Hierarchy, Behavior, and Off-policy Learning

Rich Sutton
University of Alberta

Outline

- a "micro-scale model" of cognition, in which abstractions play <u>no role</u> in producing behavior
- abstraction in state and time can be supported by options, but off-policy learning is required
- a new actor-critic-advantage algorithm for off-policy learning

---

- is hierarchical behavior a "user illusion"?
  - is it something we use it to explain our behavior, to others and to ourselves,
  - but not what are brains are really doing?
- or is it a real phenomena involved in every muscle we twitch?

Working hypothesis: Hierarchy and abstraction play *no role* in producing behavior

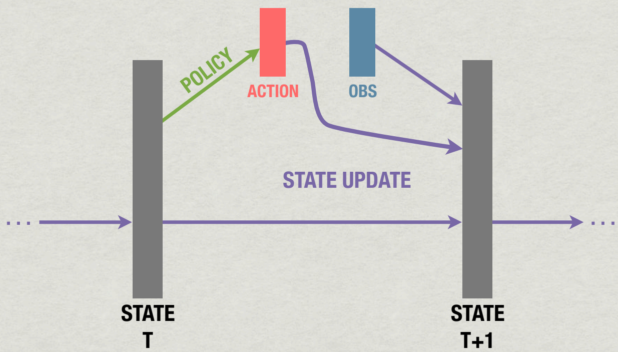
- there is no current option
- no goal stack
- no hierarchical execution
- no execution of high-level anything, ever
- <u>all execution is at a very low level (say 100hz)</u>

- some definitions:
  - <u>action</u> = lowest level action, 100hz
  - <u>observation</u> = lowest level sensation, 100hz
  - <u>state</u> = some representation/memory of the state of the world, updated at 100hz
  - <u>policy</u> = the mapping from state to action used to produce behavior, at 100hz

---

- some definitions:
  - <u>action</u> = lowest level action, 100hz
  - <u>observation</u> = lowest level sensation, 100hz
  - <u>state</u> = some representation/memory of the state of the world, updated at 100hz
  - <u>policy</u> = the mapping from state to action used to produce behavior, at 100hz

on every step, 100hz



Abstractions are used only for *changing the policy*

- by learning
- by planning
- abstractions are also used in the design of the state representation
- but in the end, to produce behavior, there is just a low-level policy

Outline

- a “micro-scale model” of cognition, in which abstractions play no role in producing behavior
- abstraction in state and time can be supported by options, but off-policy learning is required
- a new actor-critic-advantage algorithm for off-policy learning

Definitions re: options

- option = a way of behaving that terminates when one of a set of states is reached
- defined entirely in low-level terms (100hz)
- actions are a special case of options
- option outcome = how the option terminates
  - what state?
  - how much reward along the way?

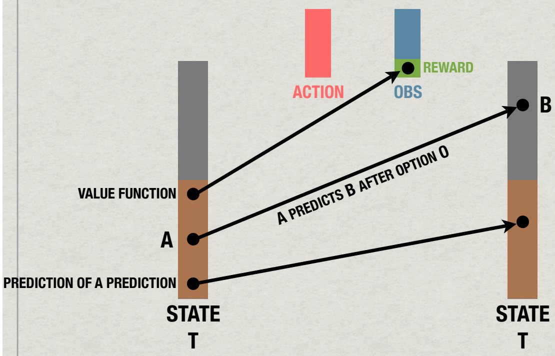
Option models as world knowledge

- option model = a mapping from states to predicted outcomes for some option
- each option model is a tiny abstract model of the world
  - if i tried to, could i open the door?
  - if i dropped this, would it make a sound?
  - if i waited, would this talk ever end?
  - if i tried to sit, would i fall on the floor?

