Approximate/Reinforcement Learning Approaches to POMDPs

Ron Parr

CSCI 2951-F

Brown University

Overview

- Value function based methods methods
- Policy search
- Augmented state methods

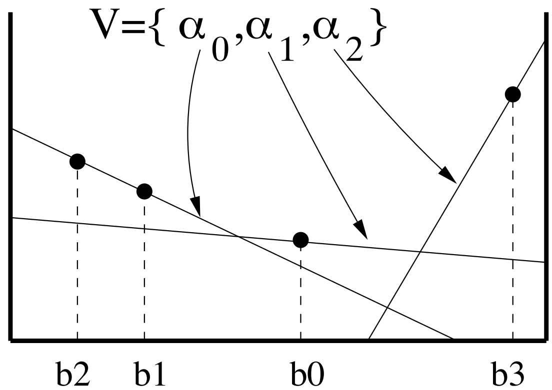
Review

- Finite horizon POMDP value function is piecewise linear and convex (for arbitrary horizon lengths)
  - Max over a set of "alpha" vectors
  - Each vector corresponds to a conditional plan
- Number of pieces can grow exponentially
- Hard to solve problems with more than high 1's or low 10's of states 😱

Point based methods

- Instead of generating ALL α vectors at each iteration, generate a subset
- Every α vector would still be a valid conditional plan
- Value function would lower-bound the true value function
- Point based algorithms generate α vectors that are optimal for only a finite set of points, rather than for the entire belief space

Visualizing PBVI (figure from Pineau et al.)



Questions

- How do we pick the points?
- How do we find the optimal α vector for each point?

Picking Points

- Typically done heuristically
- Exploration from initial dist. finds a set of reachable belief states
- Reasonable if start dist. is known and/or entire belief space is not reachable (exact POMDP algorithms may be working too hard)

PBVI performance (figure from Pineau et al.)

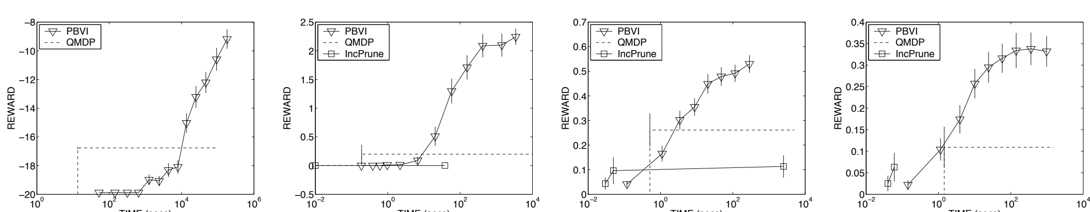


Figure 3: PBVI performance for four problems: Tag(left), Maze33(center-left), Hallway(center-right) and Hallway2(right)

QMDP is an approximation method that uses 1 a vector per action at all iterations.
Incremental Pruning was one of the best exact methods at the time.

PBVI limitations

- Fixed representation size was not adaptive to complexity of value fn.
- Only as fast as value iteration, which converges asymptotically
- Not monotonic: Can't guarantee values of all b increase at every iteration
- Is there a way to get the benefits of policy iteration?

Point Based Policy Iteration

- Combines policy iteration with point based methods
- Main idea:
  - Maintain a finite state machine (FSC) as the policy
  - Evaluate the FSC
  - Do one step of PBVI
  - Use output of PBVI to improve the FSC
  - Repeat

Limitations of Point Based Methods

- Still can be slow
- Assumes a known model:
  - At planning time
  - At execution time

Value function-based RL for POMDPs

- Since a POMDP is a continuous state MDP, why not use value function approximation on the continuous state?
- Many early efforts did this, e.g., RP's second publication from grad school
- Problems:
  - Still requires a model to update the belief state
  - Problems with huge state spaces have huge belief states
- Solution(?): Use a compressed belief state, e.g., Bayes net, but this still requires a model, and an efficient way of updating the belief state

Policy Search for POMDPs

Policy Search

- Recall policy gradient (figure from Sutton & Barto):

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta}), \forall a \in \mathcal{A}, s \in \mathcal{S}, \boldsymbol{\theta} \in \mathbb{R}^d$
Initialize policy parameter $\boldsymbol{\theta}$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  For each step of the episode $t = 0, \ldots, T-1$:
    $G \leftarrow$ return from step t
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t,\boldsymbol{\theta})$

- This still works even if Markov property is violated, but…

Naïve policy search in POMDPs

- Policy search "works", but policy is limited to a mapping from observations to actions
- Doesn't directly address the partial observability issue
- At best can randomize actions to avoid losses from state confusion

Policy search with FSCs

- Create a random Finite State Controller
- Make transition probabilities and action probabilities tunable parameters
- Use policy gradient methods to tune both of these
- Cool idea that has been rediscovered many times over the years
- Can be tricky to get working in practice for large problems

Policy Search in POMPs summary

- Advantages:
  - Does not require knowledge of the model
  - Does not need to maintain a belief state

- Disadvantages:
  - Many of the challenges of policy gradient methods:
    - Local optima
    - Variance in the gradient estimate
    - Slow
  - Estimating a value function baseline to reduce variance is also subject to state aliasing/partial observability issues

Augmented State Methods

Augmented state

- POMDPs are tricky b/c process is not Markovian in the observation
- Rather than change the algorithm, why not change the representation?

