Offline Policy Evaluation Across Representations with Applications to Educational Games

Travis Mandel[1],Yun-En Liu[1], Sergey Levine[2], Emma Brunskill[3], and Zoran Popović[1]
[1]Center for Game Science, Computer Science & Engineering, University of Washington
[2]Department of Computer Science, Stanford University
[3]School of Computer Science, Carnegie Mellon University
{tmandel, yunliu, zoran}@cs.washington.edu, svlevine@cs.stanford.edu, ebrun@cs.cmu.edu

ABSTRACT

Consider an autonomous teacher agent trying to adaptively sequence material to best keep a student engaged, or a medical agent trying to help suggest treatments to maximize patient outcomes. To solve these complex reinforcement learning problems, we must first decide on a policy representation. But determining the best representation can be challenging, since the environment includes many poorly-understood processes (such as student engagement) and is therefore difficult to accurately simulate. These domains are also high stakes, making it infeasible to evaluate candidate representations by running them online. Instead, one must leverage existing data to learn and evaluate new policies for future use. In this paper, we present a data-driven methodology for comparing and validating policies offline. Our method is unbiased, agnostic to representation, and focuses on the ability of each policy to generalize to new data. We apply this methodology to a partially-observable, high-dimensional concept sequencing problem in an educational game. Guided by our evaluation methodology, we propose a novel feature compaction method that substantially improves policy performance on this problem. We deploy the best-performing policies to 2,000 real students and show that the learned adaptive policy shows statistically significant improvement over random and expert baselines, improving our achievement-based reward measure by 32%.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Reinforcement learning, importance sampling, offline learning, POMDPs, policy evaluation, educational games

1. INTRODUCTION

Consider the problem of creating an autonomous teacher agent whose goal is to sequence material such that students remain engaged while learning as much as possible. In order to perform well, the agent must observe the behavior of students in response to different orderings so as to decide on the best order. For example, when teaching fractions, introducing 7/9 before 1/3 may cause students to be frustrated, while reversing the order may cause more students to learn. Furthermore, the best strategy will likely not be a fixed sequence, but instead a policy for adapting the sequence to meet the needs of each student. This is a challenging reinforcement learning problem, since the agent must infer meaning from the high-dimensional history of the student's past behavior, many features of which are irrelevant to building a good policy. The problem is also partially observable, since we cannot access the true mental state of the student, only observed behaviors. And the population of students may change over time, for example due to a regional school holiday causing students in some part of the world to stop playing, meaning the agent must be careful not to overfit to one specific time period. These features make the problem challenging even if constructing a policy only over a short horizon. Similar challenges arise in other human-centric domains: for example, Shortreed [20] must solve a high-dimensional problem of determining which sequence of 2 treatment choices to give each schizophrenia patient. In such high stakes environments, trying several approaches online is undesirable. Instead we propose using existing data to learn and evaluate new policies for future use.

When confronted with a new problem in these difficult domains, we must determine which of the many possible policy representations will perform best. Although the most general representations such as Predictive State Representations (PSRs) [15] may be capable of capturing the environment, learning all the parameters of a complex PSR or POMDP [22] model directly may not always be the best approach given limited data, e.g. due to overfitting. Ideally we could try many different representations, such as MDPs with different state spaces, POMDPs with different observation features, and a variety of direct policy representations, and pick the best one. This is similar to how researchers often try a variety of approaches (neural networks, SVMs, decision trees) when confronted with a new supervised learning problem. Ideally, we could identify the representation likely to generalize the best in our domain by evaluating them offline against a previously-collected dataset. This would allow us to experiment with new representations and policies, without the cost and risk of running them online.

Offline policy evaluation has been well-studied in domains where accurate simulators exist (such as a physics simulator in a robotic control task [16]). However, when an agent is designed for the purpose of human interaction, policy evaluation becomes challenging: experience is scarce and expensive, and the human mind is part of the environment and difficult to accurately simulate. Unless we have an ex-

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

tremely small number of candidate policies, we cannot run everything in the true environment. Instead, we need an evaluator which can use existing data collected from a non-deterministic policy to give us a reliable estimate of policy performance, regardless of which representation is used.

Previous work by Precup et al. [19] presented an unbiased estimator of policy reward based on importance sampling (IS). However, it has previously been used only for evaluating policies given a fixed representation, and Precup does not address the problem of evaluating policy generalization to new data, which is particularly critical in our setting where the population of students can change over time.

In this work, we present three main contributions. First, we introduce an unbiased offline evaluation methodology. We combine Precup's importance sampling estimator with a modified cross-validation approach, and show how one can use this method to tackle a previously unaddressed, yet important problem: how to evaluate the generalization ability of different representations offline. Second, guided by our evaluation methodology, we introduce a novel feature compaction algorithm, which address the problem of high-dimensionality by combining the strengths of PCA and neural networks, and outperforms other techniques for our problem. Finally, we present a challenging real-world application, involving learning the concept selection policy that optimizes engagement in an educational game. We show that our offline evaluation methodology allows us to select a policy which improves our metric of concept completion by over 30% compared to random and expert baselines when deployed to students.