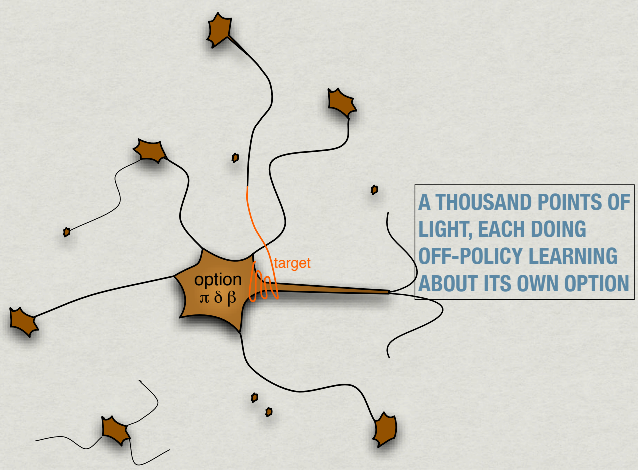
Option models as state representations

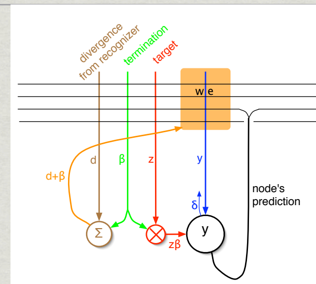
- option models are *predictions* of option outcomes
- such predictions can make great *abstract state variables*
  - if i opened the box, what would i see?
  - if i shifted lanes, would i hit another car?
  - if i ring Joe’s room, will he answer?
  - option models are PSRs (Littman et al., 2002) on steroids

on every step, at 100hz



- everything is still running at 100hz
- all options, option models, predictive state representations can be learned off-policy, in parallel, at 100hz
- even planning can run at 100hz
  - Dyna strategy: plan by learning on simulated transitions
  - use option models to generate transitions from the beginning to the end of options
  - long/variable time spans are reduced to single steps
- a parallel machine, running at the smallest time scale, yet always learning and thinking about large-scale behavior and abstract states





$$\dot{y} = u(a, o, y, \vec{w})$$

$$\dot{w}_i = \alpha\, \delta\, e_i$$

$$\delta = \dot{y} + z \cdot \beta$$

$$\dot{e}_i = \frac{\partial \dot{y}}{\partial w_i} - (d + \beta)\, e_i$$

CIRCUIT DIAGRAM FOR AN OFF-POLICY LEARNING ALGORITHM

CONTINUOUS-TIME EQUATIONS

UPDATE PROCESS

LEARNING (WEIGHT UPDATE)

LOCAL TD ERROR

ELIGIBILITY TRACES

- all this presumes we can do *off-policy*, *intra-option* learning with *function approximation*
- off-policy learning = learning about one policy while following another
  - we must learn off-policy in order to learn efficiently about options
  - you can only behave one way, but you want to learn about many different ways of behaving
- intra-option learning = learning about an overall option while only doing per-time-step operations
- function approximation = generalizing across states

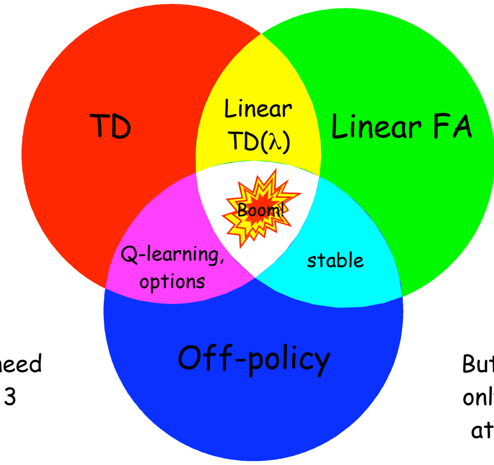
Do we know how to do off-policy learning?

- there are known, sound, off-policy learning methods
  - based on importance sampling (Precup et al, 2000)
  - based on averagers (Gordon, 1995)
- but they learn much more slowly than seems necessary
- and they are not “elegant”

