SPATIO-TEMPORAL ABSTRACTIONS IN REINFORCEMENT LEARNING THROUGH NEURAL ENCODING

Nir Baram, Tom Zahavy *
Electrical Engineering Department,
The Technion - Israel Institute of Technology, Haifa 32000, Israel
{nirb,tomzahavy}@campus.technion.ac.il

Shie Mannor
Electrical Engineering Department,
The Technion - Israel Institute of Technology, Haifa 32000, Israel
shie@ee.technion.ac.il

ABSTRACT

Recent progress in the field of Reinforcement Learning (RL) has enabled to tackle bigger and more challenging tasks. However, the increasing complexity of the problems, as well as the use of more sophisticated models such as Deep Neural Networks (DNN), has impeded the ability to understand the behavior of trained policies. In this work, we present the Semi-Aggregated Markov Decision Process (SAMDP) model. The purpose of the SAMDP modeling is to analyze trained policies by identifying temporal and spatial abstractions. In contrast to other modeling approaches, SAMDP is built in a transformed state-space that encodes the dynamics of the problem. We show that working with the *right* state representation mitigates the problem of finding spatial and temporal abstractions. We describe the process of building the SAMDP model by observing trajectories of a trained policy and give examples for using it in a toy problem and complicated DQN agents. Finally, we show how using the SAMDP we can monitor the trained policy and make it more robust.

INTRODUCTION

In the early days of RL, understanding the behavior of trained policies could be done rather easily (Sutton, 1990). Researchers focused on simpler problems (Peng and Williams, 1993), and policies were built using lighter models than today (Tesauro, 1994). As a result, a meaningful analysis of policies was possible even by working with the original state representation and relating to primitive actions. However, in recent years research has made a huge step forward. Fancier models such as Deep Neural Networks (DNNs) have become a commodity (Mnih et al., 2015), and the RL community tackles bigger and more challenging problems (Silver et al., 2016). Artificial agents are even expected to be used in autonomous systems such as self-driving cars. The need to reason the behavior of trained agents, and understand the mechanisms that govern its choice of actions is pressing more than ever.

Analyzing a trained policy modeled by a DNN (either graphically using the state-action diagram, or by any other mean) is practically impossible. A typical problem consists of an immense number of states, and policies often rely on skills (Mankowitz, Mann, and Mannor, 2014), creating more than a single level of planning. The resulting Markov reward processes induced by such policies are too complicated to comprehend through observation. Simplifying the behavior requires finding a suitable representation of the state space; a long-standing problem in machine learning, where extensive research has been conducted over the years (Boutilier, Dean, and Hanks, 1999). There, the goal is to come up with a transformation of the state space $\phi : s \rightarrow \hat{s}$, that can facilitate learning.

*These authors contributed equally

In the field of RL, where problems are sequential in nature, this problem is exacerbated since the representation of a state needs to account for the dynamics of the problem as well.

Finding a suitable state representation can be phrased as a learning problem itself (Ng, 2011; Lee et al., 2006). DNNs are very useful in this context since they *automatically* build a hierarchy of representations with an increasing level of abstraction along the layers. In this work, we show that the state representation that is learned automatically by DNNs is suitable for building abstractions in RL. To this end, we introduce the SAMDP model; a modeling approach that creates abstractions both in space and time. Contrary to other modeling approaches, SAMDP is built in a transformed state space, where the problem of creating spatial abstractions (i.e., state aggregation), and temporal abstractions (i.e., identifying skills) is facilitated using spatiotemporal clustering. We provide an example for building an SAMDP model for a basic gridworld problem where $\phi(s)$ is hand-crafted. However, the real strength of the model is demonstrated on challenging Atari2600 games solved using DQNs (Mnih et al., 2015). There, we set $\phi(s)$ to be the state representation automatically learned by the DNN (i.e. the last hidden layer). We continue by presenting methods for evaluating the fitness of the SAMDP model to the trained policy at hand. Finally, we describe a method for using the SAMDP as a monitor that alerts when the policy's performance weakens, and provide initial results showing how the SAMDP model is useful for shared autonomy systems.

