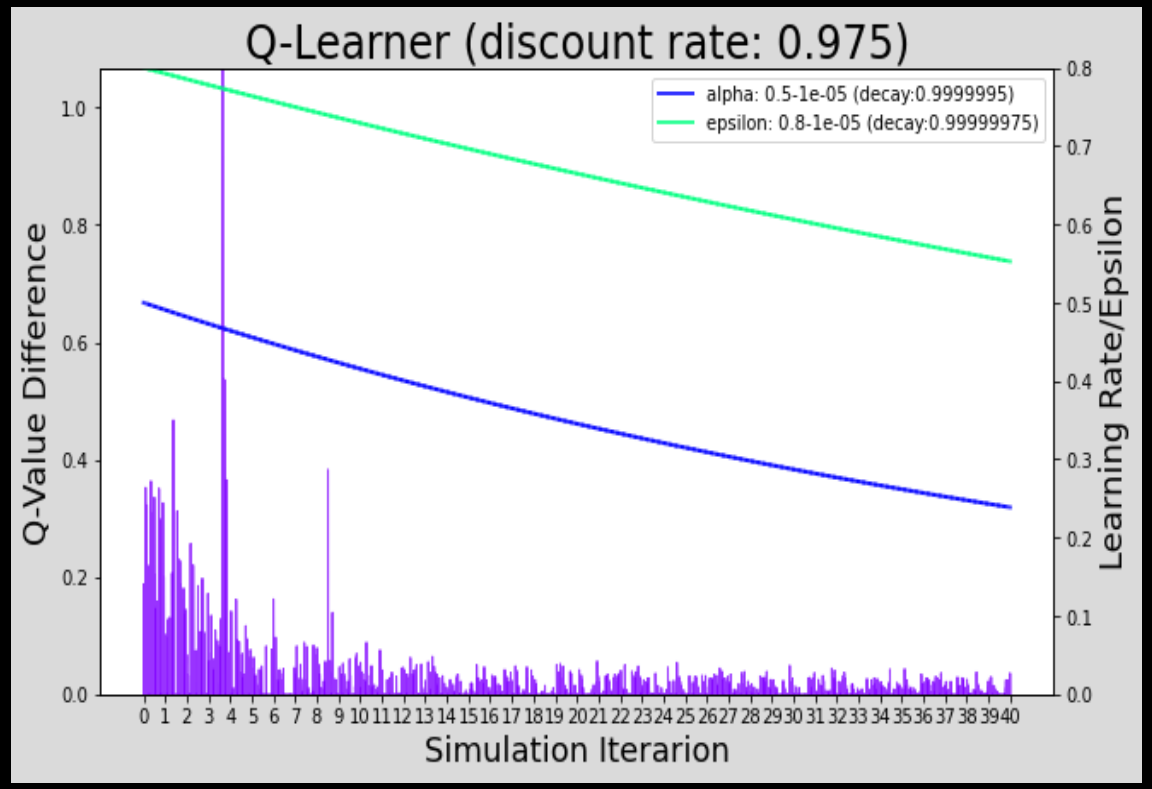
ENVIRONMENT BEHAVIOR

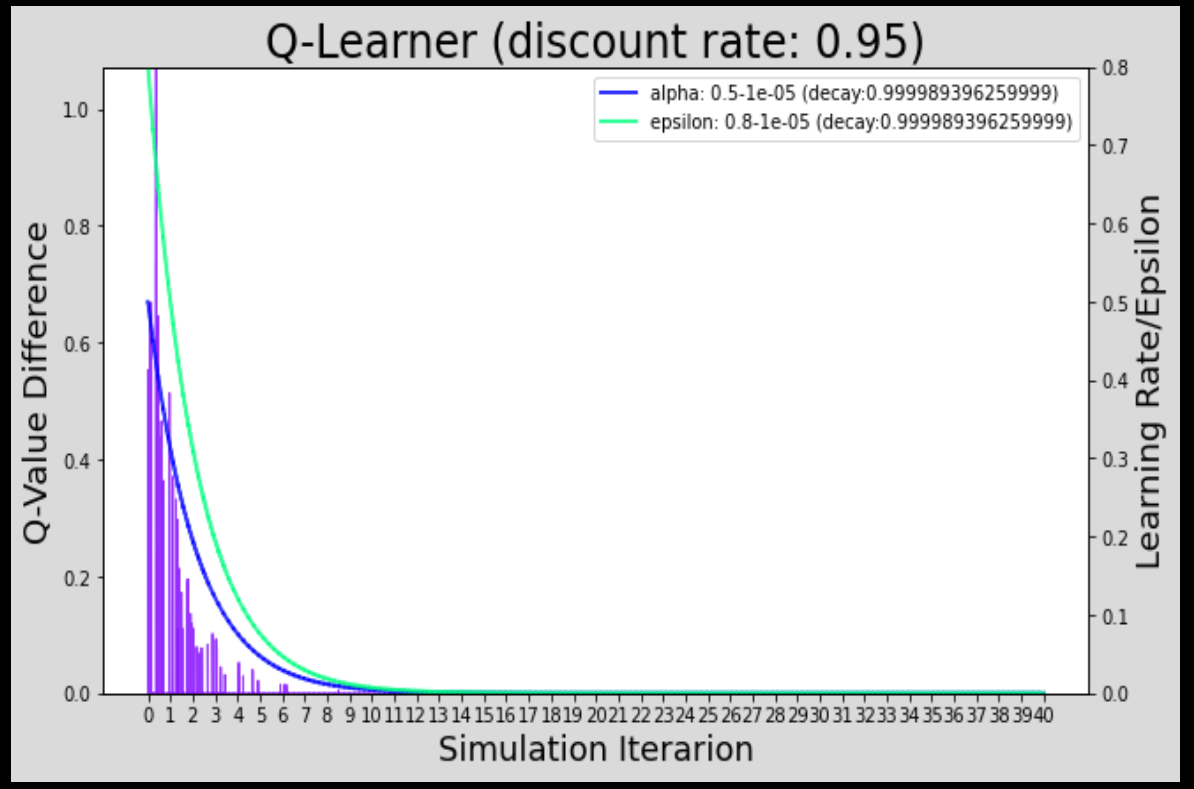
- Since we are using a model free algorithm, we need the environment to work like the real world, so environment behaves as follows:
  - At initialization, the review starts at Chapter 1 and then proceeds to succeeding chapters
  - The next question to attempt is chosen randomly, but questions cannot be reattempted
  - For a particular chapter and question pair, the available actions are limited to the score ranges seen in real life during exploration phase. So if a question had only low or high scores, actions related to none, low, medium and all are not allowed.
  - The switch to the next chapter happens when either the weighted or simple question count exceeds the maximum allowed per chapter
  - Similar to questions, chapters cannot be repeated

