Value Iteration Networks

Aviv Tamar
Joint work with Pieter Abbeel, Sergey Levine, Garrett Thomas, Yi Wu

Presented by: Kent Sommer
Most content directly from Aviv Tamar's 2016 presentation

April 25, 2017

Korea Advanced Institute of Science and Technology

INTRODUCTION

- Goal: autonomous robots

Robot, bring me the milk bottle!

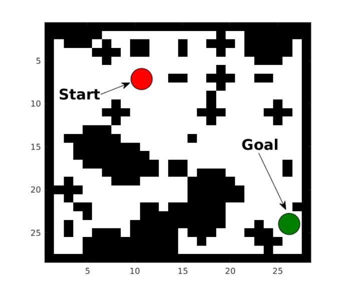
http://www.wellandgood.com/wp-content/uploads/2015/02/Shira-fridge.jpg

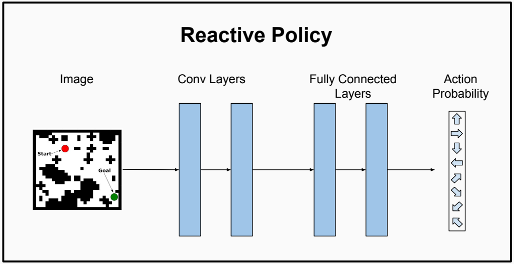
- Solution: RL?

- Deep RL learns policies from high-dimensional visual input[1,2]
- Learns to act, but does it **understand**?
- A simple test: generalization on grid worlds

[1] Mnih et al. Nature 2015
[2] Levine et al. JMLR 2016

INTRODUCTION





Why don't reactive policies generalize?

- A sequential task requires a planning computation
- RL gets around that – learns a mapping
    - State $\rightarrow$ Q-value
    - State $\rightarrow$ action with high return
    - State $\rightarrow$ action with high advantage
    - State $\rightarrow$ expert action
    - [State] $\rightarrow$ [planning-based term]
- Q/return/advantage: planning **on training domains**
- New task – need to re-plan

In this work:

- Learn to plan
- Policies that generalize to unseen tasks

BACKGROUND

Planning in MDPs

- States $s \in \mathcal{S}$, actions $a \in \mathcal{A}$
- Reward $R(s, a)$
- Transitions $P(s'|s, a)$
- Policy $\pi(a|s)$
- Value function $V^\pi(s) \doteq \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s \right]$
- Value iteration (VI)

$$
\begin{aligned}
V_{n+1}(s) &= \max_a Q_n(s, a) \quad \forall s, \\
Q_n(s, a) &= R(s, a) + \gamma \sum_{s'} P(s'|s, a) V_n(s').
\end{aligned}
$$

- Converges to $V^* = \max_\pi V^\pi$
- Optimal policy $\pi^*(a|s) = \arg\max_a Q^*(s, a)$

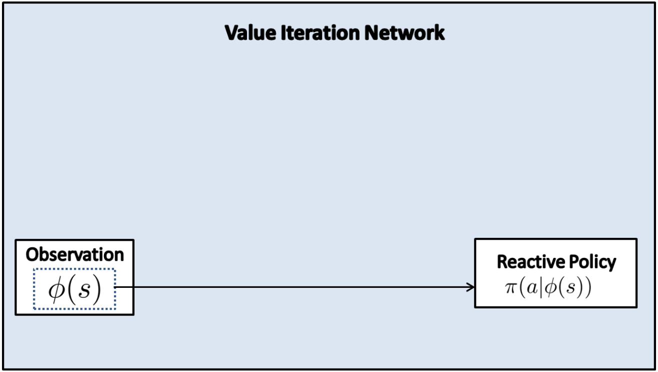
BACKGROUND

Policies in RL / imitation learning

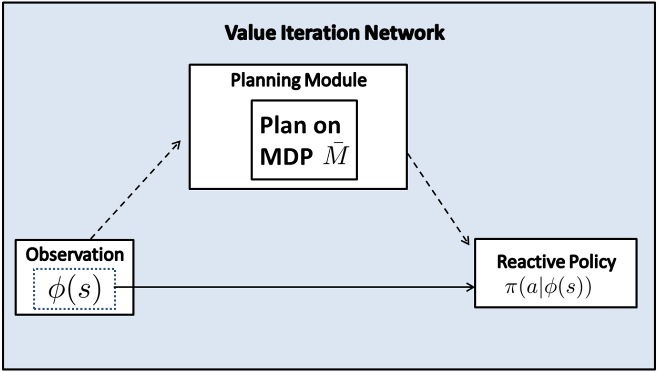
- State observation $\phi(s)$
- Policy: $\pi_\theta(a|\phi(s))$
  - Neural network
  - Greedy w.r.t. Q (DQN)
- Algorithms perform SGD, require $\nabla_\theta \pi_\theta(a|\phi(s))$
- Only loss function varies
  - Q-learning (DQN)
  - Trust region policy optimization (TRPO)
  - Guided policy search (GPS)
  - Imitation Learning (supervised learning, DAgger)
- Focus on policy representation
- Applies to model-free RL / imitation learning