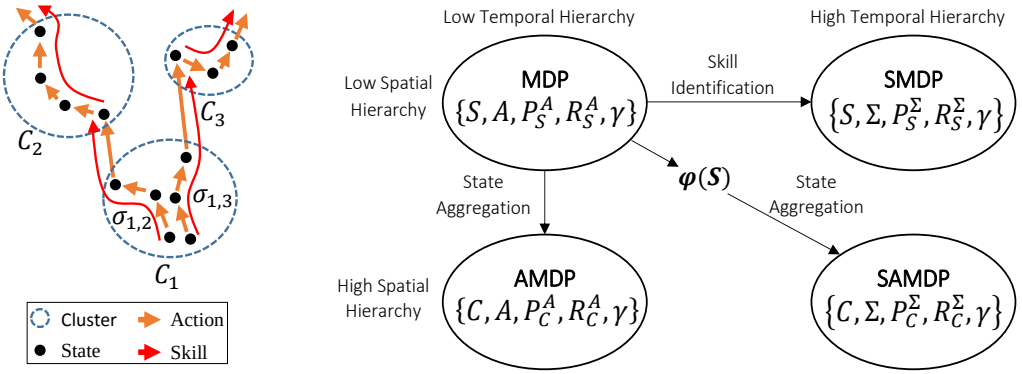


Figure 1: **Left:** Illustration of state aggregation and skills. Primitive actions (orange arrows) cause transitions between MDP states (black dots) while skills (red arrows) induce transitions between SAMDP states (blue circles). **Right:** Modeling approaches for analyzing trained policies.

BACKGROUND

We briefly review the standard reinforcement learning framework of discrete-time finite Markov decision processes (MDPs). An MDP is defined by a five-tuple $< S, A, P, R, \gamma >$. At time t an agent observes a state $s_t \in S$, selects an action $a_t \in A$, and receives a reward r_t. Following the agent's action choice, it transitions to the next state $s_{t+1} \in S$ according to a Markovian probability matrix $P_a \in P$. The cumulative return at time t is given by $R_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_t$, where $\gamma \in [0, 1]$ is the discount factor. In this framework, the goal of an RL agent is to maximize the expected return by learning a policy $\pi : S \rightarrow \Delta_A$; a mapping from states $s \in S$ to a probability distribution over actions. The action-value function $Q^\pi(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$ represents the expected return after observing state s, taking action a and then following policy π. The optimal action-value function obeys a fundamental recursion known as the optimal Bellman Equation: $Q^*(s_t, a_t) = \mathbb{E}\left[r_t + \gamma \max_{a'} Q^*(s_{t+1}, a')\right]$.

Skills, Options (Sutton, Precup, and Singh, 1999) are temporally extended control structures, denoted by σ. A skill is defined by a triplet: $\sigma =< I, \pi, \beta >$, where I defines the set of states where the skill can be initiated, π is the intra-skill policy, and β is the set of termination probabilities determining when a skill will stop executing. β is typically either a function of state s or time t. Any MDP with a fixed set of skills is a **Semi-MDP (SMDP)**. Planning with skills can be performed by learning for each state the value of choosing each skill. More formally, an SMDP is defined by a five-tuple $< S, \Sigma, P, R, \gamma >$. S is the set of states, Σ is the set of skills, P is the SMDP transition

matrix, γ is the discount factor and the SMDP reward is defined by:

$$R_s^\sigma = \mathbb{E}[r_s^\sigma] = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} | s_t = s, \sigma]. \quad (1)$$

The **Skill Policy** $\mu : S \to \Delta_\Sigma$ is a mapping from states to a distribution over skills. The action-value function $Q_\mu(s, \sigma) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t | (s, \sigma), \mu]$ represents the value of choosing skill $\sigma \in \Sigma$ at state $s \in S$, and thereafter selecting skills according to policy μ. The optimal skill value function is given by: $Q_\Sigma^*(s, \sigma) = \mathbb{E}[R_s^\sigma + \gamma^k \max_{\sigma' \in \Sigma} Q_\Sigma^*(s', \sigma')]$ (Stolle and Precup, 2002).

THE SEMI AGGREGATED MDP

Reinforcement Learning problems are typically modeled using the MDP formulation. Given an MDP, a variety of algorithms have been proposed to find an optimal policy. However, when one wishes to *analyze* a trained policy, MDP may not be the best modeling choice due to the size of the state space and the length of the planning horizon. In this section, we present the SMDP and Aggregated MDP (AMDP) models which can simplify the analysis by using temporal and spatial abstractions respectively. We also introduce the new Semi-Aggregated MDP (SAMDP) model, that combines SMDP and AMDP models in a novel way which leverages the abstractions made in each modeling approach.

SMDP (Sutton, Precup, and Singh, 1999), can simplify the analysis of a trained policy by using temporal abstractions. The model extends the MDP action space A to allow the agent to plan with temporally extended actions Σ (i.e., skills). Analyzing policies using the SMDP model shortens the planning horizon and simplifies the analysis. However, there are two problems with this approach. First, one still faces the high complexity of the state space, and second, the SMDP model requires to identify skills.