ENVIRONMENT VARIATIONS

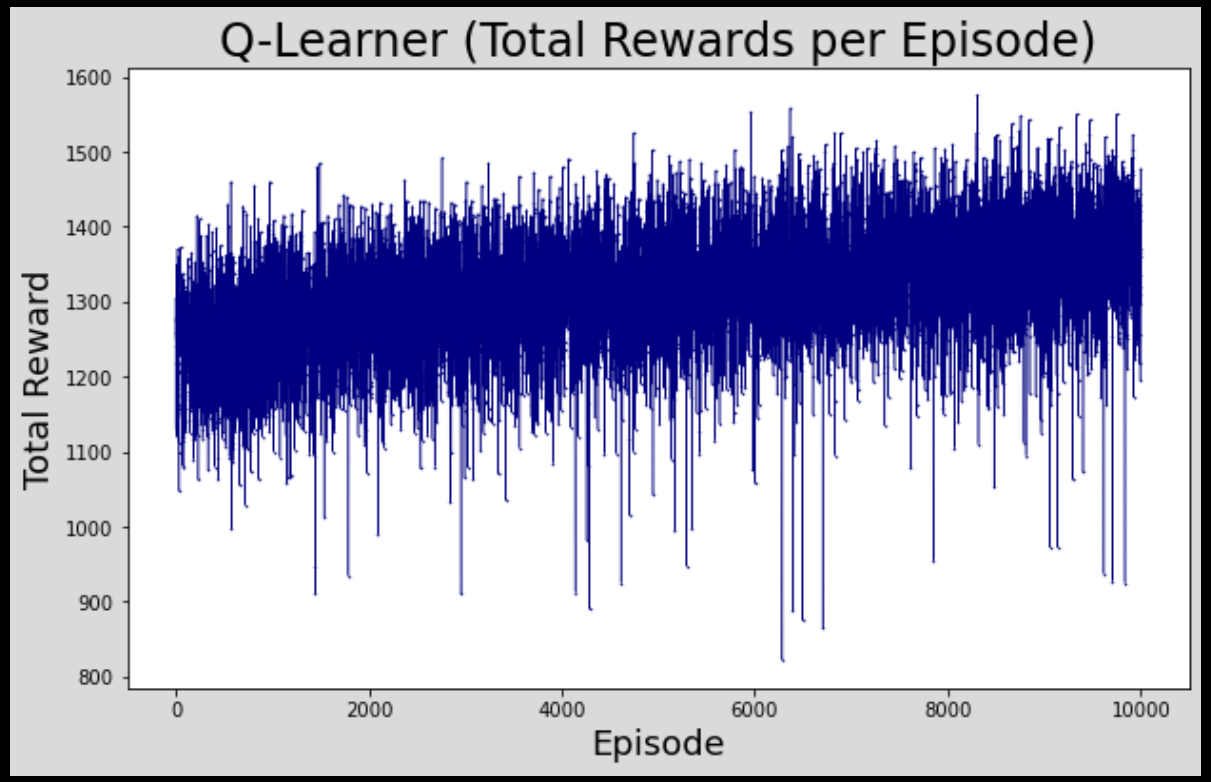
- Two environments were built, in the initial model, the action probability (during exploration) is based on the student data on question scores, in an extended model, the success probability (during exploration) is increased based on success ratio for already attempted questions for that chapter
- This was felt to be a more realistic representation of real question review setup