2. BACKGROUND

We consider partially observable environments parameterized by a discrete action space $\mathcal{A}$, an action validity function V, an episode length T, and an observation space $\mathcal{O}$. We do not assume a state space is given. $\mathcal{O}$ can be multidimensional and real-valued. For each episode, the initial history h_1 consists of an initial scalar reward r_1 and an observation $o_1 \in \mathcal{O}$ provided by the environment. At each timestep $t \in \{1 \dots T\}$, we have a valid subset of actions specified by the validity function, $V_t = V(h_t) \subseteq \mathcal{A}$. The agent chooses an action $a_t \in V_t$. Then the agent receives a scalar reward r_{t+1} and an observation $o_{t+1} \in \mathcal{O}$ from the environment. The agent then appends the tuple (a_t, o_{t+1}, r_{t+1}) to the history h_t to get h_{t+1}. A stochastic policy π is defined as a probability distribution $\pi(a_i|h_t)$, denoting the probability of selecting action $a_i \in V(h_t)$ under policy π given a history h_t. Similarly, $\pi(a_1, \dots, a_t|h_t) \equiv \prod_{i=1}^{t} \pi(a_i|h_i)$.

Our objective is to find a policy π, which maximizes the reward function $\sum_{t=1}^{T} \gamma^{t-1} r_t$, where $\gamma \in [0, 1]$ is the discount rate. To find a good policy we must find a good representation, that is, a good way of specifying the (possibly non-deterministic) mapping from h_j to a_j for all possible histories h_j. A naïve representation would simply map each history to an action, but given the large (possibly infinite) number of possible histories h_j, a good representation somehow shares data among episodes with different histories.

3. PROBLEM SETUP

Games have recently received significant interest from the education community due to their unique potential to motivate students. Fractions is a particularly important topic since it is known to be a major obstacle to progress in mathematics [17]. We focus on creating an autonomous teacher agent that adaptively selects concepts (sets of 2-4 levels meant to teach a certain skill) in the educational fractions game Refraction, developed by the Center for Game Science (CGS). The goal of this puzzle game is to split lasers to direct the right amount of power into each target spaceship (see Figure 1). The game involves both mathematical and spatial reasoning. A version of the game can be played at `http://grail.cs.washington.edu/projects/ordering`. Refraction has been played by over 500,000 players, both online and in classrooms. Despite this success, it has a high dropoff rate when played online, especially on child-centric sites. To reduce this attrition rate, we focus on improving the early part of the game where dropoff is most prevalent.

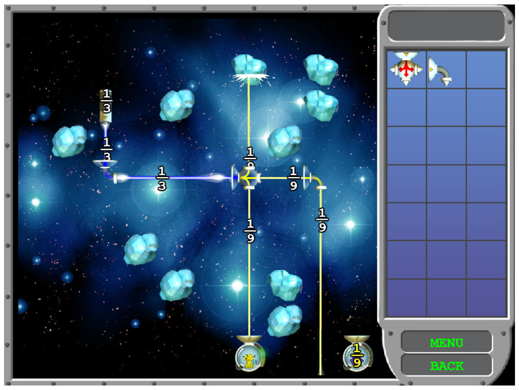

Figure 1: A level from our Refraction experiment. This level is from the Non-one Sources concept (see Section 8).

Choosing how to order concepts in the game to maximize engagement is difficult for game designers, even ignoring adaptivity, because the factors that cause a player to quit are complicated. For example, the mathematics may be in a reasonably engaging order, but the spatial difficulty of some of the levels may cause students to become frustrated and quit. In fact, we will see in sections 9 and 10 that the expert progression designed by game designers at CGS performs poorly, even compared to random sequencing, suggesting expert intuition may be limited.

Even though our problem is short horizon (there are 6 points where we can select the next concept), adaptively selecting concepts is still a challenging reinforcement learning problem for several reasons. First, we have a high dimensional set of 4,500 observation features, which capture each move the student has made while playing this concept and associated timing information. Some of these features are certainly irrelevant, but it is difficult to determine which a priori. Secondly, the problem is partially-observable, since we cannot observe the student's true mental state. The population of players that comes to the website to play Refraction is changing over time as well; for example on school holidays only a certain subset of students will be playing the game from home. Hence we must be careful not to overfit to one particular time period.

4. APPROACH OVERVIEW

Figure 2 gives an overview of our approach. Given data collected from an initial stochastic policy and an initial set of representations, we learn the representation parameters on the data, extract a policy from each representation, and evaluate the performance of each policy offline. If the estimated performance of the best policy is unsatisfactory, we create new representations to add to the representation set. Otherwise, we take the strongest policy according to our evaluation methodology and run it online, closing the loop.

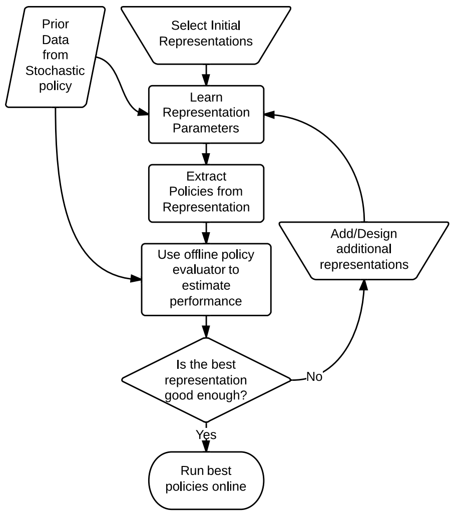
Prior Data from Stochastic policy
Select Initial Representations
Learn Representation Parameters
Extract Policies from Representation
Use offline policy evaluator to estimate performance
Add/Design additional representations
Is the best representation good enough?
No
Yes
Run best policies online

Figure 2: Approach overview.