Skill identification is an ill-posed problem that can be addressed in many ways, and for which extensive research has been done over the years. The popular approaches are to identify bottlenecks in the state space (McGovern and Barto, 2001),or to search for common behavior trajectories, or common state region policies (McGovern, 2002). A different approach can be to build a graphical model of the agent's interaction with the environment and to use *betweenness centrality measures* to identify subtasks (Şimşek and Barreto, 2009). No matter what the method is, identifying skills solely by observing an agent play is a challenging task.

Alternative approach to SMDP modeling is to analyze a policy using spatial abstractions in the state space. If there is a reason to believe that groups of states share common attributes such as similar policy or value function, it is possible to use State Aggregation (Moore, 1991). State Aggregation is a well-studied problem that typically involves identifying clusters as the new states of an Aggregated MDP, where the set of clusters C replaces the MDP states S. Applying RL on aggregated states is potentially advantageous because the dimensions of the transition probability matrix P, the reward signal R and the policy π are decreased (Singh, Jaakkola, and Jordan, 1995). However, the AMDP modeling approach has two drawbacks. First, the action space A is not modified, and therefore the planning horizon remains intractable, and second, AMDPs are not necessarily Markovian (Bai, Srivastava, and Russell, 2016).

In this paper, we propose a model that combines the advantages of the SMDP and AMDP approaches and denote it by **SAMDP**. Under SAMDP modeling, aggregation defines both the states and the set of skills, allowing analysis with spatiotemporal abstractions (the state-space dimensions and the planning horizon are reduced). However, SAMDPs are still not necessarily Markovian. We summarize the different modeling approaches in Figure 1. The rest of this section is devoted to explaining the five stages of building an SAMDP model: (0) Feature selection, (1) State Aggregation, (2) Skill identification, (3) Inference, and (4) Model Selection.

(0) Feature selection. We define the mapping from MDP states to features, by a mapping function $\phi : s \to s' \subset R^m$. The features may be raw (e.g., spatial coordinates, frame pixels) or higher level abstractions (e.g., the last hidden layer of an NN). The feature representation has a significant effect on the quality of the resulting SAMDP model and vice versa; a good model can point out a good feature representation.

(1) Aggregation via Spatio-temporal clustering. The goal of Aggregation is to find a mapping (clustering) from the MDP feature space $S' \subset R^m$ to the AMDP state space C. Clustering algorithms typically assume that data is drawn from an i.i.d distribution. However, in our problem data

is generated from an MDP which violates this assumption. We alleviate this problem using two different approaches. First, we decouple the clustering step from the SAMDP model, by creating an ensemble of clustering candidates and building an SAMDP model for each (following stages 2 and 3). In stage 4, we will explain how to run a non-analytic outer optimization loop to choose between these candidates based on spatiotemporal evaluation criteria. Second, we introduce a novel extension of the celebrated K-means algorithm (MacQueen and others, 1967), which enforces temporal coherency along trajectories. In the vanilla K-means algorithm, a point x_t is assigned to cluster c_i with mean μ_i if μ_i is the closest cluster center to x_t (for further details please see the supplementary material). We modified this step as follows:

$$c(x_t) = \left\{c_i : \left\|X_t - \mu_i\right\|_F^2 \leq \left\|X_t - \mu_j\right\|_F^2, \forall j \in [1, K]\right\},$$

where F stands for the Frobenius norm, K is the number of clusters, t is the time index of x_t, and X_t is a set of $2w+1$ centered at x_t from the same trajectory: $\left\{x_j \in X_t \iff j \in [t-w, t+w]\right\}$. The dimensions of μ correspond to a single point, but is expanded to the dimensions of X_t. In this way, we enforce temporal coherency since a point x_t is assigned to a cluster c_i if its neighbors in time along the trajectory are also close to μ_i.

We have also experimented with other clustering methods such as spectral clustering, hierarchical agglomerative clustering and entropy minimization (please refer to the supplementary material for more details).

