ε-Machines for Model-Based Reinforcement Learning

Sam Holton
sjholton@ucdavis.edu
Department of Materials Science and Engineering, UC Davis

Abstract

Reinforcement learning is a highly general problem which is considered the ultimate goal of artificial intelligence research. Here, I apply ε-Machines to model-based reinforcement learning by treating the system-agent interaction as a single dynamic and consider how policy changes can induce new overall dynamics. I develop a simple algorithm for optimizing an agent's policy.

Introduction

Before the 20th century, scientists assumed that observers were independent of the system being studied. However, parallel developments in quantum mechanics and chaos theory call into question this idealization. The sensitivity of these systems with respect to the actions of the observer fundamentally changes how we should think about them. Situations where the observer has a strong influence on the system call into question what it means to "understand" the system separate from its observer. Rather, the observer can now ask how they *should* act when interacting with the system. In this case, the observer is perhaps better described as an *agent*.

Here, I examine how we can apply the ε-Machine formalism [1] to reinforcement learning problems. By considering the agent to be a part of the system under study, we can then model the agent-system dynamics as an ε-Machine, which provides a model for how adjusting the agent's policy changes the resulting dynamics. By comparing the asymptotic behavior of the agent-system pair under different policies, we can choose the policy which maximizes the probability of observing sequences that the agent prefers.

This report is structured as follows. First, I review progress in the field of reinforcement learning. Next, I explain how to adapt the existing ε-Machine approach to a reinforcement learning problem. Afterwards, the Policy Counterfactuals section reviews several intuitive examples of how an agent can adjust their policy to change the overall dynamics. The Policy Optimization section extends this by specifying the overall approach to RL using ε-Machines and identifies a simple algorithm to select a good policy. Finally, I consider an application of this approach to a simple system and conclude.

Background: Reinforcement Learning

Reinforcement learning (RL) is a mature field examining how artificial agents should pursue their goals in a variety of environments. The problem of developing general, capable reinforcement learners is considered one of the most important steps towards creating an artificial intelligence and has applications to many fields such as robotics, high-throughput experimentation, and control systems.

A typical RL problem consists of four components:

1. A set of possible observed states
2. A set of possible actions the agent can take
3. A utility function which maps observed states to utilities
4. A policy which selects actions conditional on past observations and actions

The goal of any reinforcement learning algorithm is to find a policy which maximizes total utility. A significant amount of research examines different algorithms for finding good policies in a variety of environments. It can be generally divided amongst model-free and model-based approaches.

Model-free approaches do away with an explicit model for their environment and attempt to find an optimal policy empirically. Q-learning [2] is a commonly used model-free RL technique. It involves tabulating the average reward obtained from different combinations of states and actions and then selecting the best-performing action in each state. Though this avoids making assumptions about the environment, Q-learning can be quite slow to converge, requires a lot of data, and performs poorly in environments with a large state-space.

Model-based approaches rely on some theory of system-agent interaction in order to select policies. Adaptive dynamic programming (ADP) is a good example of model-based RL [3]. It constructs a model of how states and actions induce transitions to other states and learns the utilities of these states. This constructed model is a Markov decision process which can then be solved using dynamic programming techniques. Like with Q-learning, ADP has difficulties handling complicated environments, however, ADP often has faster convergence than Q-learning since it can use its model of the environment to identify good policies.

Both model-free and model-based approaches have been used to control drones [4], play games [5], and complete many other tasks. Despite the success of current RL approaches and large body of literature, I believe the ε-Machine approach demonstrated here offers several advantages. First, studying the application of ε-Machines to reinforcement learning can provide better information-theoretic foundations of RL and potentially help create new RL algorithms with improved performance. Second, the system-agent model extends the study of ε-Machines to situations where the observer is a part of the system under study, which may produce new concepts applicable to decision theory and the study of quantum systems. Third, the ε-Machine approach finds a parsimonious representation of the overall dynamics and naturally defines the different states of the system instead of requiring the user to define the states themselves. Both of these properties can significantly improve existing RL techniques.