5. PRIOR EVALUATION APPROACHES

A key part of our approach is to calculate a good offline estimate of how well a policy learned on some representation performs.

Many approaches acquire policies by first learning the parameters of an underlying generative representation, such as an MDP or POMDP. In this case, instead of considering policy performance directly, one instead optimizes model parameters using (log-)likelihood of the dataset under the model [6]. However, using likelihood to evaluate reinforcement learning representations can often be misleading, even if we compute the likelihood of held-out data to evaluate model generalization. The reason is that likelihood considers how well the system models every observation, but there may be observation features that are not important to model in order to build a good policy, but have a strong impact on the likelihood score. Even if the features are all relevant to the task, there may be aspects of continuous features that are not necessary to model. For example, a tea-making robot given temperature as a scalar feature might achieve high likelihood by modeling the exact temperature of a kettle of water at low temperatures, when all that matters is modeling whether it is above 100°C. For a concrete example of log-likelihood leading us astray in reality, see Figure 3.

It is not straightforward to modify log-likelihood to evaluate how well a model captures aspects of the system relevant to reward. Just restricting log-likelihood to compute $\log(p(r_1,\ldots,r_T|a_1,\ldots,a_T,\theta))$ is a poor evaluator since it ignores observations completely, and therefore cannot assess how well a model can interpret observations to better predict future rewards. Even computing $\log(p(r_1,\ldots,r_T|o_1,\ldots,o_T,a_1,\ldots,a_T,\theta))$ is a poor choice, since r_t may be easily inferable from o_t. The central issue is that a good evaluator needs to evaluate how well a model captures relevant observations needed to construct a good policy, but log-likelihood is not robust to observations that contain irrelevant information.

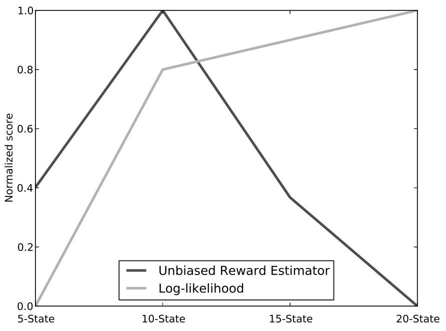

Figure 3: Unbiased reward estimator (Section 6) vs Log-Likelihood (Section 5). As we increase the number of hidden states in our learned PCA-NN-HMM model (see Section 9 for details) from 5 to 20, the cross-validated log-likelihood score increases even on held-out data, but the unbiased estimate of policy reward sharply peaks at 10 states. Because the score is cross-validated, we can see that the model is not overfitting. Instead we see that the 15 and 20 state models better capture the dynamics of the environment, but result in weaker policies. This indicates that selecting models based on likelihood alone can lead to poor policies.

Another method considered for comparing representations is ECR [24], however this method is not statistically consistent, which is a significant concern for high stakes domains. See Appendix A in the supplement[1] for more discussion.

6. IMPORTANCE SAMPLING METHOD

To evaluate policies offline, we use a method called importance sampling to estimate the expected policy reward. Intuitively, this method only uses samples where the entire action history has (by chance) matched the decisions that would have been made by the policy we wish to evaluate. Importance sampling (IS) is a general technique for estimating the expected value of a function f under a distribution p when samples are drawn from a different distribution q. Importance sampling is an attractive choice because it gives an unbiased estimate of f for any distributions p and q. This means that the expected value of the estimate is equal to the true value, which allows us a degree of trust in the estimates even on a limited amount of data, although the variance may be high. Note that some have proposed *normalized* (weighted) importance sampling [18], which has lower variance but is biased; this bias makes it unattractive for evaluation in high-stakes domains. If we let π_p be the policy we wish to evaluate, let π_q be the policy under which the data was collected, and we have a dataset of m episodes, we use IS to estimate the reward of π_p as follows [18]:

$$\hat{\mathbb{E}}\left[\sum_{t=1}^{T}\gamma^{t-1}r_t|\pi_p\right]=\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\pi_p(a_{1,i},\ldots,a_{T,i}|h_{T,i})}{\pi_q(a_{1,i},\ldots,a_{T,i}|h_{T,i})}\sum_{t=1}^{T}\gamma^{t-1}r_{t,i}\right], \tag{1}$$

where $\pi(a_{1,i},\ldots,a_{T,i}|h_{T,i})$ denotes the probability of taking actions $a_{1,i},\ldots,a_{T,i}$ given the history in episode i, as described in Section 2.