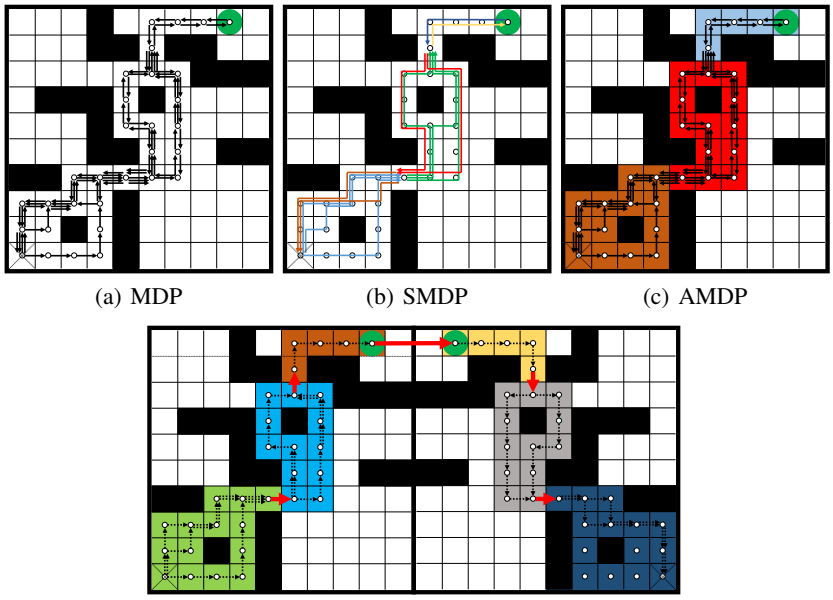
(2) Skill identification. We define an SAMDP **skill** $\sigma_{i,j} \in \Sigma$ uniquely by a single initiation state $c_i \in C$ and a single termination state $c_j \in C : \sigma_{ij} =< c_i, \pi_{i,j}, c_j >$. More formally, at time t the agent enters an AMDP state c_i at an MDP state $s_t \in c_i$. It chooses a skill according to its SAMDP policy and follows the skill policy $\pi_{i,j}$ for k time steps until it reaches a state $s_{t+k} \in c_j$, s.t $i \neq j$. We do not define the skill length k apriori nor the skill policy but infer the skill length from the data. As for the skill policies, our model does not define them explicitly, but we will observe later that our model successfully identifies skills that are localized in time and space.

(3) Inference. Given the SAMDP states and skills, we infer the skill length, the SAMDP reward and the SAMDP probability transition matrix from observations. The **skill length**, is inferred for a skill $\sigma_{i,j}$ by averaging the number of MDP states visited since entering SAMDP state c_i until leaving for SAMDP state c_j. The **skill reward** is inferred similarly using Equation 1.

The inference of the SAMDP **transition matrices** is a bit more puzzling, since the probability of seeing the next SAMDP state depends both on the MDP dynamics and the agent policy in the MDP state space. We now turn to discuss how to infer these matrices by observing transitions in the MDP state space. Our goal is to infer two quantities: (a) The SAMDP **transition probability matrices** $P_\Sigma : P_{i,j}^{\sigma \in \Sigma} = Pr(c_j|c_i, \sigma)$, measures the probability of moving from state c_i to c_j given that skill σ is chosen. These matrices are defined uniquely by our definition of skills as deterministic probability matrices. (b) The probability of moving from state c_i to c_j given that skill σ is chosen according to the agent SAMDP policy: $P_{i,j}^{\pi} = Pr(c_j|c_i, \sigma = \pi(c_i))$. This quantity involves both the SAMDP **transition probability matrices** and the agent policy. However, since SAMDP transition probability matrices are deterministic, this is equivalent to the agent policy in the SAMDP state space. Therefore by inferring transitions between SAMDP states, we directly infer the agent's SAMDP policy.

Given an MDP with a deterministic environment and an agent with a nearly deterministic MDP policy (e.g., a deterministic policy that uses an ϵ-greedy exploration ($\epsilon \ll 1$)), it is intuitive to assume that we would observe a nearly deterministic SAMDP policy. However, there are two mechanisms that cause stochasticity in the SAMDP policy: (1) Stochasticity that is accumulated along skill trajectories. (2) Approximation errors in the aggregation process. A given SAMDP state may contain more than one "real" state and therefore more than one skill. Performing inference in this setup, we might observe a stochastic policy that chooses randomly between skills.

Therefore, it is very likely to infer a stochastic SAMDP transition matrix, even though the SAMDP transition probability matrices and the MDP environment are deterministic, and the MDP policy is nearly deterministic. **(4) Model selection.** So far we have explained how to build an SAMDP from observations. In this stage, we'll explain how to choose between different SAMDP model candidates. There are two advantages of choosing between multiple SAMDPs. First, there are different hyperparameters to tune: two examples are the number of SAMDP states (K) and the window size (w) for the clustering algorithm. Second, there is randomness in the aggregation step. Hence, clustering multiple times and picking the best result will potentially yield better models.



(a) MDP (b) SMDP (c) AMDP

(d) SAMDP

Figure 2: State-action diagrams for a gridworld problem. **a.** MDP diagram: relate to individual states and primitive actions. **b.** SMDP diagram: Edge colors represent different skills. **c.** AMDP diagram: clusters are formed using spatial aggregation in the original state. **d.** SAMDP diagram: clusters are found after transforming the state space. intra-cluster transitions (dashed arrows) can be used to explain the skills, while inter-cluster transitions (big red arrows) loyally describe the governing policy.