Previously, ε-Machines have been used to study interactive learning, a problem similar to RL which eschews rewards and instead seeks to maximize the predictive power of the agent via the policy [6]. As such, this strategy is limited to agents which seek to control the system in a very particular way. This is distinct from the approach discussed here as we aim to learn policies for agents with arbitrary goals.

Applying ε-Machines to the RL problem

Readers may notice a conflict between the ε-Machine formalism and the RL problem. ε-Machines take only a single history of observations, while the RL problem has two distinct streams inputs: the sequence of observations and the sequence of selected actions. We must reconcile this difference if we are to apply the ε-Machine approach to RL problems.

Fortunately, there is a simple solution. Since we are considering the system-agent dynamic holistically, we can combine the history of observations and the history of actions by inserting the agent's "action symbols" into the system's history at regular intervals. This new sequence describes the system-agent dynamics and can *itself* be modeled as an ε-Machine. This new ε-Machine determines how histories of system output symbols and agent actions combine to produce future system output symbols and actions.

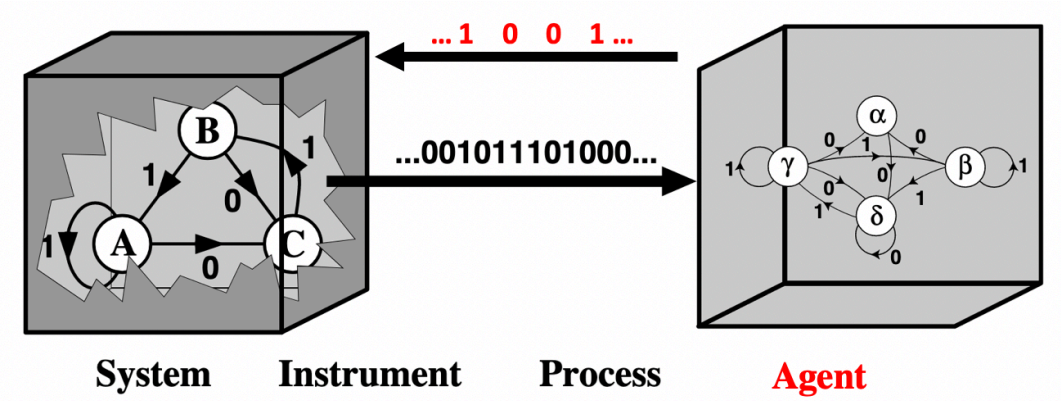

Figure 1. The agent observes system dynamics and outputs a symbol sequence corresponding to its actions at each timestep. Adapted from Physics of Computation and Information, Lecture 22, Slide 4.

With the system-agent ε-Machine constructed, we can now ask what control the agent has over the ε-Machine. By looking at the agent's history of action symbols and the outputs of previous causal states, the agent can identify which causal states it controls. For example, if the agent remembers that it emitted a '1' symbol ten times in a row, the agent can look back on the history and find which causal state also emitted a '1' symbol the last ten times it was visited. This indicates that the agent controls the output of this causal state in the ε-Machine.

Knowing the role the agent plays in the overall dynamics, we now turn to the question of how changes to the policy change the resulting ε-Machine.

Policy Counterfactuals

There are two ways an agent can consider changes to its policy, using only the inferred ε-Machine.

First, since the agent knows which causal states it controls, it can consider fixing the output of these states, which has the effect of "pruning" the ε-Machine by removing an outgoing transition

from one of these states (Fig 2b). Second, the agent can select certain outputs conditional on the causal state and the previous output symbol (Fig 2c). This has the effect of "budding" a causal state that the agent controls into separate causal states, one for each subset of symbols that the agent treats differently.