[1]`http://grail.cs.washington.edu/projects/ordering`

This estimator will have very high variance on a small amount of data, because only episodes that completely follow a policy from beginning to end are counted. The variance can be reduced by observing that rewards only depend on previous actions [19]:

$$\hat{\mathbb{E}}\left[\sum_{t=1}^{T}\gamma^{t-1}r_t|\pi_p\right]=\sum_{t=1}^{T}\frac{1}{m}\sum_{i=1}^{m}\frac{\pi_p(a_{1,i},\ldots,a_{t,i}|h_{t-1,i})}{\pi_q(a_{1,i},\ldots,a_{t,i}|h_{t-1,i})}\gamma^{t-1}r_{t,i} \quad (2)$$

Intuitively, this estimator averages the per-timestep reward for the examples that match π_2 (weighted by probability), and then combines these per-timestep rewards using the discount factor. This method is still unbiased [19], but has lower variance than equation (1).

One potential downside of this approach is that if we have episodes that are relatively long, it is unlikely there will be many samples that exactly match a given deterministic policy at later timesteps. This will yield a high variance estimate. However, in our domain (and others, e.g. [20]) the episodes are short, so we can effectively use Equation (2) to evaluate policies.

Real-world reinforcement learning problems often involve severe data sparsity and sometimes nonstationarity. Because of this, evaluating policies by running this estimator on the training data will cause us to choose policies that overfit, that is, appear to perform strongly on the training data but perform poorly in practice. We must therefore determine how well the policies generalize, which is especially important when data is limited. Following supervised learning, we could divide the data randomly into validation and training datasets, learn the policies on one portion of data and evaluate them on another fold using IS. However, this does not address nonstationary environments, which may fluctuate or change over time. We want to determine whether we have found a policy that is likely to generalize well across time. To this end, we propose a "temporal" cross-validation approach, where the dataset is divided into N folds, each of which spans a contiguous span of time (e.g. a series of days). To score well using this approach, a representation must generate policies that generalize to held-out time periods, and improving on just one time period (such as a special school holiday) will not score well. The performance of the algorithm across the 5 temporal folds also gives us a sense of the variance across time. However, the value of the initial random policy may change over time as well, and thus it may be difficult to tell how much better than random a policy is by comparing the overall variance of the random policy to the overall variance of the candidate policy. For example, a policy could always perform slightly better than random, but not appear significantly different from random if the variance across time is high. We therefore subtract the value of the random policy from the score, in order to focus on how these policies do relative to the random baseline.

7. CANDIDATE REPRESENTATIONS

Our approach (Figure 2) takes as input a variety of possible representations suitable for high-dimensional, partially observable problems, and learns policies on these representations. Here we describe a variety of representations we evaluated, including novel representations developed after observing how previous attempts performed. The set of methods considered include feature compression methods, POMDP models, and linear policies. Although they are inspired by our high dimensional, partially observable task, they are potentially applicable to other domains.

7.1 Feature Compression

Training models with thousands of potentially irrelevant observation features can be difficult, due to computational challenges, data sparsity, and overfitting. Some models may work well on the full feature space, but others (such as generative models, which model every observation feature) are not feasible on the full feature space. One approach to reduce the space from a large number of largely irrelevant observation features into a small set of relevant ones is automatic feature compression. Below, we describe two established methods for automatic feature compression, discuss their limitations, and describe a novel method that combines their strengths to address their limitations.

Principal Component Analysis (PCA) is a widely-used technique for taking a high-dimensional feature space and projecting it to a lower dimensionality. This is very useful for our purposes as the produced features are often quite rich, and are guaranteed to be orthogonal. However, doing PCA on a large feature space with a large number of features that are potentially irrelevant can lead to the relevant information being lost after the compression.

Neural networks (NN) are well-suited to compressing a large feature space to a smaller number of hidden units during a supervised prediction task, which in our case is predicting future reward from the current observation. Unlike PCA, neural networks can capture a nonlinear relationship between original and compressed features. However, neural networks often produce features which are highly correlated, which is not ideal if we want to find the most compressed representation possible.

We introduce a new feature compression method that seeks to build on the relative strengths of NN and PCA (nonlinear function approximation targeting a particular prediction task, and orthogonal features, respectively). The input features are fed into a neural network with one hidden layer of 100 units. The neural network is trained with Rprop to best predict future reward (in our case, time steps until the player quits). At this point it has found 100 non-orthogonal features that predict reward. Next, for each training example we take the output of each hidden unit and multiply it by the weight between it and the output unit:

$$\phi_i^{NN}(o) = h_i(o) * w_i$$

This causes hidden units which feature more prominently in the value of the output to also be better preserved in the compaction. Then we feed those 100 compressed features into PCA to compress further into a small number of rich, orthogonal features. We refer to this method as PCA-NN.

7.2 POMDP/IOHMM

We now consider how to use a small number of compacted or hand-selected features as the observations of a partially observable Markov decision process. A POMDP [22] is a tuple $(\mathcal{S},\mathcal{A},\mathcal{O},\mathcal{T},\Omega,R)$, where $\mathcal{S}$ is the finite set of hidden states, $\mathcal{A}$ is the action space, $\mathcal{O}$ is a space of observations, $\mathcal{T}$ is a set of transition probabilities conditioned on each state and action, Ω is a set of observation probabilities conditioned on each state (and potentially action), and R is a set of reward probabilities conditioned on each state and action.