We developed, therefore, evaluation criteria that allow us to select the best model, motivated by Hallak, Di-Castro, and Mannor (2013). We follow the Occams Razor principle and aim to find the simplest model which best explains the data. *(i)* **Value Mean Square Error(VMSE)**, measures the consistency of the model with the observations. The estimator is given by

$$VMSE = {}^{\|v-v_{SAMDP}\|}/_{\|v\|}, \tag{2}$$

where v stands for the SAMDP value function of the given policy, and v_{SAMDP} is given by: $V_{SAMDP} = (I + \gamma^k P)^{-1} r$, where P is measured under the SAMDP policy. *(ii)* **Inertia**, the K-means algorithm objective function, is given by : $I = \sum_{i=0}^{n} \min_{\mu_j \in C}(||x_j - \mu_i||^2)$. Inertia measures the variance inside clusters and encourages spatial coherency. Motivated by Ncut and spectral clustering (Von Luxburg, 2007), we define *(iii)* **The Intensity Factor** as the fraction of out/in cluster transitions. However, we define edges between states that are connected along the trajectory (a transition between them was observed) and give them equal weights (instead of defining the edges by euclidean distances as in spectral clustering). Minimizing the intensity factor encourages longer duration skills. *(iv)had been* **Entropy**, is defined on the SAMDP probability transition matrix as follows: $e = -\sum_i \{|C_i| \cdot \sum_j P_{i,j} \log P_{i,j}\}$. Low entropy encourages clusters to have less skills, i.e., clusters that are localized both in time and space.

SAMDP FOR GRIDWORLD

We first illustrate the advantages of SAMDP in a basic gridworld problem (Figure 2). In this task, an agent is placed at the origin (marked in X), where the goal is to reach the green ball and return. The state $s \in R^3$ is given by: $s = \{x, y, b\}$, where (x, y) are the coordinates and $b \in \{0, 1\}$ indicates whether the agent has reached the ball or not. The policy is trained to find skills following the algorithm of Mankowitz, Mann, and Mannor (2014). We are given trajectories of the trained agent, and wish to analyze its behavior by building the state-action graph for all four modeling approaches. For clarity, we plot the graphs on the maze using the coordinates of the state. The MDP graph (Figure 2(a)), consists of a vast number of states. It is also difficult to understand what skills the agent is using. In the SMDP graph (Figure 2(b)), the number of states remain high, however

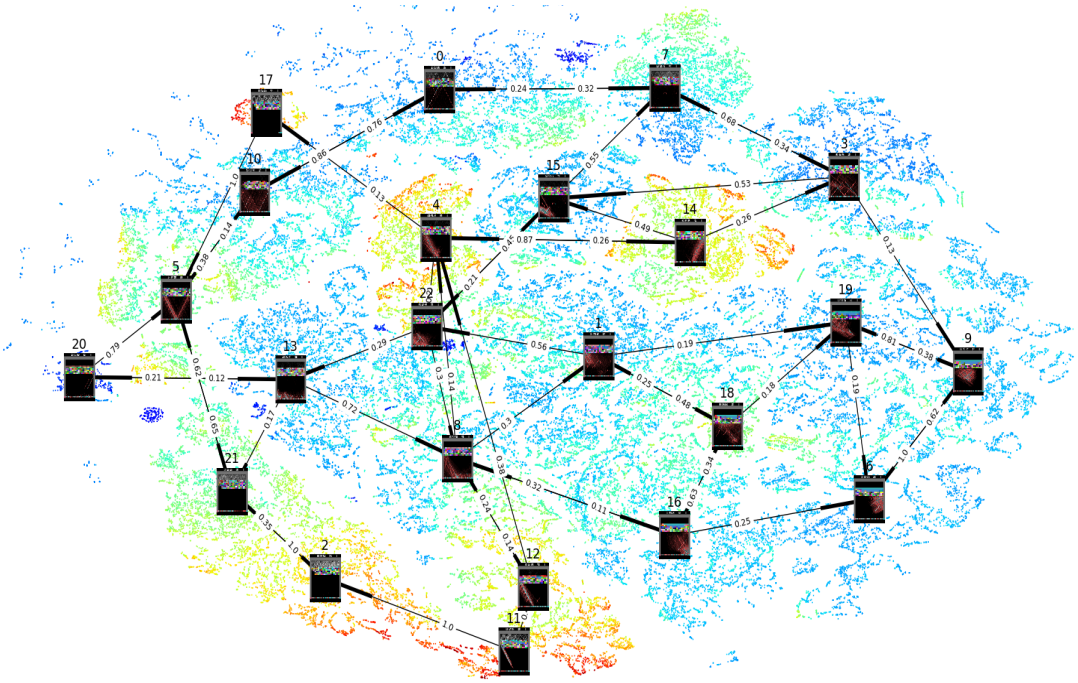


Figure 3: SAMDP visualization for Breakout. Each MDP state is represented on the map by its two t-SNE coordinates and colored by its value estimate (low values in blue and high in red). SAMDP states are visualized by their mean state (with frame pixels) at the mean t-SNE coordinate. An Edge between two SAMDP states represents a skill (bold side indicate terminal state), and the numbers above the edges correspond to the inferred SAMDP policy.