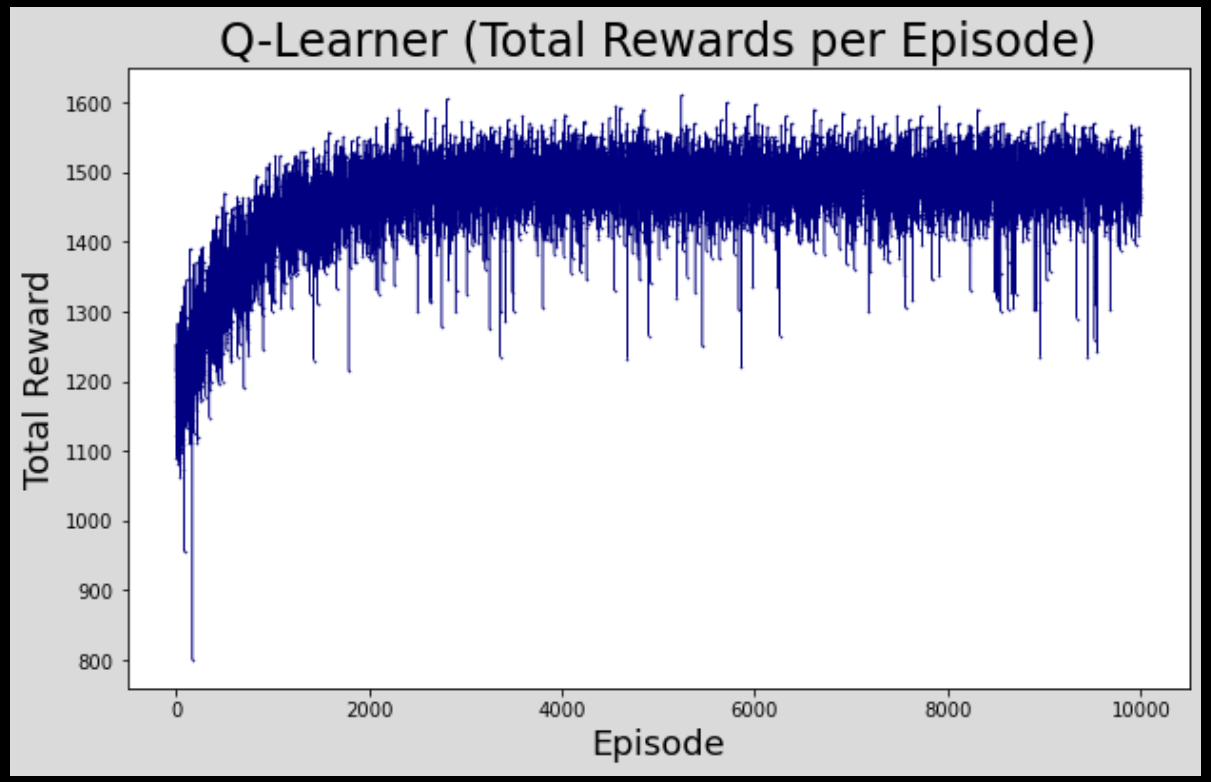
ENVIRONMENT COMPARISON: Q-VALUE CONVERGENCE



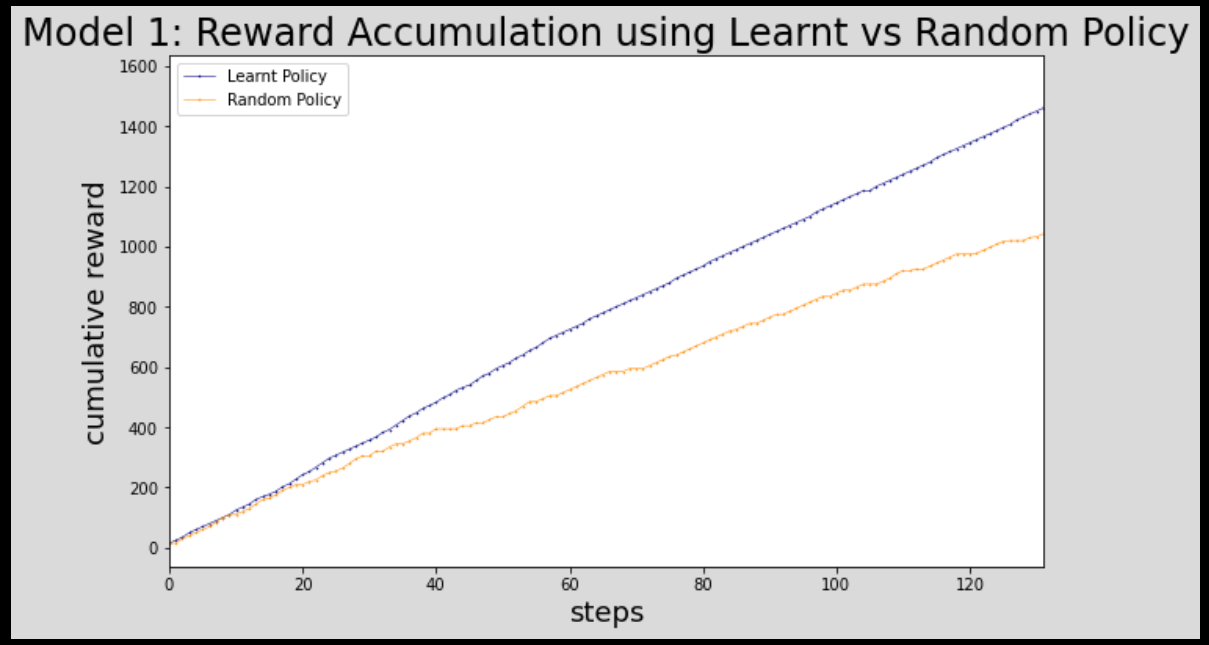


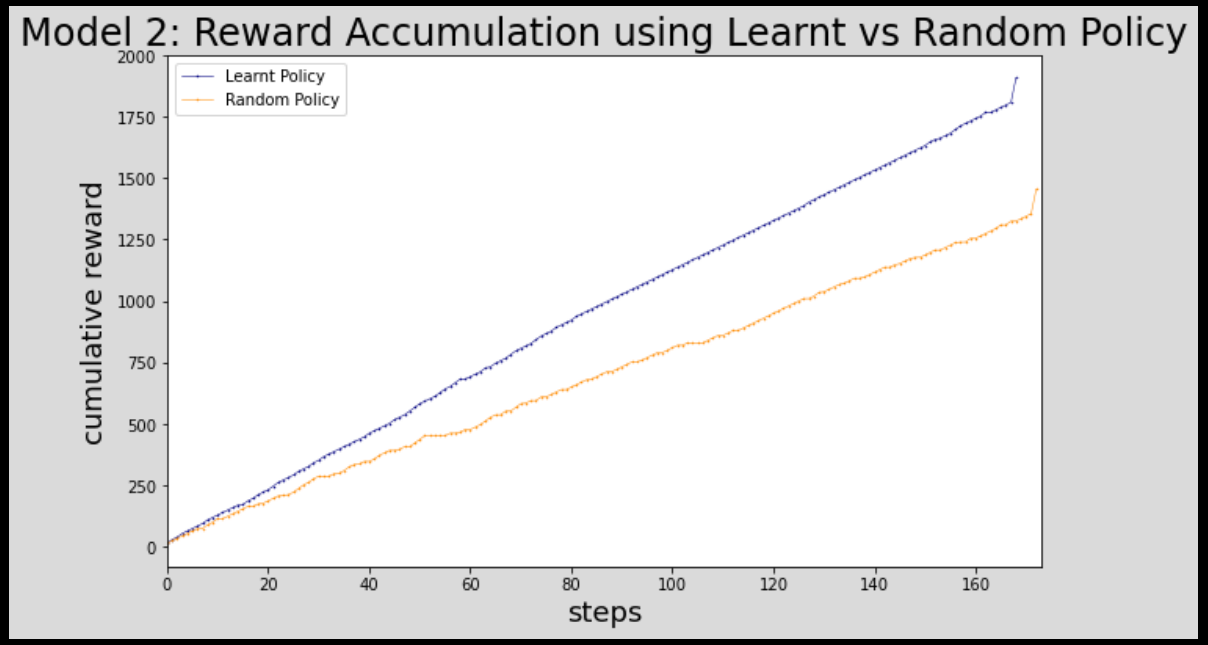
ENVIRONMENT COMPARISON: TOTAL REWARDS





ENVIRONMENT COMPARISON: POLICY COMPARISON





MODEL RETROSPECTIVE

1. **Issue**: In order to reduce the number of states, a lot of data is stored as part of the environment (like the questions attempted so far). The algorithm thus ends up treating states with different stored contextual information as the same, which reduces the value.