We can treat the POMDP as an Input Output Hidden Markov Model (IOHMM) [3], where the inputs are actions, and learn the model parameters using the Baum-Welch (EM) algorithm similar to [6]. Because EM is prone to local optima, we do 10 random restarts for each model and take the one with the highest log-likelihood.

For our application, the outputs are set to be a vector of continuous observations , each of which is modeled by a single independent Gaussian. We simplify the POMDP so that observations at each state are not conditioned on action, since we want each state in the POMDP to represent the same behavior independent of time step, although these assumptions would be easy to remove. Episodes in our domain can terminate early, so we define a separate terminal state, and binary quit observation. We initialize the observation distribution so that the terminal state is the only state that can output the $terminal = true$ observation. Furthermore, the terminal state always transitions to itself.

Many widely-used offline POMDP planning techniques such as SARSOP [11] are not applicable due to the fact that our observations are continuous. As such, we choose the QMDP [14] approximate planning algorithm for its simplicity and speed. To help prevent overfitting, we learn the policies with $\gamma = 0.5$ (determined empirically) even though the true reward signal is undiscounted. Discount factors lower than 1.0 can help the POMDP model generalize if it is not accurate enough to issue reliable long-term predictions due to inaccurate transitions caused by data sparsity. Finally, we restrict the planning algorithm to obey the domain constraints, namely that concepts cannot be repeated and certain concepts must come before others.

7.3 Linear Direct Policy Search

Since we perform policy evaluation using importance sampling, a straightforward policy training procedure is to directly maximize the importance sampled estimate of the expected reward with respect to the policy parameters. To this end, we constructed a linear policy that maps a history of observations $o_1, \ldots, o_t$ to action probabilities. The outputs are converted to action probabilities by exponentiating and normalizing by a sum over all legal actions:

$$\pi(a_i|o_1, \ldots, o_t) = \frac{\exp(-\theta_i^{\mathrm{T}}\psi_t)}{\sum_{a_j \in V(s)} \exp(-\theta_j^{\mathrm{T}}\psi_t)} \tag{3}$$

where θ_i is the policy parameter for a_i and ψ_t is the input. This input can correspond to directly to the current observations o_t, to a set of learned features ϕ_t, or to entire histories of observations or features over all preceding time steps. When using whole histories, the input vector is padded with zeros to ensure an equal length at each time step. The weights θ_i are learned by inserting Equation 3 into Equation 2, differentiating, and optimizing with LBFGS.

7.4 Extracting Static policies

One interesting question in human-centric domains is how much improvement (if any) can be achieved by adapting actions based on observations. In the context of educational games, this means adapting to meet the needs of each students instead of choosing the best static policy, that is, overall best sequence of concepts. In general, *static policies* are defined as a class of policies which depend only on the action history: $\pi_s(a|h_t) = \pi_s(a|a_1, \ldots, a_{t-1})$. If we have learned an adaptive policy with strong performance, we would like to know if it simply found a static policy that generalized well, or if the adaptive choices make a large difference. To answer this question, we have to extract the "most likely" static policy from the adaptive one. Our approach first simulates running the policy over players in the training data, picking actions until the action history no longer agrees with the policy's choices. From this information we can calculate the overall likelihood of choosing each action at each timestep, and then for each timestep add to the static policy the most likely action that obeys the constraints.

8. EXPERIMENT DETAILS

Refraction begins with two required levels constituting the introduction, which familiarize players with the basic gameplay. Next, there is a set of seven concepts we teach students in the early part of Refraction, corresponding to seven sets of 2-4 levels each. In standard ("expert") order, they are: Spatial Reasoning(1), Half Splitting(2), Fourths(3), Third Splitting(4), Ninths(5), Sixths(6), and Non-one Sources(7). This expert ordering was designed based on the CGS game designers' intuition about what concept sequence would engage players, and was playtested significantly.

In our experiment there are only 6 time steps, so the agent is allowed to choose one concept to leave out. After all 6 have been given out there is a test level incorporating some of the more difficult concepts. Also we add a small set of hard constraints to which actions are valid at any given time, one of which is that no action may be repeated. The constraints prevent an advanced level involving a game mechanic from coming before the tutorial on that mechanic. The addition of these constraints leaves us with 560 possible static policies, and an infinite set of adaptive policies.

The agent is guided by a reward signal, which is based on concepts completed. In a voluntary game setting where players can quit at any time, measuring exit performance (and therefore total learning) is impractical. However, learning and engagement are intertwined, as it is not possible to achieve learning without first engaging students. Additionally, completion of each set of levels requires a degree of mastery of the corresponding concept, making completion a reasonable metric.

The reward signal gives a reward of 1.0 for concepts completed after the first, and 0.0 otherwise. This is because players often quit early on due to a host of factors outside our control, such as disliking the overall genre of the game, which can cause a lot of noise. Also, due to the fact that it is possible to eventually complete most levels by trial-and-error, players who complete a single concept and quit seem less likely to have learned something than players who complete two or more concepts. The reward for completing the final concept is given only if the end test is completed. These rewards are all undiscounted.

In order to learn and evaluate policies, we had access to raw logging data collected from an experiment on BrainPop.com, a popular educational website for grades K-12. The experiment was deployed with a randomized exploration policy, meaning actions were chosen uniformly at random from the valid options to ensure broad coverage of the space. Data was collected for about 6 weeks, during which time 11,300 people made it to the first action. We used this data to search for agent policies which could be run online.

