Describing Value Iteration Networks

Kiran Vodrahalli
April 16, 2018

Main contributions

- The Value Iteration Network uses an *approximation of Value Iteration Algorithm*
  - IN CONTRAST: Other work in deep RL focuses on *approximation of policy* directly
- Introduces framework for learning policies which depend on value functions of approximate MDPs
- ``Learning to plan'' and generalizing to many problem instances in an environment
- Clever model training setup: Can express Value Iteration as an end-to-end, differentiable process (convolutional neural network)

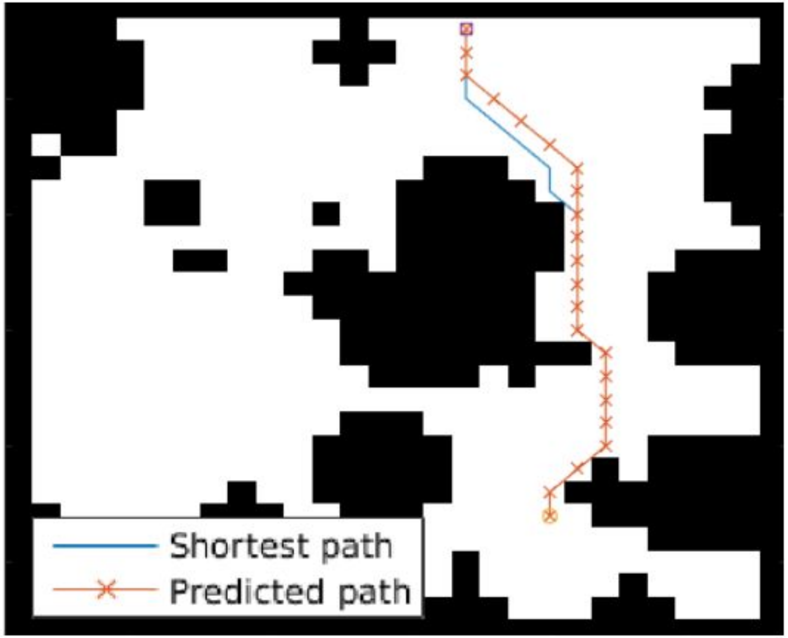
Problem setting

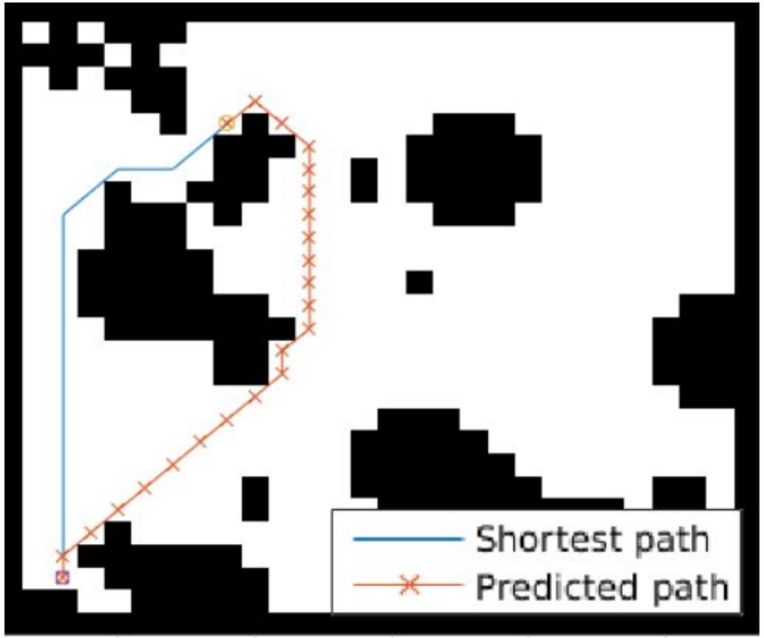
- 1) Imitation learning
    - We get to see expert actions for every state
    - Similar to supervised learning
- 2) Standard RL setup
    - Access to environment rewards
- Key point: *Generalize* on classes of environments!
    - Solve robot planning problems in a variety of locations

Concrete Examples (Goal: Shortest path)

- Gridworld
- Mars Rover
- Wikipedia Search

Gridworld (example to keep in mind)