coloring the edges by the skills, helps to understand the agent's behavior. Unfortunately, producing this graph is seldom possible because we rarely receive information about the skills. On the other hand, abstracting the state space can be done more easily using state aggregation. However, in the AMDP graph (Figure 2(c)), the clusters are not aligned with the played skills because the routes leading *to* and *from* the ball overlap. For building the SAMDP model (Figure 2(d)), we transform the state space in a way that disentangles the routes:

$$\phi(x,y) = \begin{cases} (x,y), & \text{if } b \text{ is } 0 \\ (2L-x,y), & \text{if } b \text{ is } 1 \end{cases},$$

where L is the maze width. The transformation ϕ flip and translate the states where $b = 1$. Now that the routes *to* and *from* the ball are disentangled, the clusters are perfectly aligned with the skills. Understanding the behavior of the agent is now possible by examining inter-cluster and intra-cluster transitions.

SAMDPs FOR DQNs

Feature extraction: We evaluate a pre-trained DQN agent for multiple trajectories with an ϵ-greedy policy on three Atari2600 games, Pacman (a game where DQN performs very well), Seaquest (for the opposite reason) and Breakout (for its popularity). We let the trained agent play 120k game states, and record the neural activations of the last hidden layer as well as the Q values. We also keep the time index of each state to be able to find temporal neighbors. Features from other layers can also be used. However, we rely on the results from Zahavy, Zrihem, and Mannor (2016) that showed that the features learned in the last hidden layer capture a spatiotemporal hierarchy and therefore make a good candidate for state aggregation. We then apply t-SNE on the neural activations data, a non-linear dimensionality reduction method that is particularly good at creating a single map that reveals structure at many different scales. We use the two coordinates of the t-SNE map and the value estimation as the MDP state features. Each coordinate is normalized to have zero mean and

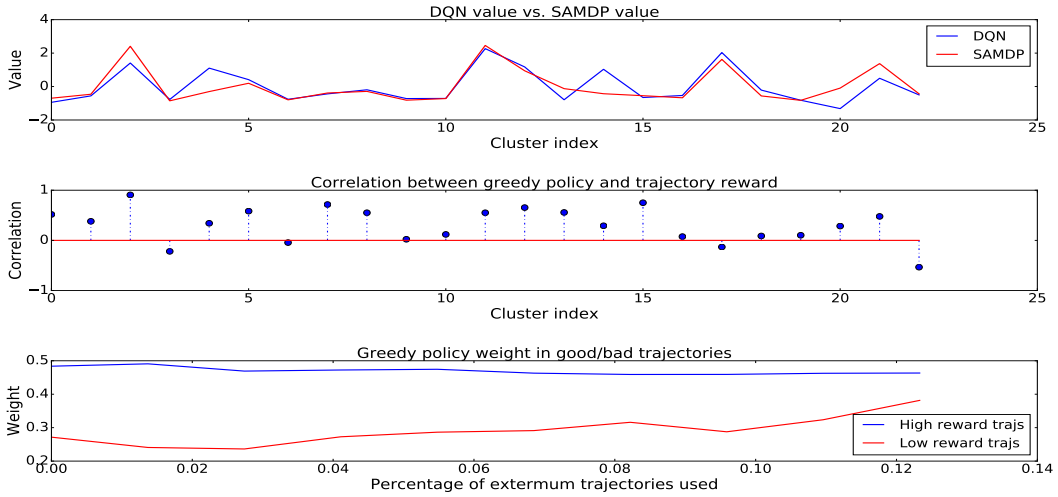


Figure 4: **Model Evaluation. Top:** VMSE. **Center:** greedy policy correlation with trajectory reward. **Bottom:** top (blue), least (red) rewarded trajectories.

unit variance. We have experimented with other configurations such as using the activations without t-SNE as well as different normalization. However, we found that this configuration results in better SAMDP models. We also use two approximations in the **inference** stage which we found to work well: 1) overlooking transitions with a small skill length (shorter than 2) and 2) truncating transitions with probability less than 0.1. We only present results for the Breakout game and refer the reader to the supplementary material for results on Pacman and Seaquest.