We calculate 180 base features per level, and while the user normally plays 2-4 levels between timesteps, it is possible to

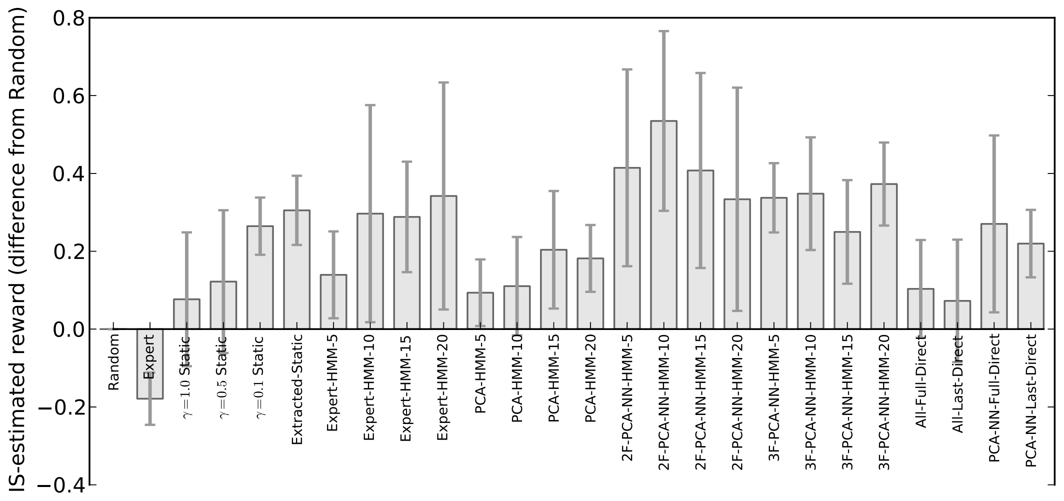

Figure 4: Importance sampling results. The error bars are 95% confidence intervals computed with 5-fold temporal cross validation using normal assumptions.

Random	**Expert**	**Static** Only depends on action history		**(IO)HMM-n** POMDP policies with n states (sec 7.2)				**Direct** Direct linear policies (sec 7.3)			
Initial Random Policy	Expert Designed Policy	**γ = n**	**Extracted**	**Expert**	**PCA**	**PCA-NN** (feature compression sec 7.1)		**Full** (full feature history)		**Last** (last timestep features)	
		Exhaustive search on IS with discount factor n	Extracted (sec 7.4) from PCA-NN-HMM-10	Uses two hand-chosen features	Two PCA features (sec 7.1)	**2F**	**3F**	**All**	**PCA-NN**	**All**	**PCA-NN**
						2 PCA-NN features	3 PCA-NN features	All features	2 PCA-NN features	All features	2 PCA-NN features

Table 1: Methods in Figure 4. The feature counts in this table refer to the feature count per timestep.

replay any of the 25 levels at any time, resulting in 4,500 features split across levels. These 180 base features include simple features such as total number of moves and total time, and more complicated features calculated based on the player's search graph for a level. Specifically, a playthrough of one level of Refraction consists of a human player moving through the game state space. Depending on the feature, each board configuration can be a state in this space, or we can aggregate states, for example by considering all board configurations with the same number of pieces on the board to be identical. We can think of playthroughs as directed multigraphs through a given state space, where the vertices are game states and the edges are labeled with the time the player takes to move from one state to the next. For each possible state space and associated graph, we can calculate its properties to use as features, such as the total number of nodes, the average vertex degree, the sum of all edge weights, and so on. These serve as our 180 base playthrough features, which are then expanded to 4,500 per timestep.

9. OFFLINE EVALUATION RESULTS

In this section, we evaluate all the approaches described in Section 7 using the importance sampling evaluator mentioned in Section 6. The approaches are described in detail in Table 1. The Random and Expert policies serve as baselines. We learn static policies (which give the same concept sequence to everyone), by performing exhaustive search on IS with various discount factors. Even though the true reward is undiscounted, learning with small discount factors can prevent overfitting since we have much more data at the beginning of the game. Note that since we are searching over policies, a discount factor of 0 is not a reasonable option since it would only consider the initial reward. We also extract a static policy from the best adaptive policy (see Section 7.4). We learn IOHMM/POMDP models (Section 7.2) with various feature compaction methodologies (Section 7.1), and two features were selected by hand which were predictive of quitting in other tasks. Lastly, we tried our linear direct policy search methodology (Section 7.3) with a variety of feature representations, investigating feature compaction and whether to include the full history of observation features or just the features from the last timestep.

One conclusion apparent from Figure 4 is that the variance of most approaches across folds is fairly high. This is due to a combination of temporal changes in the population, the learning methods being prone to local optima (hence they find better solutions on some folds and worse on others), and the inherent high variance of importance sampling. Next, we observe that 2F-PCA-NN-HMM-10 appears to do by far the best, indicating that the PCA-NN feature compaction had a significant benefit over the expert-chosen features. The expert features do show an improvement over PCA compaction however, illustrating that the PCA features are not very relevant to the task at hand. On the other end of the spectrum, the hand-designed expert policy does poorly, performing worse than random, suggesting that despite the best efforts of the CGS game designers, it is a challenging problem to design a good concept sequence by hand. Interestingly, the method that attempts to optimize importance sampling directly also fares poorly, although that may be partly due to the simple linear representation.

