VALUE ITERATION NETWORKS

Aviv Tamar
Joint work with Pieter Abbeel, Sergey Levine, Garrett Thomas, Yi Wu
June 23, 2016

UC Berkeley
INTRODUCTION
∙ Goal: autonomous robots

Robot, bring me the milk bottle!

∙ Solution: RL?
· Deep RL learns policies from high-dimensional visual input\textsuperscript{1,2}
· Learns to act, but does it \textit{understand}?  
· A simple test: generalization on grid worlds

\textsuperscript{1}\textsuperscript{Mnih et al. Nature 2015}  
\textsuperscript{2}\textsuperscript{Levine et al. JMLR 2016}
INTRODUCTION

Reactive Policy

Image -> Conv Layers -> Fully Connected Layers -> Action Probability

Start

Goal
INTRODUCTION

Train

Goal

Start
Train
Observation: reactive policies do not generalize well
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 $V^\pi(s) = \mathbb{E}^\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s \right]$
- Value iteration (VI)

\[
V_{n+1}(s) = \max_a Q_n(s, a) \quad \forall s,
\]

\[
Q_n(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) V_n(s').
\]

- Converges to $V^* = \max_{\pi} V^\pi$
- Optimal policy $\pi^*(a|s) = \arg \max_a Q^*(s, a)$
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
A PLANNING-BASED POLICY MODEL

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

- 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?
Fact 1: value function = sufficient information about plan
Idea 1: add as features vector to reactive policy
A PLANNING-BASED POLICY MODEL

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

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

• Similar to attention models, effective for learning\(^1\)

\(^1\)Xu et al. ICML 2015
- Policy is still a mapping $\phi(s) \rightarrow \text{Prob}(a)$
- Parameters $\theta$ for mappings $\mathcal{R}, \mathcal{P}$, attention
- Can we backprop?

How to backprop through planning computation?
VALUE ITERATION = CONVNET
Value iteration

K iterations of:

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

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

Convnet

- \( \bar{A} \) channels in \( \bar{Q} \) layer
- Linear filters \( \leftrightarrow \gamma \tilde{P} \)
- Tied weights
- Channel-wise max-pooling

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

- Use VI module for planning
VALUE ITERATION NETWORK

- Value iteration network (VIN)
VALUE ITERATION NETWORK

- Just another policy representation $\pi_\theta(a|\phi(s))$
- That can learn to plan
- Train like any other policy!
- Backprop – just like a convnet
- Implementation – few lines of Theano code
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 $\tilde{Q}$ values for current state
· Reactive policy: FC, softmax
• VI state space: grid-world
• VI Reward map: convnet
• VI Transitions: $3 \times 3$ kernel
• Attention: choose $\bar{Q}$ values for current state
• Reactive policy: FC, softmax

$$s = (x, y)$$
GRID-WORLD DOMAIN

- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: 3 x 3 kernel
- Attention: choose \( \bar{Q} \) values for current state
- Reactive policy: FC, softmax
GRID-WORLD DOMAIN

- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: $3 \times 3$ kernel

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

• Attention: choose $\tilde{Q}$ values for current state
• Reactive policy: FC, softmax
GRID-WORLD DOMAIN

- VI state space: grid-world
- VI Reward map: convnet
- VI Transitions: $3 \times 3$ kernel

- Attention: choose $\tilde{Q}$ 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
GRID-WORLD DOMAIN

Evaluation:

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

Results:

<table>
<thead>
<tr>
<th>Domain</th>
<th>VIN</th>
<th>CNN</th>
<th>FCN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prediction loss</td>
<td>Success rate</td>
<td>Pred. loss</td>
</tr>
<tr>
<td>8 × 8</td>
<td>0.004</td>
<td>99.6%</td>
<td>0.02</td>
</tr>
<tr>
<td>16 × 16</td>
<td>0.05</td>
<td>99.3%</td>
<td>0.10</td>
</tr>
<tr>
<td>28 × 28</td>
<td>0.11</td>
<td>97%</td>
<td>0.13</td>
</tr>
</tbody>
</table>

VINs learn to plan!
GRID-WORLD DOMAIN

Results:
GRID-WORLD DOMAIN

Results:
GRID-WORLD DOMAIN

Results:

VIN

FCN
Results:

VIN

FCN
GRID-WORLD DOMAIN

Results:
Depth vs. Planning

- Planning requires **depth** – why not just add more layers?
- Experiment: untie weights in VINs
  - Degrades performance
  - Especially with less data

- The VI structure is important
GRID-WORLD DOMAIN

Training using RL

- Q-learning, TRPO\(^1\)
- Same network structure
- Curriculum learning for exploration
- Similar findings as supervised case

\(^1\)Schulman et al. ICML 2015
MARS-NAVIGATION DOMAIN
MARS-NAVIGATION DOMAIN

- Grid-world with natural image observations
- Overhead images of Mars terrain
- Obstacle = slope of 10° or more
- Elevation data not part of input
MARS-NAVIGATION DOMAIN

- Grid-world with natural image observations
- Overhead images of Mars terrain
- Obstacle = slope of 10° or more
- Elevation data not part of input
MARS-NAVIGATION DOMAIN

- Grid-world with natural image observations
- Overhead images of Mars terrain
- Obstacle = slope of 10° or more
- Elevation data not part of input
Same grid-world VIN, 3 layers in $\overline{R}$ convnet

<table>
<thead>
<tr>
<th></th>
<th>Pred. loss</th>
<th>Succ. rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIN</td>
<td>0.089</td>
<td>84.8%</td>
</tr>
<tr>
<td>Best achievable</td>
<td>-</td>
<td>90.3%</td>
</tr>
</tbody>
</table>

- Best achievable: train classifier with **obstacle labels**, predict map and plan
- VIN **did not** observe any labeled obstacle data
- Conclusion: can handle non-trivial **perception**
CONTINUOUS CONTROL DOMAIN
CONTINUOUS CONTROL DOMAIN

- Move particle between obstacles, stop at goal
- 4d state (position, velocity), 2d action (force)
- Input: state + low-res (16 × 16) map
CONTINUOUS CONTROL DOMAIN

- Move particle between obstacles, stop at goal
- 4d state (position, velocity), 2d action (force)
- Input: state + low-res (16 x 16) map
CONTINUOUS CONTROL DOMAIN

- VI state space: grid-world
- Attention: 5 × 5 patch around current state
- Reactive policy: FC, Gaussian mean output

\[ s = (x, \dot{x}, y, \dot{y}) \]
CONTINUOUS CONTROL DOMAIN

- VI state space: grid-world
- Attention: $5 \times 5$ patch around current state

- Reactive policy: FC, Gaussian mean output
CONTINUOUS CONTROL DOMAIN

- VI state space: grid-world
- Attention: $5 \times 5$ patch around current state
- Reactive policy: FC, Gaussian mean output
CONTINUOUS CONTROL DOMAIN

Compare with:

- CNN inspired by DQN architecture$^1,^2$
  - 2 conv layers + $2 \times 2$ pooling + 3 FC layers

Training:

- 200 random maps
- iLQG with unknown dynamics$^3$
- Supervised learning (equiv. 1 iteration of guided policy search)

---

$^1$Mnih et al. Nature 2015
$^2$Lillicrap et al. ICLR 2016
$^3$Levine & Abbeel, NIPS 2014
CONTINUOUS CONTROL DOMAIN

Evaluation:

· Distance to goal on final time

Results:

<table>
<thead>
<tr>
<th>Network</th>
<th>Train Error</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIN</td>
<td>0.30</td>
<td>0.35</td>
</tr>
<tr>
<td>CNN</td>
<td>0.39</td>
<td>0.58</td>
</tr>
</tbody>
</table>

![Error Histogram](image1)

![Graphs](image2)
WEB-NAV DOMAIN — LANGUAGE-BASED SEARCH
Navigate website links to find a query

The Enigma was used commercially from the early 1920s on, and was also adopted by the military and governmental services of a number of nations—most famously by Nazi Germany before and during World War II.
The mechanical parts act in such a way as to form a varying electrical circuit—the actual encipherment of a letter is performed electrically. When a key is pressed, the circuit is completed; current flows through the various components and ultimately lights one of many different lamps, indicating the output letter.
WEB-NAV DOMAIN

- Navigate website links to find a query

The Enigma was used commercially from the early 1920s on, and was also adopted by the military and governmental services of a number of nations—most famously by Nazi Germany before and during World War II.