To make these ideas more concrete, we can apply these concepts to a simple example. Let's consider a process where first bit is random, the second bit is controlled by our agent, and the third bit is a copy of the agent's bit.

Under a random policy, the system-agent interaction is described by the ε-Machine in Figure 2a. The agent knows that they control causal state B. The agent can prune the outputs of state B to only emit the '0' symbol leading to the ε-Machine in Figure 2b. Alternatively, the agent can bud causal state B into states B and C, conditional on the incoming symbol, resulting in the ε-Machine in Figure 2c.

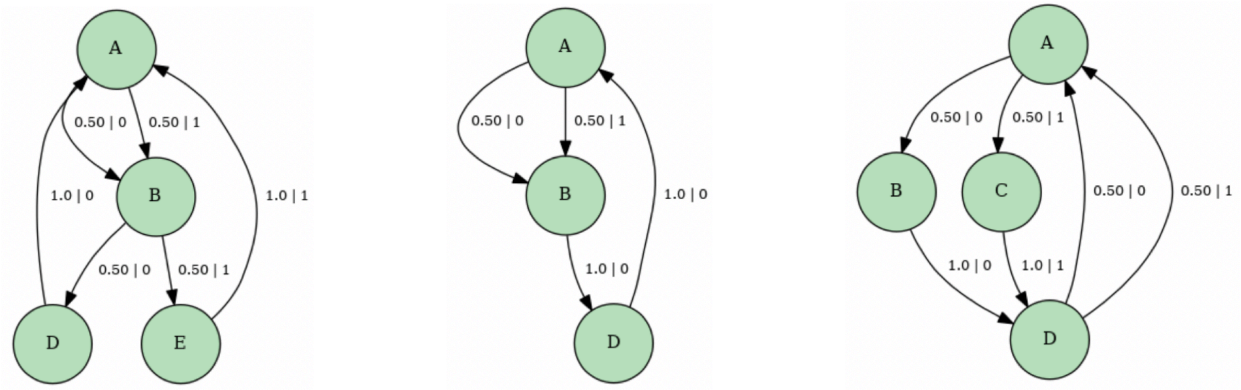

***Figure 2**. A) Random policy. B) Pruning the output of state B. C) Budding state B into two causal states.*

Policy Optimization

We can now specify the overall approach to RL using ε-Machines. First, the agent pursues a random policy and observes the system-agent dynamics. Next, the agent infers the overall ε-Machine corresponding to these dynamics and identifies which causal states it controls. With this knowledge, the agent can then derive the ε-Machines resulting from different policies. The best policy induces an ε-Machine with the most favorable asymptotic word distribution.

But selecting an optimal policy is not as simple as it sounds. Note that if an agent controls N causal states with a possible output symbols (actions), then there are $O(a^N)$ different possible ε-Machines the agent can produce by pruning alone. This exponential dependence on the number of controlled causal states makes it infeasible to evaluate the impacts of all policies. Instead, a simple algorithm which finds a good (as opposed to optimal) policy is of practical necessity.

Greedy algorithms are a good option when facing a task with a large search space where working solutions will suffice. The problem of choosing a policy on an ε-Machine is naturally suited to a greedy approach and can be pursued as follows.

First, the agent selects a starting controlled causal state. Next, they "bud" or split this causal state into several states conditional on the different incoming symbols to this state. Then, they iterate through each of these budded states and consider pruning different output symbols. If a pruning improves the asymptotic word distribution, it is kept, otherwise, it is rejected. After iterating through all of the buds, some of the causal states will have the same incoming and outgoing transitions, and can be combined. Once this process is complete, the agent moves to the next controlled causal state in the ε-Machine, starting with states close to the initial state and then moving to causally downstream states, repeating the budding and pruning process.

In the next section, I apply this model-based RL algorithm to an example problem.

Demonstration

With the fundamental approach to RL using ε-Machines specified, we can consider a demonstration on a simple system.