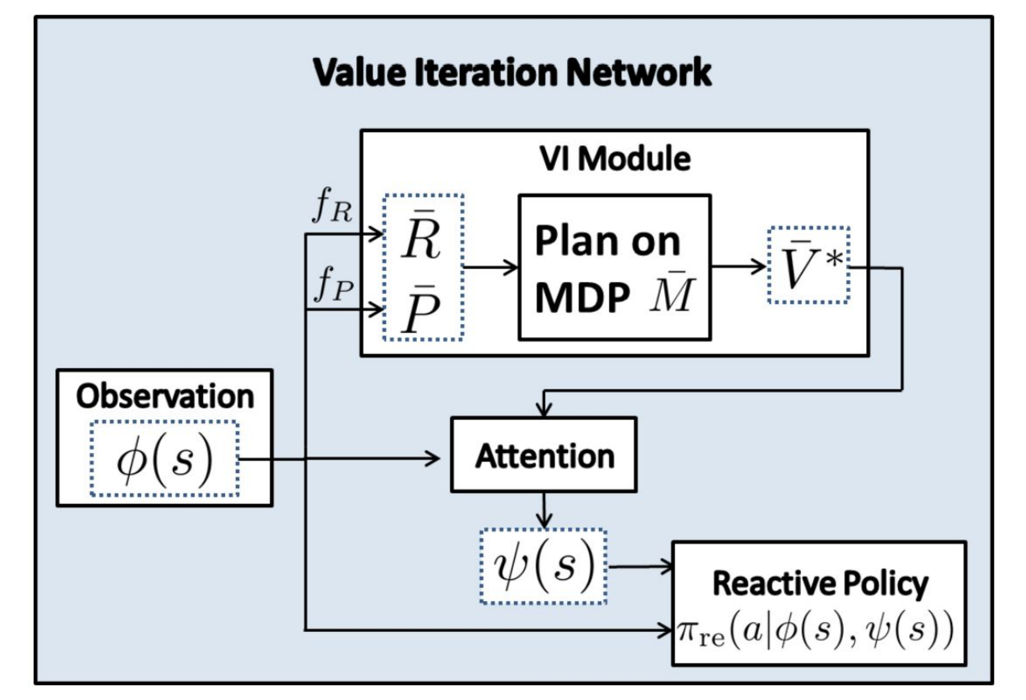
Recall Value Iteration

- Given MDP (S, A, R, P) (states, actions, $R(s, a)$ = reward, $P(s' \mid s, a)$ = transitions)
- Value Iteration Algorithm Recap:
    - Goal: Calculate the value function $V(s, \pi^*)$, the expected discounted reward under optimal policy π^* from state s.
    - We will learn $V_n(s)$ recursively so that $V_n \to V^* = V(s, \pi^*)$ as $n \to \infty$
    - Define $Q_n(s, a) = R(s, a) + \gamma \sum_{s'} P(s' \mid s, a)\, V_n(s)$.
    - Update $V_{n+1}(s) = \max_a Q_n(s, a)$
    - $\pi^* = \operatorname{argmax}_a Q_{infinity}(s, a)$

Key Idea: Value Iteration Network

- We don’t know the true MDP of the environment we are in.
- Observe state $\phi(s)$
- Learn a fake MDP (perhaps same state and action space, but parametrized reward $f_R(\phi(s))$ and parametrized transition $f_P(\phi(s))$)
- Value Iteration Module (approximate value iteration) $\rightarrow$ value function of fake MDP
- Use value of fake MDP and a subset of $\phi(s)$ to parametrize final policy
    - $\Psi(s)$ = an attention mechanism applied to $\phi(s)$

VIN Diagram



Value Iteration Network
VI Module
f_R
f_P
$\bar{R}$
$\bar{P}$
Plan on MDP $\bar{M}$
$\bar{V}^*$
Observation
$\phi(s)$
Attention
$\psi(s)$
Reactive Policy
$\pi_{\mathrm{re}}(a|\phi(s), \psi(s))$

Value Iteration Module

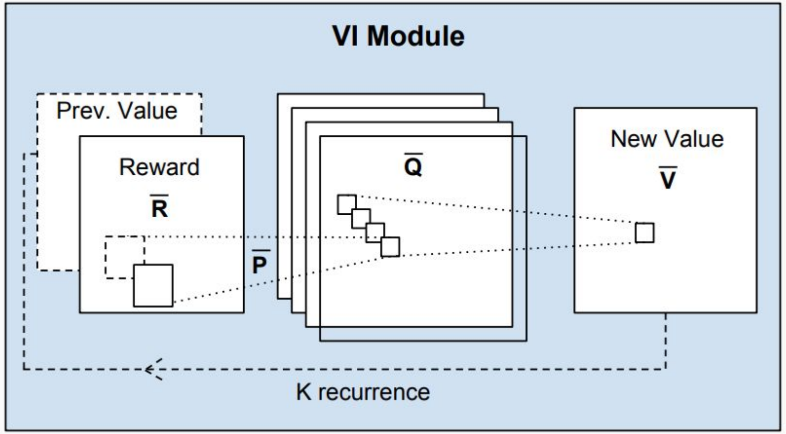
Value iteration

K iterations of:

$$\bar{Q}_n(\bar{s},\bar{a}) = \bar{R}(\bar{s},\bar{a}) + \sum_{\bar{s}'} \gamma \bar{P}(\bar{s}'|\bar{s},\bar{a})\bar{V}_n(\bar{s}')$$

$$\bar{V}_{n+1}(\bar{s}) = \max_{\bar{a}} \bar{Q}_n(\bar{s},\bar{a}) \quad \forall \bar{s}$$

Convnet



Ex: GridWorld

- In GridWorld, $f_R(\phi(s))$ is a CNN from the image input with trainable parameters.
- $f_P(\phi(s))$ are defined as 3x3 convolution kernels in the VI module.
  - Note they don't depend on s
- The attention mechanism selects the output of the VI module corresponding to the current state
- The final policy is a fully-connected NN mapping the output of the attention mechanism to a softmax distribution over actions

Remarks on VI Module

- Extremely convenient in the GridWorld setup (2D convolution)
  - Generalizes to other settings with small state spaces
  - Especially if #states s.t. P(s' | s, a) > 0 is small (sparse == attention)
- Max-pooling happens across actions
- Unroll K times: ensure that one can reach the goal in state space with K actions.
- Linear convolution operation:

$$\bar{Q}_{\bar{a},i',j'} = \sum_{l,i,j} W^{\bar{a}}_{l,i,j} \bar{R}_{l,i'-i,j'-j}$$

What was learned?

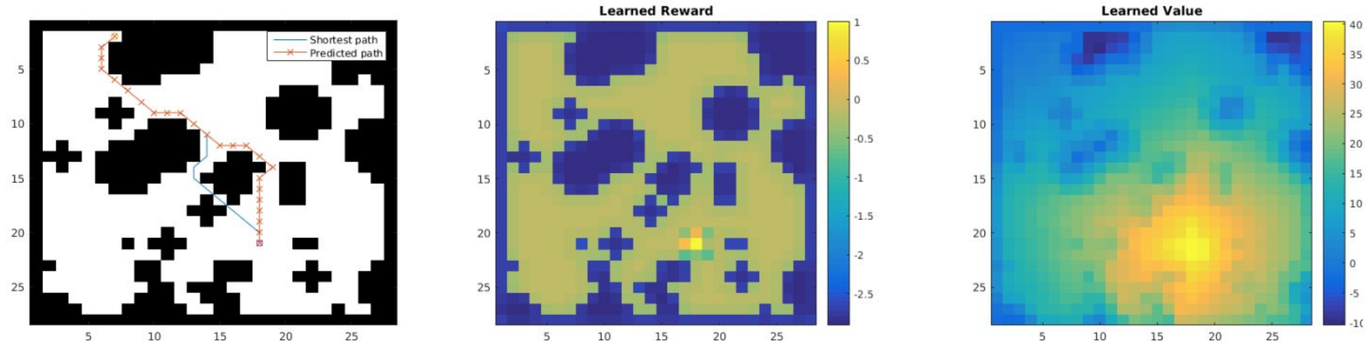


Figure 5: Visualization of learned reward and value function. Left: a sample domain. Center: learned reward f_R for this domain. Right: resulting value function (in VI block) for this domain.

Extensions

- Hierarchy
  - Different levels of resolution
  - Input is downsampled ⇒ grainy resolution input to Vin #1
  - output of Vin #2 is upsampled and given as addt'l input into VIN #2 state
  - Repeat as many times as you like
- Continuous State/Action space
  - Discretize the view

Brief summary of experimental results

- Imitation learning experiments: See a bunch of expert actions at various states, directly backprop
- RL experiments: use policy from VIN in RL as per usual, use sampled reward to backprop
- VIN typically did better compared to ``reactive policies'', which have no ``learning to plan'' component
    - No functions $f_R(\phi(s))$ and $f_P(\phi(s))$ to learn
    - In some experiments, barely did better; others, much better