Model Selection: We perform a grid search on two parameters: *i*) number of clusters $K \in [15, 25]$. *ii*) window size $w \in [1, 7]$. We found that models larger (smaller) than that are too cumbersome (simplistic) to analyze. We select the best model in the following way: we first sort all models by the four evaluation criteria (SAMDP Section, stage 4) from best to worst. Then, we iteratively intersect the p-prefix of all sets (i.e., the first p elements of each set) starting with 1-prefix. We stop when the intersection is nonempty and choose the configuration at the intersection. The resulted SAMDP model for Breakout can be seen in Figure 3.

We also measure the p-value of the chosen model. For the null hypothesis, we take the SAMDP model constructed with random clusters. We tested 10000 random SAMDP models, none of which scored better than the chosen model (for all the evaluation criteria).

Qualitative Evaluation: Examining the resulting SAMDP (Figure 3) it is interesting to note the sparsity of transitions, which implies low entropy. Inspecting the mean image of each cluster reveals insights about the nature of the skills hiding within and uncovers the policy hierarchy as described in Zahavy, Zrihem, and Mannor (2016). The agent begins to play in low value (blue) clusters (e.g., 1,5,8,9,13,16,18,19). These clusters are well connected between them and are disconnected from other clusters. Once the agent transitions to the "tunnel-digging" option in clusters 4,12,14, it stays in there until it finishes to curve the tunnel, then it transitions to cluster 11. From cluster 11 the agent progresses through the "left banana" and hops between clusters 2,21,5,10,0,7 and 3 in that order.

Model Evaluation: We first measure the VMSE criterion, as defined in Equation 2 (Figure 4, top). We infer v by averaging the DQN value estimates in each cluster: $v^{DQN}(c_j) = \frac{1}{|C_j|}\sum_{i:s_i \in c_j} v^{DQN}(s_i)$, and evaluate V_{SAMDP} as defined above. Since the Atari environment is deterministic, the v_{SAMDP} estimate is accurate with respect to the DQN policy. Therefore, the VMSE criterion measures how well the SAMDP model approximates the true MDP. In practice, we observe that the DQN and SAMDP values are very similar; indicating that the SAMDP model fits the data well.

Second, we evaluate the greedy policy with respect to the SAMDP value function by: $\pi_{greedy}(c_i) = \underset{j}{\text{argmax}}\{R_{\sigma_{i,j}} + \gamma^{k_{\sigma_{i,j}}} v_{SAMDP}(c_j)\}$. We then measure the correlation between the greedy policy decisions and the trajectory reward. For a given trajectory j we measure P_i^j : the empirical distribution of choosing the greedy policy at state c_i and the cumulative reward R^j. Finally, we present the correlation between these two measures in each state: $corr_i = corr(P_i^j, R^j)$ in (Figure 4, center). A

positive correlation indicates that following the greedy policy leads to high reward. Indeed for most of the states, we observe positive correlation, supporting the consistency of the model.
The third evaluation is close in spirit to the second one. We partition the data to a train and test sets. We evaluate the greedy policy based on the train set and create two transition matrices T^+, T^- using the k top and bottom rewarded trajectories respectively from the test set. We measure the correlation of the greedy policy T^G with each of the transition matrices for different values of k (Figure 4 bottom). As clearly seen, the correlation of the greedy policy and the top trajectories is higher than the correlation with the bottom trajectories.
Eject Button: The motivation for this experiment stems from the idea of shared autonomy Pitzer et al. (2011). There are domains where errors are dreadful, and performance must be as high as possible. The idea of shared autonomy, is to allow an operator to intervene in the decision loop at critical times. For example, in 20% of commercial flights, the auto-pilot returns the control to the human pilots. In the following experiment, we show how the SAMDP model can help to identify where the agent's behavior deteriorates. **Setup.** *(a)* Evaluate a DQN agent, create a trajectory data set, and evaluate the features for each state (stage 0). *(b)* Divide the data into two groups: train (100 trajectories) and test (60). then build an SAMDP model (stages 1-4) on the train data. *(c)* Split the train data to k top and bottom rewarded trajectories T^+, T^- and re-infer the model parameters separately for each (stage 3). *(d)* Project the test data on the SAMDP model (mapping each state to the nearest SAMDP state). *(e) Eject* when the transitions of the agent are more likely under the T^- matrix rather then under T^+ (inspired by the idea of option interruption Sutton, Precup, and Singh (1999)). *(f)* We average the trajectory reward on *(i)* the entire test set, and *(ii)* the un-ejected trajectories sub set. We measure $\mathbf{36}\% \pm \mathbf{7.7}\%$, $\mathbf{20}\% \pm \mathbf{8.0}\%$, and $\mathbf{4.7}\% \pm \%\mathbf{1.2}$ performance gain for Breakout Seaquest and Pacman, respectively. The eject experiment indicates that the SAMDP model can be used to make a given DQN policy robust by identifying when the agent is not going to perform well and return control to a human operator or some other AI agent. Other eject mechanisms are also possible. For example, ejecting by looking at MDP values. However, the Q value is not monotonically decreasing along the trajectory as expected (See Figure 3). The solution we propose is to eject by monitoring transitions and not state values, which makes the MDP impractical in this case because it's state-action diagram is too large to construct, and too expensive to process.

