Structured Exploration for Reinforcement Learning

Nicholas K. Jong

Department of Computer Sciences
The University of Texas at Austin

December 1, 2010 / PhD Final Defense
Outline

1. Introduction
   - The Reinforcement Learning Problem
   - Reinforcement Learning Methods
   - Thesis Focus

2. Exploration and Approximation

3. Exploration and Hierarchy

4. Conclusion
One Solution to Many Problems

Reality  Many tasks mean many engineering problems
Dream  Opportunity for a single general learning algorithm

Potential Payoffs
- Reduce engineering costs
- Solve problems beyond our current abilities
- Achieve solutions robust to uncertainty
One Formalism for Many Problems

Environment

- Generate reward $r \in \mathbb{R}$ with expected value $R(s, a)$
- Generate next state $s' \in S$ with probability $P(s, a, s')$
- Using unknown reward and transition functions $R$ and $P$

Goal

Find a policy $\pi : S \rightarrow A$ that maximizes future rewards
One Formalism for Many Problems

**Agent**
- Observes state $s \in S$
- Chooses action $a \in A$
- For arbitrary $S$ and $A$

**Environment**
- Generate reward $r \in \mathbb{R}$ with expected value $R(s, a)$
- Generate next state $s' \in S$ with probability $P(s, a, s')$
- Using unknown reward and transition functions $R$ and $P$

**Goal** Find a policy $\pi : S \rightarrow A$ that maximizes future rewards
One Formalism for Many Problems

Agent
- Observes state $s \in S$
- Chooses action $a \in A$
- For arbitrary $S$ and $A$

Environment
- Generate reward $r \in \mathbb{R}$ with expected value $R(s, a)\$
- Generate next state $s' \in S$ with probability $P(s, a, s')$
- Using unknown reward and transition functions $R$ and $P$

Goal
Find a policy $\pi : S \rightarrow A$ that maximizes future rewards
Example: A Resource Gathering Simulation

Simulated Robot’s Task
- **Gather** each of \( n \) resources
- **Navigate** around danger zones

\( n + 2 \) State Variables
- Boolean flag for each resource: \( A, B, \ldots \)
- \( x \) and \( y \) coordinates

\( n + 4 \) Actions
- **north**, **south**, **east**, **west** change \( x \) and \( y \)
- **pickup\( A \)** sets flag \( A \) if near resource \( A \), etc.
- Actions cost \(-1\) generally but up to \(-40\) in “puddles”
The Bellman Equation

- **State value** depends on policy and **action values**
- **Long-term value** equals present value plus future value.

\[
V^\pi(s) = Q^\pi(s, \pi(s)) \\
Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') V^\pi(s')
\]
Evaluating Policies with Value Functions

The Bellman Equation

- State value depends on policy and action values
- Long-term value equals present value plus future value.

\[ V^\pi = \pi Q^\pi \]
\[ Q^\pi = R + \gamma PV^\pi \]
The Bellman Equation

- State value depends on policy and action values
- Long-term value equals present value plus future value.

\[ V^\pi = \pi Q^\pi \]
\[ Q^\pi = R + \gamma PV^\pi \]
The Reinforcement Learning Problem

- Some policy achieves maximal $V^*$
- Planning algorithms compute $V^*$ from $R$, $P$
- But RL algorithms don’t know $R$ and $P$

Example: An Optimal Value Function

V^*(x, y, \{C\})

V^*(x, y, \{C, D\})

V^*(x, y, \{D\})
Example: An Optimal Value Function

- Some policy achieves maximal $V^*$
- Planning algorithms compute $V^*$ from $R, P$
- But RL algorithms don’t know $R$ and $P$
Temporal Difference Learning

- Estimate $V^\pi$ directly from data.
  
  (Sutton and Barto, 1998)

- Given each piece of data $\langle s, a, r, s' \rangle$
  
  $r + \gamma \hat{V}^\pi (s')$ is an estimate of $V^\pi (s)$.
  
  Update $\hat{V}^\pi (s)$ towards this estimate.
  
  Improve $\pi$.