Let's consider a system of four repeating bits where the first bit is random, the second is controlled by the agent, the third is random, and the fourth is controlled by the agent. For the purposes of this example, we assume that the agent seeks to maximize the probability of observing the '0000' and '1111' strings. Starting with a random policy, the agent will identify the presentation in Figure 3a. The agent then begins with controlled causal state B by budding it into states B and C conditional on the incoming symbols, resulting in Figure 3b. Next, the outputs from B and C are each pruned to maximize the probability of observing the '0000' or '1111' strings, resulting in Figure 3c.

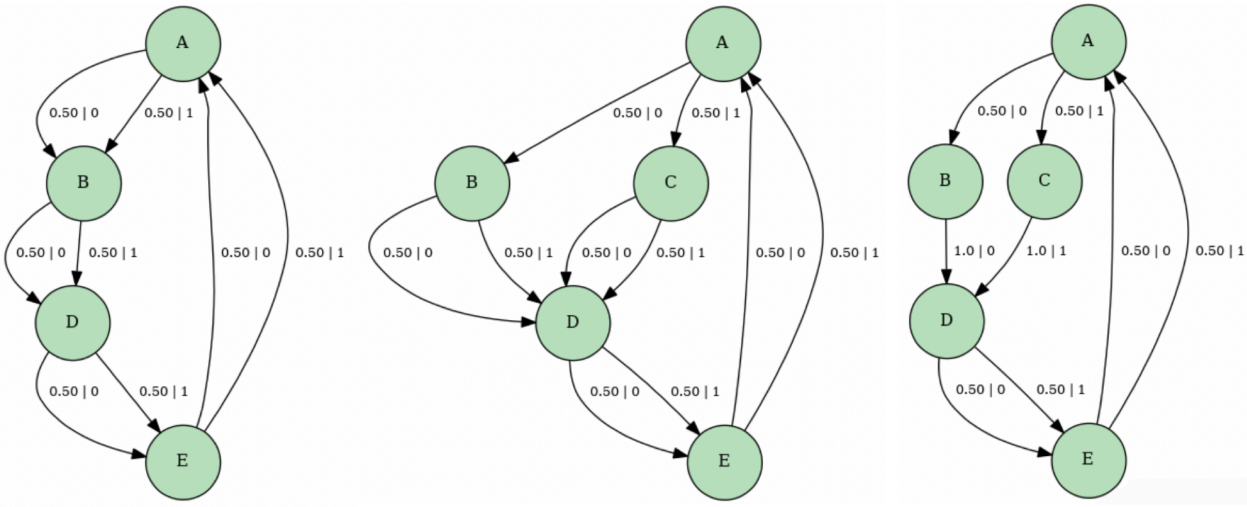

***Figure 3**. A) Random policy. B) Budding state B into two causal states. C) Pruning the output of states B and C.*

With the policy adjustments at state B complete, we can now examine how to change the policy at state E. Figure 4a reproduces Figure 3c for clarity. We start by budding causal state E by into states E and F conditional on the incoming symbols, resulting in Figure 4b. Next, the outputs from E and F are each pruned to maximize the probability of observing the '0000' or '1111' strings, resulting in Figure 4c.