- Advantage: Get to run regular MDP algorithms on the new state

- Challenge: How to do this

Finite History Window

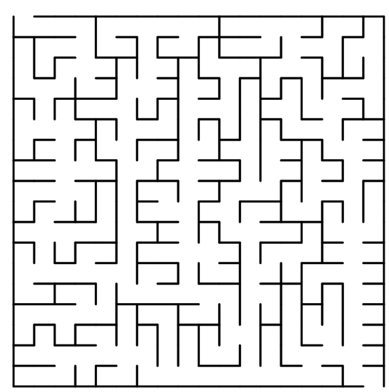
- Problem might not be Markovian in current observation, but
- Perhaps it is Markovian if we augment the state to include a k-step window of previous states – see, e.g., DQN for Atari
- Advantages:
  - Obviously the right thing to do if you can afford to do it
  - Simple
- Disadvantages:
  - For n states, d step history, state space grows with n^d
  - Not always obvious how large to make d

History Trees

- Long history windows probably waste a lot of effort tracking irrelevant info:
  - Many states may have unique/unambiguous observations
  - No need to remember history when we see these
- History trees define state as a variable length vector of previous states and actions sufficient to ensure Markov property
- In practice:
  - Collect statistics on histories
  - When violations of Markov property are detected, extend history
- See e.g., McCallum '95, "Reinforcement Learning with Selective Perception and Hidden State"

History tree example

- Robot going through maze
- Suppose two intersections look alike
- History tree can be used to remember how the robot **got to** the intersection, to help distinguish between similar states
- How to discover this:
  - Need to collect statistics on all possible extensions of current histories
  - When next states or next utilities diverge based upon different extensions of the history, grow the history



https://commons.wikimedia.org/wiki/File:Prim_Maze.svg

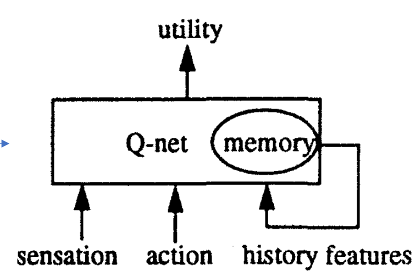
History Tree Pros and Cons

- Works very well in some problems where short(ish) histories are sufficient to recover the Markov property
- More efficient than finite window methods
- Limitations:
  - May need to collect a lot of data (for long histories)
  - Can be hard to determine when to augment histories if there is a lot of noise
  - Myopic/greedy (will miss if you need to remember something from 20 steps in the past, and remembering something 1…19 steps in the past doesn't help.)

Augmented state w/Function Approximation

- Idea: Use function approximation to learn how to augment the state "automatically" with a recurrent neural network (RNN)
- Old idea (at least as far back as Lin in the 90's)

From page 109 of Long Ji Lin's Ph.D. thesis (CMU 1993)!

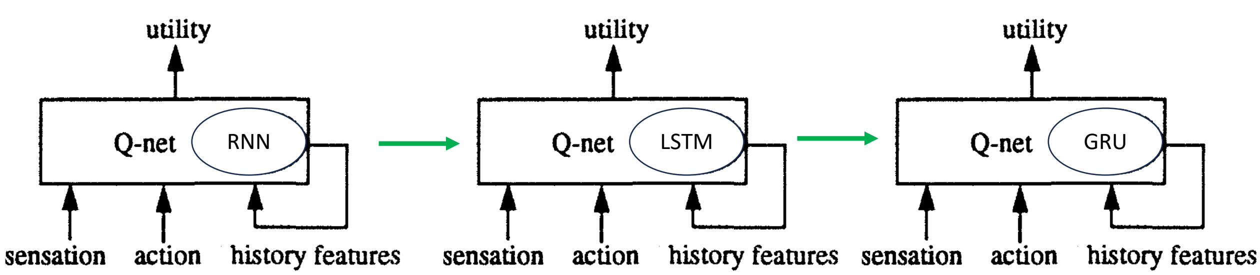


Learned, Augmented State

- Cool idea
- Agent is essentially learning an encoded belief state and method for updating the belief state simultaneously
- Historically, such efforts were plagued by the difficulties associated with RNNs in general:
  - Convergence concerns
  - Difficulty with long term memory

Learned, augmented states strike back

- LSTMs are a type of RNN designed to maintain long term memory
- GRUs are a simplification of LSTMs that *may* work better



Some references

- Jozefowicz et al. 2015 compare different memory architectures (LSTM, GRU,…) in general (not for RL)

- Ni et al. 2022 claim GRUs are a “strong baseline” for RL in POMDPs, (but use a particular type of problem to make this claim)

What about Transformers?

- Deep Transformer Q-networks (Esslinger et al. 22) one of the more compelling efforts to use transformers in RL
- Use attention transform observations from a finite window of the past into an encoded state
- Enjoys advantages of transformers:
  - Quadratic in size of window, rather than exponential
  - No decay/forgetting within window size

- Will transformers overtake RNNs for POMDP RL?

POMDP approximation summary

- Known model of moderate size: Use point based methods, or value function approximation on a (compressed?) state
- Modest history dependence: Augment state, possibly using a learning method to discover required augmentation (e.g., history trees)
- Unknown model, unknown (bounded?) history dependence:
  - Deep learning with LSTM/GRU or similar methods to learn representation
  - Up-and-coming transformer approaches?