- Converges to the optimal policy in the limit, given appropriate data.

- In practice, converges very slowly!

Most RL research focuses on ways to compute value functions more efficiently from data.
Scaling to Real-World Problems

**Theory**  Eventual convergence to optimal behavior

**Practice**  Too slow for interesting problems

**Branches of RL Research**
- Function Approximation
- Hierarchical RL
- Relational RL
- Inverse RL
- Etc.
Scaling to Real-World Problems

Theory  Eventual convergence to optimal behavior
Practice Too slow for interesting problems

Branches of RL Research
- Function Approximation
- Hierarchical RL
- Relational RL
- Inverse RL
- Etc.
Scaling to Real-World Problems

**Theory**  Eventual convergence to optimal behavior

**Practice**  Too slow for interesting problems

**Brances of RL Research**
- Function Approximation
- Hierarchical RL
- Relational RL
- Inverse RL
- Etc.
Exploration and Exploitation

**Exploitation**
- How to estimate $Q^*$ from data
- Focus of most RL research

**Exploration**
- How to gather better data
- Emphasized by model-based RL
- Focus of this thesis
Exploration and Exploitation

**Exploitation**
- How to estimate $Q^*$ from data
- Focus of most RL research

**Exploration**
- How to gather better data
- Emphasized by model-based RL
- Focus of this thesis
Exploration and Exploitation

**Exploitation**
- How to estimate $Q^*$ from data
- Focus of most RL research

**Exploration**
- How to gather better data
- Emphasized by model-based RL
- Focus of this thesis
Thesis Contributions

Merging Branches of RL
- Previously studied in isolation
- Demonstration of synergies

- Efficient exploration in continuous state spaces
- Efficient exploration given hierarchical knowledge
- Framework for combining algorithmic ideas
- Publicly available implementation of final agent
Thesis Contributions

Merging Branches of RL
- Previously studied in isolation
- Demonstration of synergies

- Efficient exploration in continuous state spaces
- Efficient exploration given hierarchical knowledge
- Framework for combining algorithmic ideas
- Publicly available implementation of final agent
Thesis Contributions

Merging Branches of RL
- Previously studied in isolation
- Demonstration of synergies

- Efficient exploration in continuous state spaces
- Efficient exploration given hierarchical knowledge
- Framework for combining algorithmic ideas
- Publicly available implementation of final agent
Outline

1. Introduction

2. Exploration and Approximation
   - Model-Based Exploration
   - Generalization in Large State Spaces
   - The Fitted R-MAX Algorithm

3. Exploration and Hierarchy

4. Conclusion
Model-Based Reinforcement Learning

- Structured Exploration
  - Approximate Exploration
  - Hierarchical Exploration

- Function Approximation
- Model-Based Exploration
- Hierarchical Decomposition

- Reinforcement Learning
Model-Based Reinforcement Learning

Indirection Permits Simplicity
- $R, P$ predict only one time step
- $R, P$ involve only one action at a time
- Direct training data permits supervised learning

Uncertainty Guides Exploration
- Use model of known states to reach the unknown
- First polynomial-time sample-complexity bounds (Kearns and Singh, 1998; Kakade, 2003)
Model-Based Reinforcement Learning

Indirection Permits Simplicity
- $R, P$ predict only one time step
- $R, P$ involve only one action at a time
- Direct training data permits supervised learning

Uncertainty Guides Exploration
- Use model of known states to reach the unknown
- First polynomial-time sample-complexity bounds (Kearns and Singh, 1998; Kakade, 2003)
Model-Based Reinforcement Learning

Indirection Permits Simplicity
- $R, P$ predict only one time step
- $R, P$ involve only one action at a time
- Direct training data permits supervised learning

Uncertainty Guides Exploration
- Use model of known states to reach the unknown
- First polynomial-time sample-complexity bounds
  (Kearns and Singh, 1998; Kakade, 2003)
Simple and Efficient Learning with R-max
(Moore and Atkeson, 1993; Brafman and Tennenholtz, 2002)

Maximum-Likelihood Estimation
- Straightforward in finite state spaces
- Unreliable with small sample sizes