DISCUSSION

SAMDP modeling offers a way to present a trained policy in a concise way by creating abstractions that relate to the spatiotemporal structure of the problem. We showed that by using the right representation, time-aware state aggregation could be used to identify skills. It implies that the crucial step in building an SAMDP is the state aggregation phase. The aggregation depends on the state features and the clustering algorithm at hand.

In this work, we presented a basic K-means variant that relies on temporal information. However, other clustering approaches are possible. We also experimented with agglomerative methods but found them to be significantly slower without providing any benefit. We believe that clustering methods that better relate to the topology, such as spectral clustering, would produce the best results. Regarding the state features; in the DQN example, we used the 2D t-SNE map. This map, however, is built under the i.i.d assumption that overlooks the temporal dimension of the problem. An interesting line of future work will be to modify the t-SNE algorithm to take into account temporal distances as well as spatial ones. A tSNE algorithm of this kind may produce 2D maps with even lower entropy which will decrease the aggregation artifacts that affect the quality of the SAMDP model.

In this work we analyzed discrete-action policies, however SAMDP can also be applied for continuous-action policies that maintain a value function (since our algorithm depends on it for construction and evaluation), as in the case of actor-critic methods. Another issue we wish to investigate is the question of consistency in re-building an SAMDP. We would like the SAMDP to be unique for a given problem. However, there are several aspects of randomness that may cause divergence. For instance, when using a DQN, randomness exists in the creation of the t-SNE map, and in the clustering phase. From our experience, though, different models built for the same problem are reasonably consistent. In future work, we wish to address the same problem by laying out an optimization problem that will directly account for all of the performance criteria introduced here. It would be interesting to see what clustering method will be drawn out of this process and to compare the principled solution with our current approach.

REFERENCES

Bai, A.; Srivastava, S.; and Russell, S. 2016. Markovian state and action abstractions for mdps via hierarchical mcts. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11(1):94.

Hallak, A.; Di-Castro, D.; and Mannor, S. 2013. Model selection in markovian processes. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Lee, H.; Battle, A.; Raina, R.; and Ng, A. Y. 2006. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, 801–808.

MacQueen, J., et al. 1967. Some methods for classification and analysis of multivariate observations.

Mankowitz, D. J.; Mann, T. A.; and Mannor, S. 2014. Time regularized interrupting options. *Internation Conference on Machine Learning*.

McGovern, A., and Barto, A. G. 2001. Automatic discovery of subgoals in reinforcement learning using diverse density.

McGovern, A. 2002. Autonomous discovery of abstractions through interaction with an environment. In *International Symposium on Abstraction, Reformulation, and Approximation*, 338–339. Springer.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540).

Moore, A. 1991. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In Birnbaum, L., and Collins, G., eds., *Machine Learning: Proceedings of the Eighth International Conference*. Morgan Kaufmann.

Ng, A. 2011. Sparse autoencoder. *CS294A Lecture notes* 72:1–19.

Peng, J., and Williams, R. J. 1993. Efficient learning and planning within the dyna framework. *Adaptive Behavior* 1(4):437–454.

Pitzer, B.; Styer, M.; Bersch, C.; DuHadway, C.; and Becker, J. 2011. Towards perceptual shared autonomy for robotic mobile manipulation. In *IEEE International Conference on Robotics Automation (ICRA)*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529:484–503.

Şimşek, Ö., and Barreto, A. S. 2009. Skill characterization based on betweenness. In *Advances in neural information processing systems*, 1497–1504.

Singh, S. P.; Jaakkola, T.; and Jordan, M. I. 1995. Reinforcement learning with soft state aggregation. *Advances in neural information processing systems* 361–368.

Stolle, M., and Precup, D. 2002. Learning options in reinforcement learning. Springer.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1):181–211.

Sutton, R. S. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *In Proceedings of the Seventh International Conference on Machine Learning*, 216–224. Morgan Kaufmann.

Tesauro, G. 1994. TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation* 6:215–219.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and computing* 17(4):395–416.

Zahavy, T.; Zrihem, N. B.; and Mannor, S. 2016. Graying the black box: Understanding dqns. *Proceedings of the 33 rd International Conference on Machine Learning (ICML-16), JMLR: volume48*.