   Resolution: Vowpal Wabbit is a contextual bandit reinforcement learning algorithm that supports the concept of a context associated with the state, which might be the better approach that simple Q-Learning

2. **Issue**: Currently actions have deterministic results, which causes the environment to be formed as a deterministic MDP which can be solved without using RL

   Resolution: Change the actions to correspond to selecting a type of question to answer and make the result of the action stochastic

CONCEPTUAL ITS PROBLEM

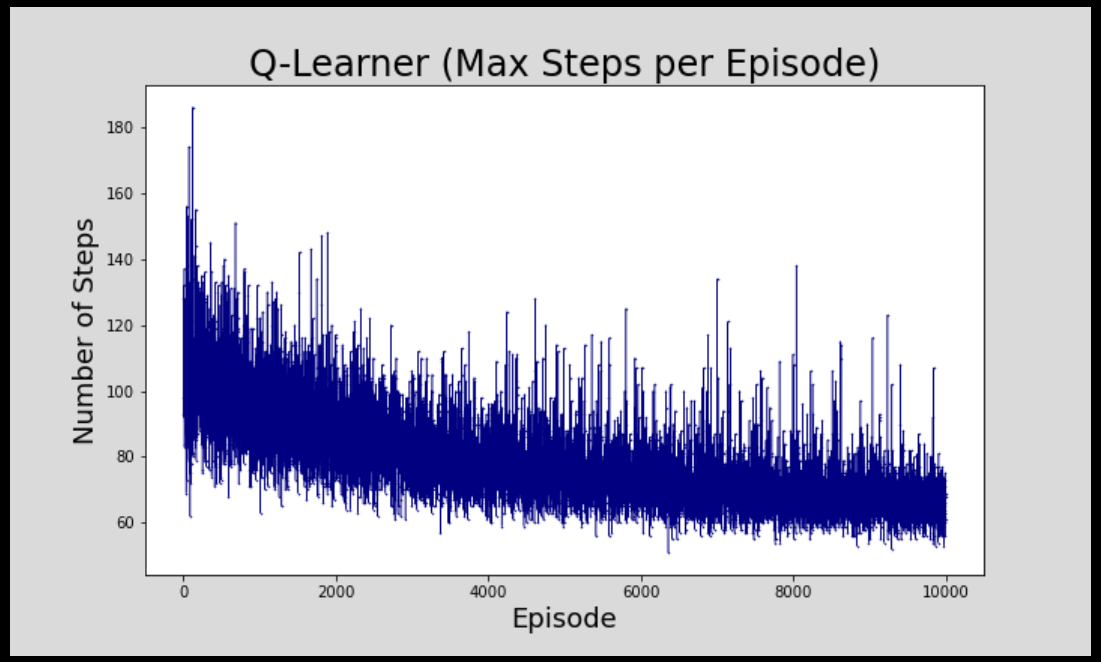
- States: 0-100000, all digits except the 1st stand for the level gained for a concept doing different activities
- Actions: The different activities allowed – query_piazza, query_chatbot, attempt_question, attempt_lab, get_labhints, quit
- Transition probabilities: A model free algorithm is used, so there is no explicit transition probability
- Reward: Only associated with lab and question, with an additional extra reward given for attaining max level (9) for each activity