Small sample sizes
Use optimistic model

Given enough data
Use MLE model

Nicholas K. Jong
Structured Exploration for Reinforcement Learning
Simple and Efficient Learning with R-max
(Moore and Atkeson, 1993; Brafman and Tennenholtz, 2002)

Maximum-Likelihood Estimation
- Straightforward in finite state spaces
- Unreliable with small sample sizes

Small sample sizes
Use optimistic model
Given enough data
Use MLE model
Simple and Efficient Learning with R-max
(Moore and Atkeson, 1993; Brafman and Tennenholtz, 2002)

Maximum-Likelihood Estimation
- Straightforward in finite state spaces
- Unreliable with small sample sizes

Small sample sizes
Use optimistic model

Given enough data
Use MLE model
Simple and Efficient Learning with R-max
(Moore and Atkeson, 1993; Brafman and Tennenholtz, 2002)

Maximum-Likelihood Estimation
- Straightforward in finite state spaces
- Unreliable with small sample sizes

Small sample sizes
Use optimistic model

Given enough data
Use MLE model

Simple and Efficient Learning with R-max
(Moore and Atkeson, 1993; Brafman and Tennenholtz, 2002)
Simple and Efficient Learning with R-max
(Moore and Atkeson, 1993; Brafman and Tennenholtz, 2002)

Maximum-Likelihood Estimation
- Straightforward in finite state spaces
- Unreliable with small sample sizes

Small sample sizes
- Use optimistic model

Given enough data
- Use MLE model
Simple and Efficient Learning with R-max
(Moore and Atkeson, 1993; Brafman and Tennenholtz, 2002)

Maximum-Likelihood Estimation
- Straightforward in finite state spaces
- Unreliable with small sample sizes

Nicholas K. Jong
Structured Exploration for Reinforcement Learning
Challenges for Model-Based Reinforcement Learning

Computational Complexity
- **MDP planning** can be expensive...
- But **CPU cycles** are cheaper than data

Representational Complexity
- State distributions harder to represent than scalar values...
- But simple approximations may suffice

Exhaustive Exploration
- Exploring every unknown state seems unnecessary...
- But intuitive domain knowledge can constrain exploration
Challenges for Model-Based Reinforcement Learning

Computational Complexity
- MDP planning can be expensive...
- But CPU cycles are cheaper than data

Representational Complexity
- State distributions harder to represent than scalar values...
- But simple approximations may suffice

Exhaustive Exploration
- Exploring every unknown state seems unnecessary...
- But intuitive domain knowledge can constrain exploration
Challenges for Model-Based Reinforcement Learning

**Computational Complexity**
- MDP planning can be expensive...
- But CPU cycles are cheaper than data

**Representational Complexity**
- State distributions harder to represent than scalar values...
- But simple approximations may suffice

**Exhaustive Exploration**
- Exploring every unknown state seems unnecessary...
- But intuitive domain knowledge can constrain exploration
Function Approximation

Structured Exploration

Approximate Exploration    Hierarchical Exploration

Function Approximation    Model-Based Exploration    Hierarchical Decomposition

Reinforcement Learning

Nicholas K. Jong
Structured Exploration for Reinforcement Learning
**Problem**  Exact representation of $V$ requires a parameter for each state. Many environments have infinite states!

**Key Idea**  Represent $V^\pi$ using a small number of parameters.

- **Examples**
  - The weights of a neural network
  - Coefficients of some basis functions: $V^\pi = \sum_i w_i^\pi \phi_i$

- **Generalization of values**
  - Changing $V^\pi(s)$ changes one or more parameters.
  - Each parameter influences the value of several states.
Problem

Exact representation of $V$ requires a parameter for each state. Many environments have infinite states!

Key Idea

Represent $V^\pi$ using a small number of parameters.