A MODEL FOR POLICIES THAT PLAN

- Start from a reactive policy



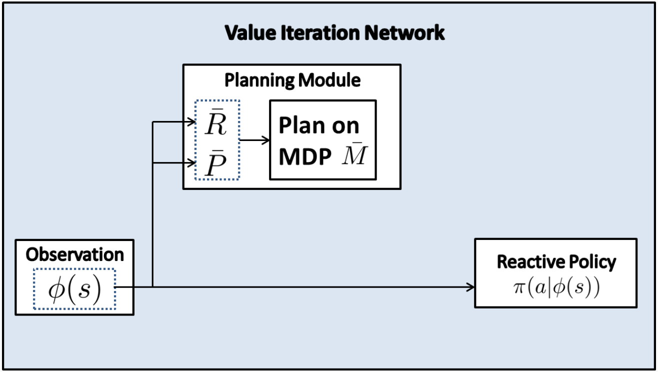
- Add an explicit planning computation
- Map observation to planning MDP $\bar{M}$



- Assumption: observation can be mapped to a useful (but **unknown**) planning computation

A PLANNING-BASED POLICY MODEL

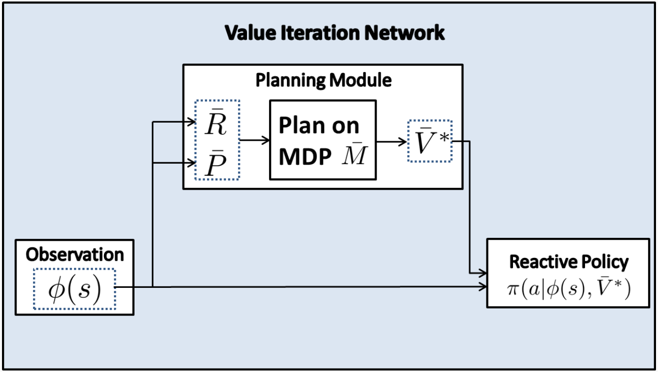
- NNs map observation to reward and transitions
- Later - learn these



How to use the planning computation?

A PLANNING-BASED POLICY MODEL

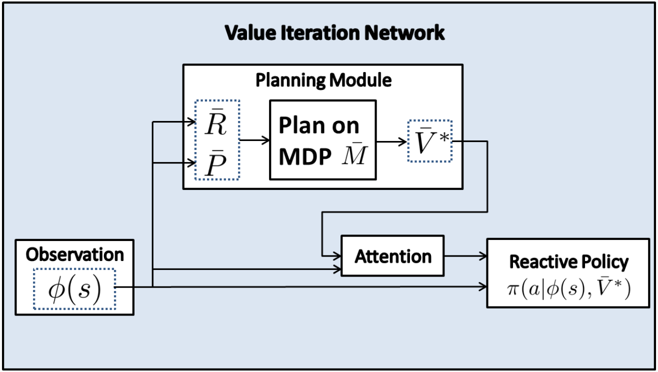
- Fact 1: value function = sufficient information about plan
- Idea 1: add as features vector to reactive policy



- Fact 2: action prediction can require only subset of $\bar{V}^*$

$$\pi^*(a|s) = \arg\max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$$

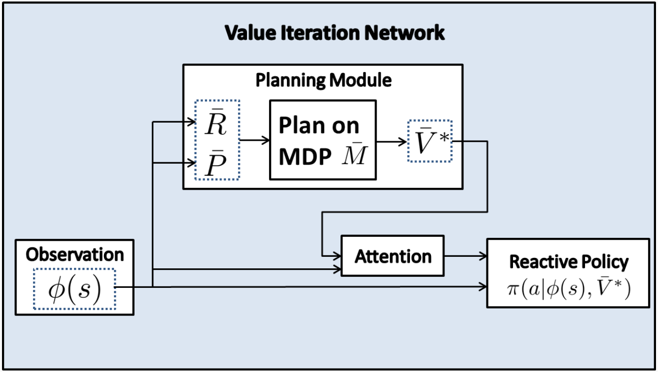
- Similar to attention models, effective for learning[1]



[1] Xu et al. ICML 2015

A PLANNING-BASED POLICY MODEL

- Policy is still a mapping $\phi(s) \rightarrow \text{Prob}(a)$
- Parameters θ for mappings $\bar{R}, \bar{P}$, attention
- Can we backprop?