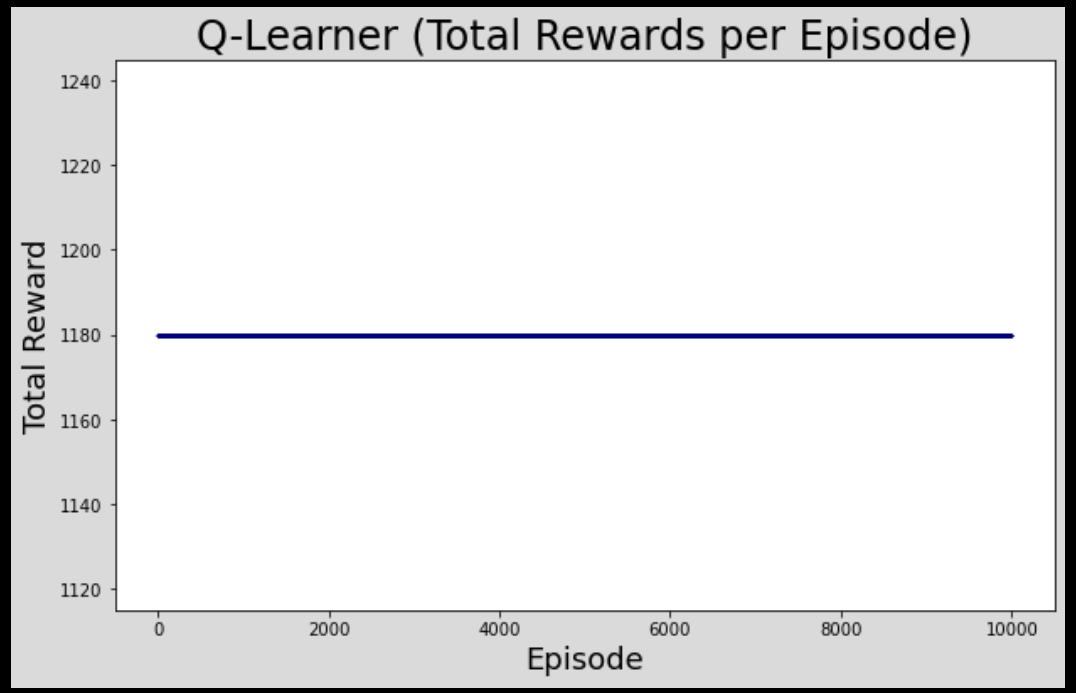
ITS SAMPLE ENVIRONMENT

- All levels (Piazza based knowledge, Chatbot based knowledge, TutorJS based knowledge, Question score and Lab score) go from 0-9
- All actions except ‘quit’ has a stochastic outcome.
- The actions ‘query_piazza’, ‘query_chatbot’, and ‘get_labhints’ has no rewards, but allows the agent to move to the next level for the related knowledge at 90% probability
- The actions ‘attempt_question’ and ‘attempt_lab’ has associated rewards (10) if successful, the success probability starts at zero and can improve to 90% based on score achieved in piazza and chatbot achieved knowledge levels for Questions, and TutorJS based knowledge level for Lab
- If all levels are completed before quitting, the user receives an additional big rewards (1000)