- Examples
  - The weights of a neural network
  - Coefficients of some basis functions: $V^\pi = \sum_i w_i^\pi \phi_i$

- Generalization of values
  - Changing $V^\pi(s)$ changes one or more parameters.
  - Each parameter influences the value of several states.
Function Approximation

**Problem**  Exact representation of $V$ requires a parameter for each state. Many environments have infinite states!

**Key Idea**  Represent $V^\pi$ using a small number of parameters.

- **Examples**
  - The weights of a neural network
  - Coefficients of some basis functions: $V^\pi = \sum_i w_i^\pi \phi_i$

- **Generalization of values**
  - Changing $V^\pi(s)$ changes one or more parameters.
  - Each parameter influences the value of several states.
Problem Exact representation of $V$ requires a parameter for each state. Many environments have infinite states!

**Key Idea**
Represent $V^\pi$ using a small number of parameters.

- **Examples**
  - The weights of a neural network
  - Coefficients of some basis functions: $V^\pi = \sum_i w_i^\pi \phi_i$

- **Generalization of values**
  - Changing $V^\pi(s)$ changes one or more parameters.
  - Each parameter influences the value of several states.
Fitted Value Iteration
(Gordon, 1995)

**Averagers**
- Parameterize $V^\pi$ with values $V^\pi(X)$ on $X \subseteq S$
- $V^\pi(s)$ is a weighted average $\sum_{x \in X} \phi(s, x) V^\pi(x)$

**Discrete Planning in Continuous State Spaces**
- Approximate planning with an exact MDP
- Exact planning with an approximate MDP
Fitted Value Iteration
(Gordon, 1995)

Averagers
- Parameterize $V^\pi$ with values $V^\pi(X)$ on $X \subset S$
- $V^\pi(s)$ is a weighted average $\sum_{x \in X} \phi(s, x) V^\pi(x)$

Discrete Planning in Continuous State Spaces
- Approximate planning with an exact MDP
- Exact planning with an approximate MDP
Fitted Value Iteration
(Gordon, 1995)

Averagers

- Parameterize $V^\pi$ with values $V^\pi(X)$ on $X \subset S$
- $V^\pi(s)$ is a weighted average $\sum_{x \in X} \phi(s, x) V^\pi(x)$

Discrete Planning in Continuous State Spaces

- Approximate planning with an exact MDP
- Exact planning with an approximate MDP
Fitted Value Iteration
(Gordon, 1995)

Averagers
- Parameterize $V^\pi$ with values $V^\pi(X)$ on $X \subset S$
- $V^\pi(s)$ is a weighted average $\sum_{x \in X} \phi(s, x) V^\pi(x)$

Discrete Planning in Continuous State Spaces
- Approximate planning with an exact MDP
- Exact planning with an approximate MDP
Model Approximation

- Reinforcement Learning
- Model-Based Exploration
- Hierarchical Decomposition
- Function Approximation
- Model-Based Exploration
- Hierarchical Exploration
- Approximate Exploration
- Structured Exploration
Model Approximation
(Jong and Stone, 2007b)

Approximate $sa$ using instances $i = \langle s_i, a_i, r_i, s'_i \rangle$

$\Psi(sa, i)$ Model averager weighting $sa$ against $s_i a_i$

$D_P(s_i, s')$ Empirical effect applying transition at $i$ to $s$
Approximate $sa$ using instances $i = \langle s_i, a_i, r_i, s'_i \rangle$

$\Psi(sa, i)$ Model averager weighting $sa$ against $s_i a_i$

$D_P(s_i, s')$ Empirical effect applying transition at $i$ to $s$
Approximate $sa$ using instances $i = \langle s_i, a_i, r_i, s'_i \rangle$

$\Psi(sa, i)$ Model averager weighting $sa$ against $s_ia_i$

$D_P(si, s')$ Empirical effect applying transition at $i$ to $s$
Model Approximation
(Jong and Stone, 2007b)

Approximate $sa$ using instances $i = \langle s_i, a_i, r_i, s'_i \rangle$

$\Psi(sa, i)$ Model averager weighting $sa$ against $s_i a_i$

$D_P(si, s')$ Empirical effect applying transition at $i$ to $s$
Fitted R-MAX
(Jong and Stone, 2007a)