How to backprop through planning computation?

VALUE ITERATION = CONVNET

VALUE ITERATION = CONVNET

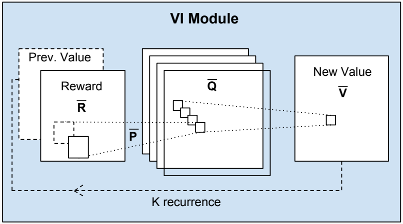
Value iteration

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

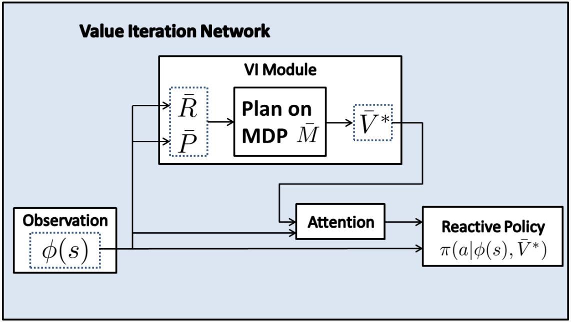


- $\bar{\mathcal{A}}$ channels in $\bar{Q}$ layer
- Linear filters $\Longleftrightarrow \gamma\bar{P}$
- Tied weights
- Channel-wise max-pooling

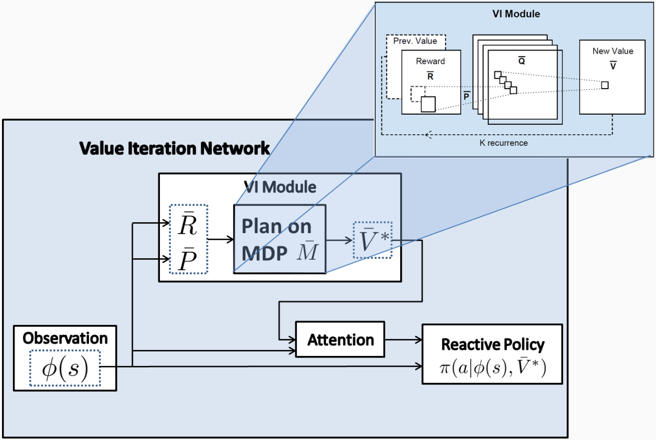
- Best for locally connected dynamics (grids, graphs)
- Extension – input-dependent filters

VALUE ITERATION NETWORKS

- Use VI module for planning



- Value iteration network (VIN)



EXPERIMENTS

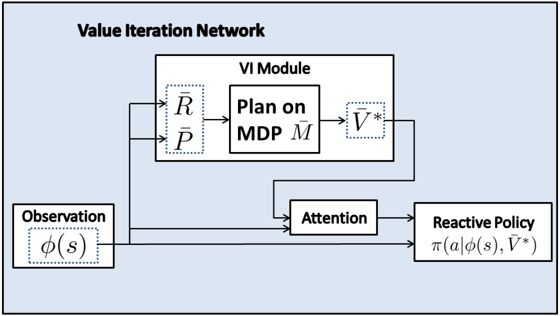
Questions

1. Can VINs learn a planning computation?
2. Do VINs generalize better than reactive policies?

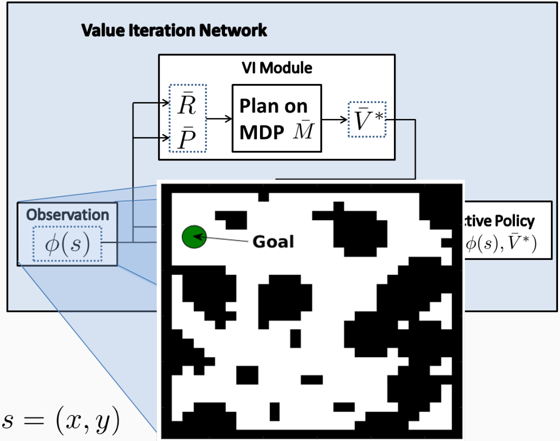
GRID-WORLD DOMAIN

- Supervised learning from expert (shortest path)
- Observation: image of obstacles + goal, current state
- Compare VINs with reactive policies