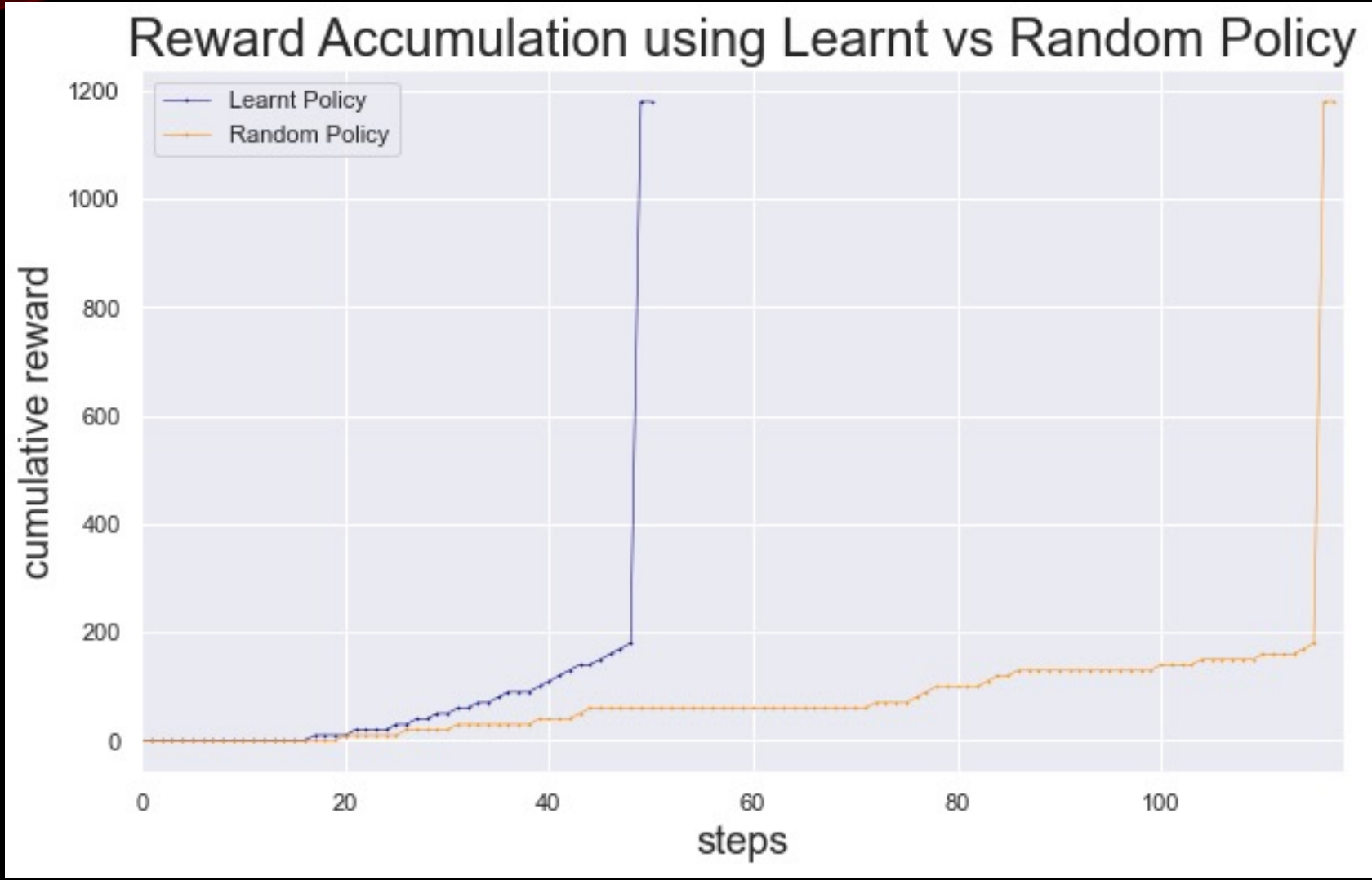
RESULTS: LEARNING PROGRESS

The intent is to find optimal (fastest) policy to gather all points, so though the total reward is unchanged, we see improvement in the number of steps for maximizing rewards.





RESULTS: POLICY COMPARISON



MODEL RETROSPECTIVE

- The ITS Sample environment, though simpler is modelled better as a decision-making problem than the Intelligent question review model
- The model is a very simplified representation of learning environment, and currently models the experience learning a single concept.
  - Generalizing it to all concepts will require the creation of a better state representations.
  - The system will need to support the interrelations between the different concepts, and potentially abstract it further in terms of meta concepts with multiple sub concepts
- Some areas of further research could be (much more complex)
  - POMDPs (Partially observable MDPs)
  - Multi-task reinforcement learning

FURTHER EXPLORATIONS

- Implement Reinforcement learning in the Chatbot recommendations and and TutorJS hints context
- Experiment with new models and RL algorithms like Contextual Bandit and Deep reinforcement learning algorithms
- Explore state generalization and discretization methods within the Conceptual ITS problem to expand the model to cover multiple concepts (we need 100000 states currently to cover a single concept)

RELATED READINGS

- AgentX: Using Reinforcement Learning to Improve the Effectiveness of Intelligent Tutoring Systems: Kimberly N. Martin and Ivon Arroyo
- https://towardsdatascience.com/self-improving-chatbots-based-on-reinforcement-learning-75cca62debce
- https://www.researchgate.net/publication/343874756_REINFORCEMENT_LEARNING_HELPS_SLAM_LEARNING_TO_BUILD_MAPS
- https://vowpalwabbit.org/tutorials/contextual_bandits.html#the-contextual-bandits-approach

Research Topics:

- https://www.mdpi.com/2079-9292/9/9/1363/pdf
- https://papers.nips.cc/paper/2019/file/677e09724f0e2df9b6c000b75b5da10d-Paper.pdf