Model approximation, R-MAX exploration, value approximation

- **Action Selection**
- **Planning**
- **Model Estimation**
  - **Data**
  - **$R,P$**
  - **$Q$**
  - **$\pi$**
Fitted R-MAX (Jong and Stone, 2007a)

Model approximation, R-MAX exploration, value approximation
Fitted R-MAX
(Jong and Stone, 2007a)

Model approximation, R-MAX exploration, value approximation

Nicholas K. Jong
Structured Exploration for Reinforcement Learning
Fitted R-MAX 
(Jong and Stone, 2007a)

Model approximation, R-MAX exploration, value approximation
An Instance of Fitted R-MAX

Model Averager

\[ \psi(sa, si) \propto K(s, s_i) \delta(a, sa) \]

\[ K(s, s') = \exp \left( \frac{d(s, s')^2}{b^2} \right) \]

“radial basis data”

R-MAX Exploration

- sa known if sufficient weight: \( \sum_{i \mid a_i = a} K(s, s_i) \geq m \)

Value Averager

- Interpolation over a uniform grid
An Instance of Fitted R-MAX

**Model Averager**

- $\psi(sa, si) \propto K(s, s_i)\delta(a, s_a)$
- $K(s, s') = \exp \left( \frac{d(s, s')^2}{b^2} \right)$
- “radial basis data”

**R-MAX Exploration**

- $sa$ known if sufficient weight: $\sum_{i \mid a_i=a} K(s, s_i) \geq m$

**Value Averager**

- Interpolation over a uniform grid
An Instance of Fitted R-MAX

Model Averager

- $\psi(sa, si) \propto K(s, s_i)\delta(a, sa)$
- $K(s, s') = \exp\left(\frac{d(s, s')^2}{b^2}\right)$
- “radial basis data”

R-MAX Exploration

- $sa$ known if sufficient weight: $\sum_{i \mid a_i=a} K(s, s_i) \geq m$

Value Averager

- Interpolation over a uniform grid
For $n = 1$ resource, almost equivalent to benchmark domain “Puddleworld”

Can compare against performance data from NIPS RL Benchmarking Workshop (2005)

State-of-the-art algorithms implemented and tuned by other researchers
Fitted Value Functions for PuddleWorld

Policy and value function with 250 instances
Policy and value function with 500 instances
Fitted Value Functions for PuddleWorld

Policy and value function with 750 instances
Fitted Value Functions for PuddleWorld

Policy and value function with 1000 instances
Fitted Value Functions for PuddleWorld

Policy and value function with 1500 instances
Fitted Value Functions for PuddleWorld

Policy and value function with 2000 instances
Fitted Value Functions for PuddleWorld

Policy and value function with 3000 instances
Fitted Value Functions for PuddleWorld

Policy and value function with 4000 instances
Fitted Value Functions for PuddleWorld

Policy and value function with 5000 instances
Generalization and Exploration

Inductive Bias

**Model-Free**  Similar states have **similar values**

**Model-Based**  Similar states have **similar dynamics**

Model Generalization

- Is the effect of $sa$ known or unknown?
- Less generalization leads to more exploration

Value Generalization

- How good is my policy $\pi$?
- Less generalization leads to more computation
Generalization and Exploration

Inductive Bias
- Model-Free: Similar states have similar values
- Model-Based: Similar states have similar dynamics

Model Generalization
- Is the effect of sa known or unknown?
- Less generalization leads to more exploration

Value Generalization
- How good is my policy $\pi$?
- Less generalization leads to more computation
### Generalization and Exploration

#### Inductive Bias

| Model-Free | Similar states have **similar values** |
| Model-Based | Similar states have **similar dynamics** |

#### Model Generalization

- Is the effect of $sa$ known or unknown?
- **Less generalization leads to more exploration**

#### Value Generalization

- How good is my policy $\pi$?
- **Less generalization leads to more computation**
Outline

1. Introduction
2. Exploration and Approximation
3. Exploration and Hierarchy
   - Hierarchical Decomposition
   - The R-MAXQ and Fitted R-MAXQ Algorithms
   - The Utility of Hierarchy
4. Conclusion
The Appeal of Hierarchy