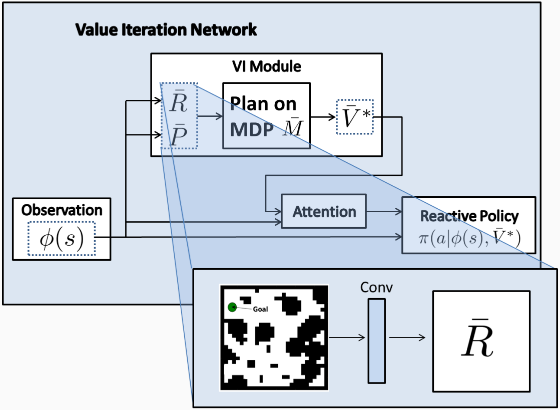
- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: 3×3 kernel
- Attention: choose $\bar{Q}$ values for current state
- Reactive policy: FC, softmax



- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: 3×3 kernel
- Attention: choose $\bar{Q}$ values for current state
- Reactive policy: FC, softmax

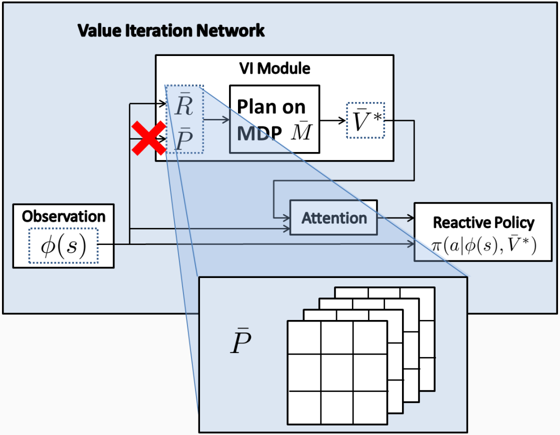


- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: 3×3 kernel
- Attention: choose $\bar{Q}$ values for current state
- Reactive policy: FC, softmax

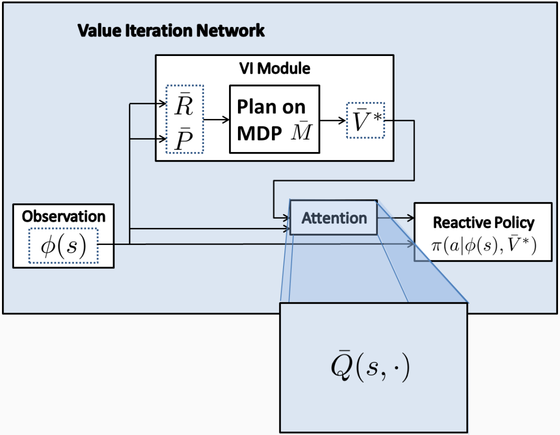


- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: 3×3 kernel

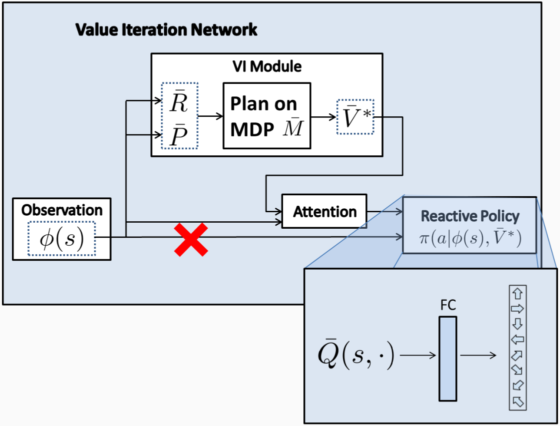
- Attention: choose $\bar{Q}$ values for current state
- Reactive policy: FC, softmax



- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: 3×3 kernel
- Attention: choose $\bar{Q}$ values for current state
- Reactive policy: FC, softmax



- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: 3×3 kernel
- Attention: choose $\bar{Q}$ values for current state
- Reactive policy: FC, softmax



Compare with:

- CNN inspired by DQN architecture[1]
    - 5 layers
    - Current state as additional input channel
- Fully convolutional net (FCN)[2]
    - Pixel-wise semantic segmentation (labels=actions)
    - Similar to our attention mechanism
    - 3 layers
    - Full-sized kernel – receptive field always includes goal