Among the static policies, extracting static policies from the best adaptive policy (2F-PCA-NN-HMM-10) generalizes better than training them to maximize IS. This is likely because the IOHMM is compacting action histories in a way that generalizes better than looking at the full history.

10. REAL-WORLD RESULTS

Since our estimator is unbiased, we would expect that with high probability each policy's true value will fall within the error bars shown in Figure 4. Since evaluation is expensive in our domain, it is not possible to run all policies on the game. Indeed, this is the motivation for evaluating policies offline in the first place.

However, to empirically verify that our offline evaluation methodology allowed us to discover policies that actually perform well in the game, we evaluated our two baselines (Expert and Random), the adaptive policy which appeared to do best offline (2F-PCA-NN-HMM-10) and the the best-performing offline static policy (Static Extracted). Because we saw high variance during training, out of the 5 folds we picked the 2F-PCA-NN-HMM-10 policy that had the highest difference from Random on its test fold, which we will henceforth refer to as "Adaptive", and the static policy extracted from it. The policies were run simultaneously and the experiment was stopped when we had 500 players in each condition, for 2000 total players.

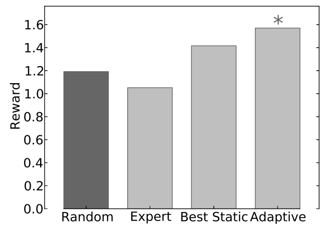

Figure 5: Experimental results on N=2000 players distributed equally among the four conditions. Adaptive received the highest average reward, and was the only policy that differed significantly from Random (p=0.002), as indicated by the asterisk.

The results are presented in Figure 5. As expected, Adaptive has the highest average reward. To determine statistical significance, we use the non-parametric Wilcoxon rank-sums test due to the non-normality of our data. We find that Adaptive is the only condition which is significantly different than random (p=0.002, W=263682.5, Z=3.17951) and Adaptive is also significantly different from expert (p<0.0001, W=267305.5, Z=4.04048). It performs on average 32% better than the Random and Expert baselines. This is an absolute difference of 0.38, which can be interpreted as approximately 1 level (recall that reward is measured in concepts). Expert had a slightly lower mean than Random (p=0.389, W=246684, Z=-0.862), and Best Static showed no significant improvement over Random (p=0.112, W=256878, Z= 1.58744).

We find that importance sampling gives us good estimates of actual performance, as all observed average rewards fall within their 95% confidence interval (Figure 4). However, this is unsurprising given that the estimator is unbiased.

The best static policy was the following: Half Splitting(2), Third Splitting(4), Spatial Reasoning(1), Non-one Sources(7), Ninths(5), Sixths(6). Compared to the expert policy in Section 8, it has delayed the introduction of spatial reasoning, and put the fractions which require single-splits (e.g. 1/2) before the double-splits (e.g. 1/9 = 1/3 * 1/3).

We observe that the adaptive policy chooses between two static policies roughly equally based on performance in the introduction. It seems (but is difficult to confirm precisely), that if the policy determines that the player does well on the introduction, they are given the Halves concept next, which involves mathematics, but if they do poorly they are given the Spatial concept, which does not.

11. RELATED WORK

There is a large body of work in learning policies offline, such as LSPI [12] or Fitted Q-Iteration [7]. When online evaluation of policies learned offline is expensive, successful real-world applications of reinforcement learning have generally relied on simulations or domain knowledge to guide the choice of representation. For example, several prominent RL applications, such as autonomous flight [16] or backgammon [23], relied on use of a simulator to determine the appropriate representation. In other cases, state-spaces were directly designed by experts [21]. An alternate approach is to use a complex set of input features, and use value function approximation over the feature set; however, this still requires choosing the right combination of input features and function approximator [10, 8]. In contrast, we seek to explore the space of representations without access to an accurate model of the environment, and without a priori knowledge about which representation best fits our problem.

Unbiased offline policy evaluation has been studied in the restricted domain of contextual bandits [13]. In reinforcement learning, importance sampling has been previously considered as an unbiased estimator of policy reward. Precup et al. [19] proposes importance sampling for learning the value function given an MDP state space, and Peshkin and Shelton [18] investigate using IS to guide a policy gradient algorithm. However, in contrast to our temporal cross-validation approach, neither addressed how to evaluate the generalization ability of policies to new data. Hachiya et al. [9] use a cross-validated importance sampling estimate of value function error (as estimated by the Bellman residual) to determine the best value of a single parameter of a fixed policy representation. Evaluating models based on value function error is a concern for two reasons, first, an improvement in modeling the value function does not necessarily improve policy performance, and second, this method cannot evaluate approaches which do not compute an explicit value function (such as policy gradient approaches). Our methodology does not have these restrictions, being able to directly compare the expected reward of the policy, and making no assumptions about the presence of a value function. Lastly, all three of these methods focus on tuning a few parameters of a fixed representation rather than comparing estimated reward across representations.