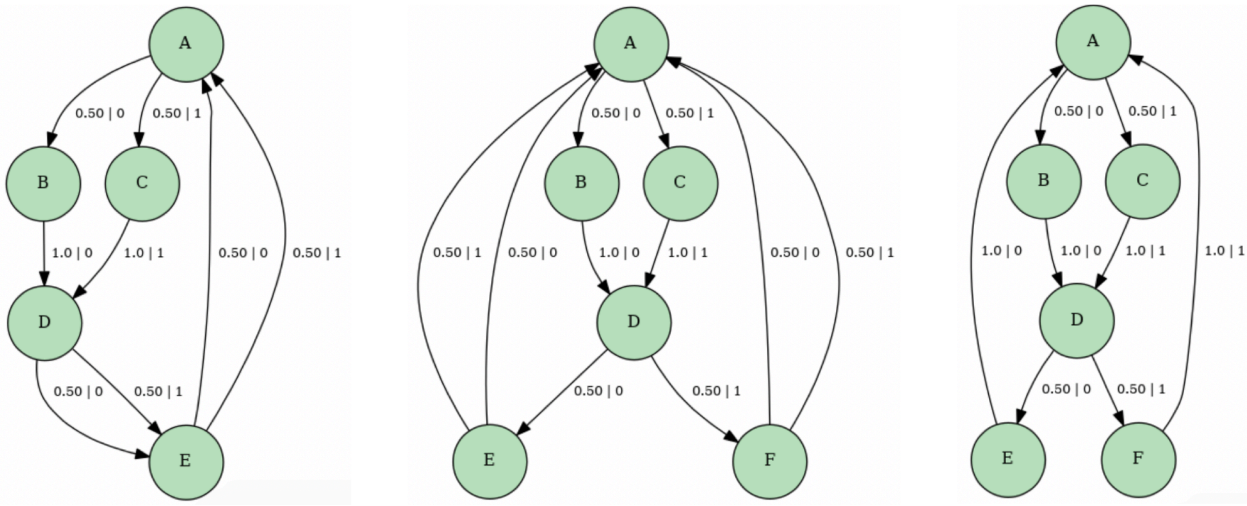

Figure 4. *A) Reproduction of Figure 3c. B) Budding state E into two causal states. C) Pruning the output of states E and F.*

Conclusion

There are several promising routes by which this work could be extended. First, new algorithms for inference, larger memory policies, and better policy selection algorithms would greatly improve performance. Second, determining the formal relationship between the complexity of the optimal policy and the complexity of the system might provide a new grounds for proving the Good-Regulator Theorem [7]. Third, extending this model to consider multiple agents interacting with a system could capture interesting phenomena from game theory and evolutionary dynamics.

Overall, this approach presents a promising new method for model-based RL. By inserting symbols associated with action sequences into the observed history, the agent can model the dynamics of their interaction with a system as an ε-Machine and consider how changes to the policy change the overall dynamics. Contrast this with model-free approaches which are characterized by numerous incremental updates to the policy, leading to slow adaptation to new environments and slow convergence to a good policy. By directly modelling the impact of changes to the policy, the agent can immediately jump to a good strategy, and quickly adapt if the dynamic changes. Additionally, unlike other model-based approaches, the ε-Machine approach automatically defines the states of the system and minimizes the number of states, which should accelerate convergence to good policies.

References

1. Shalizi, C.R., Crutchfield, J.P. Computational Mechanics: Pattern and Prediction, Structure and Simplicity. *Journal of Statistical Physics* **104,** 817–879 (2001). https://doi.org/10.1023/A:1010388907793

2. Russell, S. J., Norvig, P. *Artificial Intelligence: A Modern Approach, 3rd edition*; Pearson: New York, NY, 2010; p. 858-861.

3. F. Wang, H. Zhang and D. Liu, "Adaptive Dynamic Programming: An Introduction," in *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 39-47, May 2009, DOI: 10.1109/MCI.2009.932261

4. Azar, A.T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H.A.; Ibrahim, Z.F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A.M.; Hameed, I.A.; Casalino, G. Drone Deep Reinforcement Learning: A Review. *Electronics* **2021**, *10*, 999. https://doi.org/10.3390/electronics10090999

5. Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484–489 (2016). https://doi.org/10.1038/nature16961

6. Still, S. Information-theoretic approach to interactive learning. *EPL* **85** 28005 (2009). https://doi.org/10.1209/0295-5075/85/28005

7. R C. Conant, W. R. Ashby (1970) Every good regulator of a system must be a model of that system. International Journal of Systems Science, 1:2, 89-97, DOI: 10.1080/00207727008920220