Training:

- 5000 random maps, 7 trajectories in each
- Supervised learning from shortest path

[1] Mnih et al. Nature 2015
[2] Long et al. CVPR 2015

Evaluation:

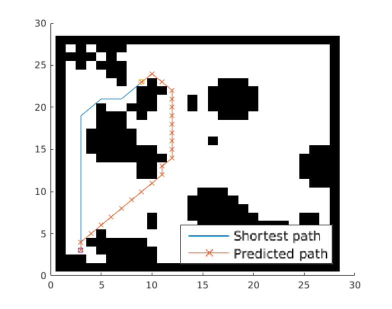
- Action prediction error (on test set)
- Success rate – reach target without hitting obstacles

Results:

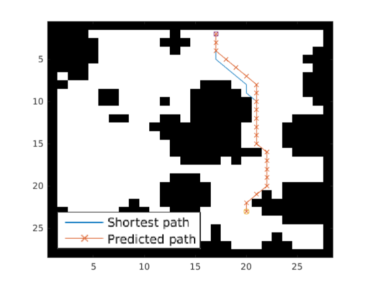
Domain	VIN		CNN		FCN	
	Prediction loss	Success rate	Pred. loss	Succ. rate	Pred. loss	Succ. rate
8×8	0.004	**99.6%**	0.02	97.9%	0.01	97.3%
16×16	0.05	**99.3%**	0.10	87.6%	0.07	88.3%
28×28	0.11	**97%**	0.13	74.2%	0.09	76.6%

VINs learn to plan!

Results:

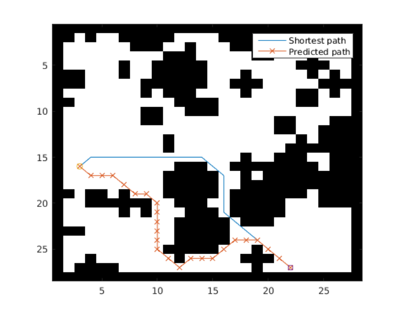


Results:

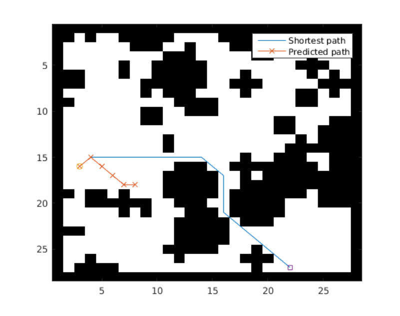


Results:

VIN

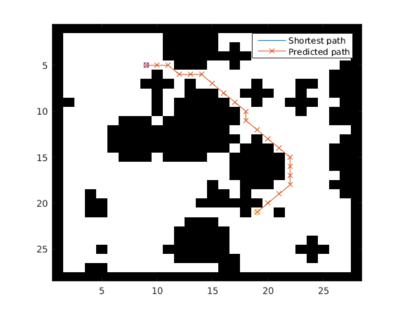


FCN

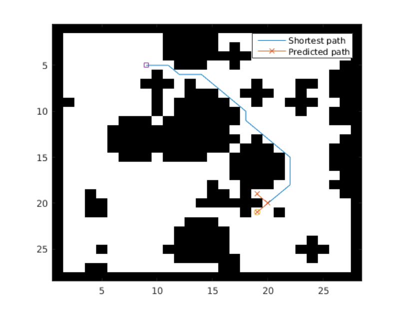


Results:

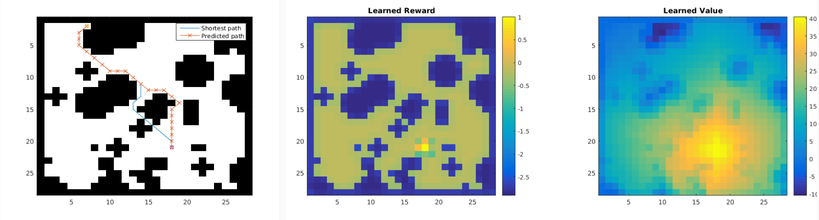
VIN



FCN



Results:


Shortest path
Predicted path
Learned Reward
Learned Value

SUMMARY & OUTLOOK

- Learn to plan → generalization
- Framework for planning based NN policies
    - Motivated by dynamic programming theory
    - Differentiable planner (VI = CNN)
    - Compositionality of NNs – perception & control
    - Exploits flexible prior knowledge
    - Simple to use

THANK YOU!