There has been significant work in applying reinforcement learning to develop educational strategies in intelligent tutoring systems (ITSes), such as using MDPs for hint generation tutors [2] or POMDPs to select topics of focus [4]. Work by Chi et al. [5] also focuses on improving an ITS via reinforcement learning, and addresses a similar problem of deciding between potential state spaces offline. Their approach did achieve encouraging learning gains, but did not use a statistically consistent estimator (for the proof, see Appendix A in the supplement[2]) for evaluating different representations, and only considered restricted MDP feature selection among a small set of features.

12. FUTURE WORK AND CONCLUSION

We are the first to show how an existing importance sampling estimator can be used to compare the generalization of

[2]`http://grail.cs.washington.edu/projects/ordering`

different representations, such as policy gradient approaches and POMDPs, even if the environment changes over time. We show that in a challenging educational games application, our methodology is a useful guide to allow us to choose the best representation for our domain. Without our importance sampling methodology, choosing between the many different representations, algorithms, and parameter choices would have been challenging. We would have been unlikely to design our novel PCA-NN feature compaction methodology, not knowing that it would outperform PCA in our domain. We show that our methodology allows us to select the best representations to run online, allowing us to see a 30% reward improvement in a domain where we cannot evaluate a large number of policies in the true environment. These results suggest that our evaluation methodology has the potential to improve performance in other challenging reinforcement learning tasks in which the choice of representation is unclear.

Since the variance of importance sampling estimates depends heavily on the horizon, it will take a very large amount of randomized data to evaluate long-horizon deterministic policies, and hence research is needed into better offline evaluation techniques for these problems. Another direction would be to explore how POMDPs can be trained to maximize importance sampling directly, and what effect that has on generalization. Yet another promising avenue for future work is exploring further methods of learning representations for high-dimensional problems.

Our evaluation methodology is general and can be applied to other discrete-action reinforcement learning problems where the choice of representation is unclear and an accurate simulator is not available. Similar situations arise in dialogue systems [24], healthcare [20], intelligent tutoring systems [5], and online advertisement delivery [1]. Our evaluation methodology could be applied to select appropriate representations for these domains, allowing us to better solve these challenging reinforcement learning problems.

13. ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation Graduate Research Fellowship grant No. DGE-0718124, as well as the Office of Naval Research grant N00014-12-C-0158, the Bill and Melinda Gates Foundation grant OPP1031488, the Hewlett Foundation grant 2012-8161, Adobe, Google, and Microsoft.

14. REFERENCES

[1] B. Baccot, R. Grigoras, and V. Charvillat. Reinforcement learning for online optimization of banner format and delivery. *Online Multimedia Advertising: Techniques and Technologies*, 2011.

[2] T. Barnes and J. Stamper. Toward automatic hint generation for logic proof tutoring using historical student data. In *ITS*. Springer, 2008.

[3] Y. Bengio and P. Frasconi. An input output HMM architecture. *NIPS*, 1994.

[4] E. Brunskill and S. Russell. RAPID: A reachable anytime planner for imprecisely-sensed domains. *UAI*, 2012.

[5] M. Chi, P. Jordan, K. VanLehn, and M. Hall. Reinforcement learning-based feature selection for developing pedagogically effective tutorial dialogue tactics. In *EDM*, 2008.

[6] L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *AAAI*, 1992.

[7] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. In *Journal of Machine Learning Research*, pages 503–556, 2005.

[8] A. Guez, R. D. Vincent, M. Avoli, and J. Pineau. Adaptive treatment of epilepsy via batch-mode reinforcement learning. In *IAAI*, 2008.

[9] H. Hachiya, T. Akiyama, M. Sugiayma, and J. Peters. Adaptive importance sampling for value function approximation in off-policy reinforcement learning. *Neural Networks*, 22(10):1399–1410, 2009.

[10] C. L. Isbell, M. Kearns, S. Singh, C. R. Shelton, P. Stone, and D. Kormann. Cobot in LambdaMOO: An adaptive social statistics agent. *AAMAS*, 2006.

[11] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *RSS*, 2008.

[12] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *JMLR*, 4:1107–1149, 2003.

[13] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, 2011.

[14] M. L. Littman, A. R. Cassandra, and L. Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *Machine Learning*, pages 362–370, 1995.

[15] M. L. Littman, R. S. Sutton, and S. P. Singh. Predictive representations of state. In *NIPS*, 2001.

[16] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics IX*, pages 363–372, 2006.

[17] N. M. A. Panel. *Foundations for success: The final report of the National Mathematics Advisory Panel*. US Department of Education, 2008.

[18] L. Peshkin and C. R. Shelton. Learning from scarce experience. *ICML*, 2002.

[19] D. Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[20] S. M. Shortreed, E. Laber, D. J. Lizotte, T. S. Stroup, J. Pineau, and S. A. Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84(1):109–136, 2011.

[21] S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *JAIR*, 16:105–133, 2002.

[22] E. J. Sondik. The optimal control of partially observable markov processes., 1971.

[23] G. Tesauro. Temporal difference learning and TD-Gammon. *Comm. of the ACM*, 38(3):58–68, 1995.

[24] J. R. Tetreault and D. J. Litman. Comparing the utility of state features in spoken dialogue using reinforcement learning. In *NAACL HLT*, 2006.