Learning to Plan Without a Planner

Jacob Andreas **Mitchell Stern** **Dan Klein**
Computer Science Division
University of California, Berkeley
{jda,mitchell,klein}@cs.berkeley.edu

1 Introduction

We describe a simple recurrent architecture for learning planning procedures from experience.

Agents acting under a fixed computational budget face fundamental tradeoffs when determining how to allocate computation. On one hand, determining the optimal action at the first timestep may require completely determining the agent's long-term behavior (e.g. when finding a path through a maze, Figure 2b). On the other hand, uncertainty about environment dynamics and observations may mean that nontrivial computational effort is required to compute the best next action in every state. Strategies for allocating computation thus form a spectrum, with *planning agents* (which do almost all of their computation initially) at one end and *reflex agents* (which distribute computation uniformly across timesteps) at the other.

The bulk of recent research on deep learning for interaction (in both reinforcement and imitation learning settings) has focused on reflex agents. In these approaches, a fixed-structure deep network maps from the current state of the environment to values [3] or probabilities [2] of subsequent actions. While generally effective, they have been less successful at taks that require long-term reasoning. Several recent papers propose coupling deep models with classical planning procedures like value iteration [4] and Kalman filtering [1]. But these techniques are not generally applicable: exact planning procedures of these kinds kind are known for only a simple world representations, and must be specified ahead of time. What we really want is an agent architecture with the capacity to plan, but which can *learn* the specific form of an appropriate planning algorithms from experience.

In this abstract, we present a set of preliminary experiments evaluating an extremely simple deep agent architecture which allocates generic computationa to a planning stage before it begins to act. We find that in both reinforcement imitation learning settings, across multiple domains, our approach outperforms a standard deep reflex architecture. We make two claims: (1) that given fixed computational resources, models implementing an "inference" precomputation phase outperform reflex agents; and (2) more strongly, that the inference mechanism imposes a useful inductive bias, and that shallow agents with n only for precomputation learn faster than deep agents that are allowed n steps of computation at *every* timestep.

2 Approach

Our approach is motivated by three basic observations about more explicit planning-based approaches to control:

1. Planning agents generally implement an initial, computation-heavy "inference" phase that produces a compact representation of a plan or policy.
2. Inference is usually performed by an *iterative* algorithm (e.g. a dynamic program or optimizer).
3. Once the agent begins to act, the effort required to predict the correct next action from any given state is small relative to the planning phase.

1st Workshop on Neural Abstract Machines & Program Induction (NAMPI), @NIPS 2016, Barcelona, Spain.

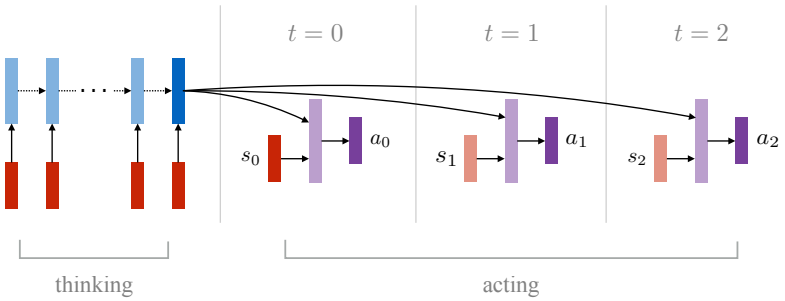


Figure 1: Model architecture. Our agent first implements a recurrent "thinking" phase (shown with dotted connection) which repeatedly receives the initial state as input and outputs a hidden "policy representation". Once the agent begins to act, it conditions its action at every timestep on both the current environment state and this representation.

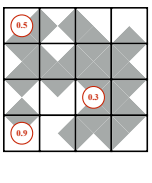
The agent architecture is depicted in Figure 1. It consists of a deep "thinking" phase implemented as an RNNfollowed by an "acting" phase during which the agent makes a shallow prediction based on both the current environment state and the output of the thinking phase. At each step during the acting phase, the final hidden state computed during the thinking phase is concatenated together with features for the current environment state; these are then passed through a shallow feedforward network that maps to a distribution over actions (in discrete action spaces) or parameters for the action sampling distribution (in continuous spaces).

At training time, minibatches are constructed by randomly sampling a set of (initial state, current state, action, reward) experiences from replay memory. These experiences are provided by agent rollouts (in the RL setting) or from a task-specific optimal planner (in the imitation setting). For each individual sample, the model is run forward and used to compute either a Q-value or a distribution over next actions; the appropriate loss is then backpropagated all the way through the network.

3 Experiments

We present experiments for two simple problems: a maze world and lattice world. In the maze world (Fig 2b), the model is presented with an image of a maze, as well as a start position and a goal position in absolute (x, y) coordinates. The action space consists of all real (x, y) values, and the model must predict a walk through the maze that reaches the goal cell without crossing any walls. In the lattice world (Fig 2a), the model is presented with a representation of a discrete, partially connected grid. The lattice world has four navigation actions corresponding to the four directions, as well as an "exit" action that causes the agent to receive the reward associated with the given cell and end the game. Each cell is associated with features that describe which of the navigational actions are available, the reward for taking the exit action in the cell, and an indicator feature on whether or not it is currently occupied by the agent.

Results are shown in Figure 2a–b. It can be seen that models implementing a planning phase outperform (sometimes significantly) straightforward reflex models, even when those models are as deep as the entire planning phase itself. While these results are preliminary, they suggest that for problems requiring long-term reasoning, task-specific inference algorithms can be learned end-to-end, and more generally that deep RL agents benefit from the ability to think before they act.



(a) (b)

Model	d_t	d_a	r_{re}	r_{im}
Reflex	-	1	45.0	53.9
Reflex	-	2	44.6	56.8
Reflex	-	4	48.3	57.5
Planner	4	1	51.9	59.6
Planner	8	1	53.6	60.4

(c)

Model	d_t	d_a	r_{im}
Reflex	-	8	59.2
Reflex	-	12	46.8
Planner	8	4	68.2

(d)

Figure 2: (a) The lattice world, with a discrete action space. (b) The maze world, with a continuous action space. (c) Experimental results for the lattice world. d_t is the number of layers in the "thinking" module, d_a is the number of layers in the "acting" module. r_{re} is the best reward accrued when training via reinforcement learning, while r_{im} is the best reward accrued during imitation learning. (d) Results for the maze world.

References

[1] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop kf: Learning discriminative deterministic state estimators. *arXiv preprint arXiv:1605.07148*, 2016.

[2] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[4] Aviv Tamar, Sergey Levine, and Pieter Abbeel. Value iteration networks. *arXiv preprint arXiv:1602.02867*, 2016.