RL Algorithm Space

Tsitsiklis & Van Roy 1997
Tadic 2000

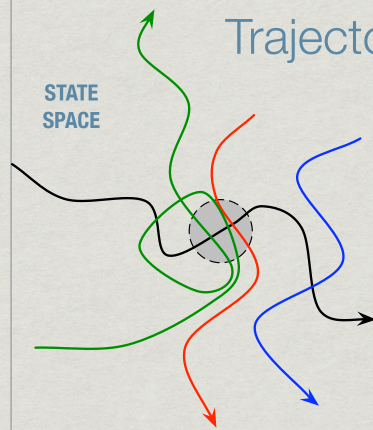
Baird 1995
Gordon 1995
NDP 1996



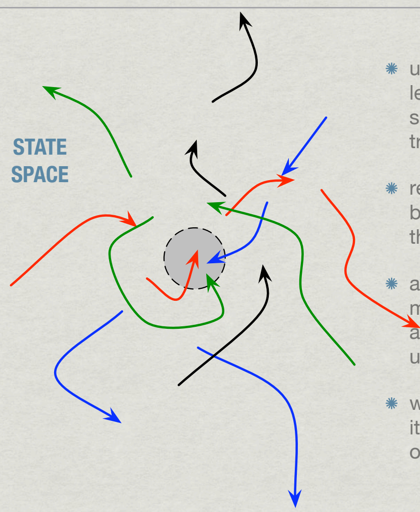
We need all 3

But we can only get 2 at a time

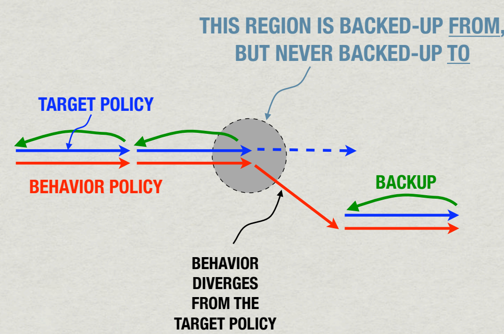
Trajectories are good

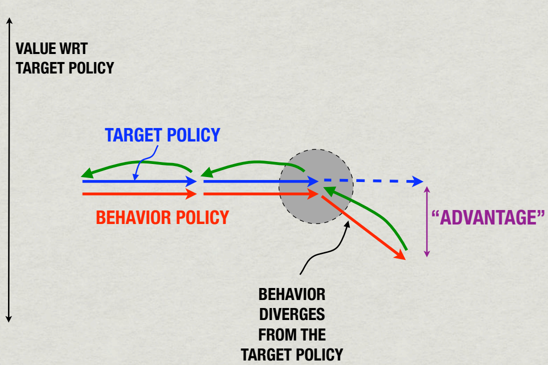


- under on-policy training, learning occurs along whole trajectories
- each region of state space is exited the same number of times as it is entered
- each region's estimated value is corrected as many times as it is used



- under off-policy training, learning occurs along segments of broken trajectories
- regions of state space may be entered more times than they are exited
- a region's estimated value may be used many times as a target, yet rarely be updated itself
- with function approximation, its estimate can get severely out of wack





The OPACA algorithm

- Off-Policy Actor-Critic-Advantage algorithm
- advantages: $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$
- the advantages are estimated using a third independent linear function approximator (in addition those for the actor and the critic)

OPACA alg. structures

Actor:

$$\pi(s,a,\theta) = \frac{e^{\theta^\top \phi(s,a)}}{\sum_b e^{\theta^\top \phi(s,b)}} \qquad \theta, \phi \in \Re^m$$

Critic:

$$V_t(s) = \mathbf{v}_t^\top \mathbf{f}(s) \qquad \mathbf{v}, \mathbf{f} \in \Re^n$$

Advantages:

$$A_t(s,a) = \mathbf{w}_t^\top \psi(s,a) \qquad \mathbf{w} \in \Re^m$$

using the actor-compatible feature vectors:

$$\psi(s,a) = \phi(s,a) - \sum_b \pi(s,b)\phi(s,b)$$

OPACA learning rule

Leading to an advantage-based TD error:

$$\begin{aligned} \delta_t &= r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) - A_t(s_t,a_t) \\ &= r_{t+1} + \gamma \mathbf{v}_t^\top \mathbf{f}(s_{t+1}) - \mathbf{v}_t^\top \mathbf{f}(s_t) - \mathbf{w}_t^\top \psi(s_t,a_t) \end{aligned}$$

and to one-step TD updates:

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \alpha \delta_t \mathbf{f}(s_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \delta_t \psi(s_t,a_t)$$

- so far we have only been able to prove convergence for the table-lookup case

Conclusions (1)

- it is perfectly feasible to generate behavior that is informed by abstract, higher-level considerations without using anything high-level at execute-time
- this makes for a simple, uniform, distributed architecture with local communication
- options can support abstraction in state and time, but require off-policy methods for efficient learning
- the search for an elegant off-policy algorithm continues

The “micro-scale model” of cognition and behavior

- in which the brain is viewed as an “experience engine,” humming along at 100hz,
- rapidly responding and predicting, and changing its responses and predictions
- creating and “compiling away” abstractions for immediate recall
- a demon model. a million recognizers, each watching for a single aspect of experience, yelling out their prediction of it