The mechanical parts act in such a way as to form a varying electrical circuit—the actual encipherment of a letter is performed electrically. When a key is pressed, the circuit is completed; current flows through the various components and ultimately lights one of many different lamps, indicating the output letter.

Navigate website links to find a query

The Enigma was used commercially from the early 1920s on, and was also adopted by the military and governmental services of a number of nations—most famously by Nazi Germany before and during World War II. The mechanical parts act in such a way as to form a varying electrical circuit—the actual encipherment of a letter is performed electrically. When a key is pressed, the circuit is completed; current flows through the various components and ultimately lights one of many different lamps, indicating the output letter.
Navigate website links to find a query

The Enigma was used commercially from the early 1920s on, and was also adopted by the military and governmental services of a number of nations—most famously by Nazi Germany before and during World War II. The mechanical parts act in such a way as to form a varying electrical circuit—the actual encipherment of a letter is performed electrically. When a key is pressed, the circuit is completed; current flows through the various components and ultimately lights one of many different lamps, indicating the output letter.
The Enigma was used commercially from the early 1920s on, and was also adopted by the military and governmental services of a number of nations—most famously by Nazi Germany before and during World War II. The mechanical parts act in such a way as to form a varying electrical circuit—the actual encipherment of a letter is performed electrically. When a key is pressed, the circuit is completed; current flows through the various components and ultimately lights one of many different lamps, indicating the output letter.

- Navigate website links to find a query
- Observe: $\phi(s), \phi(q), \phi(s'|s,a)$
- Features: average word embeddings
- Baseline policy: $h = \text{NN}(\phi(s), \phi(q)), \quad \pi(a|s) \propto \exp(\langle h, \phi(s') \rangle)$
· Idea: use an approximate graph for planning
· Wikipedia for Schools website (6K pages)
· Approximate graph: 1st+2nd level categories (3%)
WEB-NAV DOMAIN

- VI state space + transitions: approx. graph
- VI Reward map: weighted similarity to \( \phi(q) \)
- Attention: average weighted by similarity to \( \phi(s') \)
- Reactive policy: add feature to \( \phi(s') \)
WEB-NAV DOMAIN

- VI state space + transitions: approx. graph
- VI Reward map: weighted similarity to $\phi(q)$
- Attention: average weighted by similarity to $\phi(s')$
- Reactive policy: add feature to $\phi(s')$

$$
\sum_{\bar{s}} \sigma \left( \langle \phi(s'), \phi(\bar{s}) \rangle_w \right) \bar{V}(\bar{s})
$$
Evaluation:

- Success – all correct actions within top-4 predictions
- Test set 1: start from index page

Results:

<table>
<thead>
<tr>
<th>Network</th>
<th>Success set 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>1025/2000</td>
</tr>
<tr>
<td>VIN</td>
<td>1030/2000</td>
</tr>
</tbody>
</table>
Evaluation:

- Success – all correct actions within top-4 predictions
- Test set 1: start from index page
- Test set 2: start from random page

Results:

<table>
<thead>
<tr>
<th>Network</th>
<th>Success set 1</th>
<th>Success set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>1025/2000</td>
<td>304/4000</td>
</tr>
<tr>
<td>VIN</td>
<td>1030/2000</td>
<td>346/4000</td>
</tr>
</tbody>
</table>
Evaluation:

- Success – all correct actions within top-4 predictions
- Test set 1: start from index page
- Test set 2: start from random page

Results:

<table>
<thead>
<tr>
<th>Network</th>
<th>Success set 1</th>
<th>Success set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>1025/2000</td>
<td>304/4000</td>
</tr>
<tr>
<td>VIN</td>
<td>1030/2000</td>
<td>346/4000</td>
</tr>
</tbody>
</table>

Preliminary results: full English Wikipedia website, using wiki-school as approximate graph
SUMMARY & OUTLOOK
SUMMARY

- 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
· Different planning algorithms
  · MCTS
  · Optimal control$^1$
  · Inverse RL$^2$

· How to obtain approximate planning problem
  · Game manual in Atari

· Generalization in RL$^3$
  · theory?
  · benchmarks?
  · Algorithms?

· Generalization $\neq$ lifelong RL, transfer learning$^4$

· Hierarchical policies, but not options/skills/etc.

---

$^1$Watter et al. NIPS 2015
$^2$Zucker & Bagnell, ICRA 2011
$^4$Taylor & Stone, JMLR 2009
THANK YOU!