Realistic Problems
- Many states and many actions...
- But also deep structure
- Multiple levels of abstraction
- Local dependencies

Structured Learning and Planning
- Don’t write all programs in assembly!
- Reason above the level of primitive actions.
Structured Exploration for Reinforcement Learning

- Hierarchical Decomposition
- The R-MAXQ and Fitted R-MAXQ Algorithms
- The Utility of Hierarchy

Hierarchy in Reinforcement Learning

Structured Exploration

- Approximate Exploration
- Hierarchical Exploration

- Function Approximation
- Model-Based Exploration
- Hierarchical Decomposition

Reinforcement Learning
Options (Sutton, Precup, and Singh, 1999)

- Partial policies as macros
- An option $o$ comprises:
  - An initiation set $I^o \subset S$
  - An option policy $\pi^o : S \rightarrow A$
  - A termination function $T^o : S \rightarrow [0, 1]$

MAXQ (Dietterich, 2000)

- A hierarchy of RL problems
- A task $o$ comprises:
  - A set of subtasks $A^o$
  - A goal reward function $G^o : T^o \rightarrow \mathbb{R}$
  - A set of terminal states $T^o$
Introduction
Exploration and Approximation
Exploration and Hierarchy
Conclusion

Hierarchy in Reinforcement Learning

**Options** (Sutton, Precup, and Singh, 1999)

- Partial policies as macros
- An option $o$ comprises:
  - An initiation set $I^o \subset S$
  - An option policy $\pi^o : S \rightarrow A$
  - A termination function $T^o : S \rightarrow [0, 1]$

**MAXQ** (Dietterich, 2000)

- A hierarchy of RL problems
- A task $o$ comprises:
  - A set of subtasks $A^o$
  - A goal reward function $G^o : T^o \rightarrow \mathbb{R}$
  - A set of terminal states $T^o$
High-Level \textbf{Rewards} are Low-Level \textbf{Values}

- Separate $Q^o$ into \textbf{components} $Q^o_a$ by action
- Compute $R^o_a = V^o$ recursively
- Learn $C^o_a := \gamma P^o_a V^o$ directly

\begin{align*}
V^{\text{Drive}}(s) &= \text{Drive to Campus} \\
&\quad \text{Reach Highway} \\
&\quad \text{Turn Left}
\end{align*}
MAXQ Value Function Decomposition

### High-Level Rewards are Low-Level Values

- Separate $Q^o$ into components $Q^o_a$ by action
- Compute $R^o_a = V^o$ recursively
- Learn $C^o_a := \gamma P^o_a V^o$ directly

\[ V^{\text{Drive}}(s) = \]

\[ Q^{\text{Drive \ Reach}}(s) \]
## MAXQ Value Function Decomposition

### High-Level Rewards are Low-Level Values

- Separate $Q^o$ into components $Q^o_a$ by action
- Compute $R^o_a = V^o$ recursively
- Learn $C^o_a := \gamma P^o_a V^o$ directly

\[
V^{\text{Drive}}(s) = V^{\text{Drive}}(s) + V^{\text{Reach}}(s) + Q^{\text{Drive Reach}}(s)
\]

---

Nicholas K. Jong  
Structured Exploration for Reinforcement Learning
High-Level Rewards are Low-Level Values

- Separate $Q^o$ into components $Q^o_a$ by action
- Compute $R^o_a = V^o$ recursively
- Learn $C^o_a := \gamma P^o_a V^o$ directly

MAXQ Value Function Decomposition

$V^{\text{Drive}}(s) = V^{\text{Drive}}(s)$
MAXQ Value Function Decomposition

High-Level Rewards are Low-Level Values
- Separate $Q^o$ into components $Q^o_a$ by action
- Compute $R^o_a = V^o$ recursively
- Learn $C^o_a := \gamma P^o_a V^o$ directly

$V^{Drive}(s) = Q^{Drive}(s)$

$Q^{Drive}(s)$

Nicholas K. Jong
Structured Exploration for Reinforcement Learning
High-Level **Rewards** are Low-Level **Values**

- Separate $Q^o$ into components $Q^o_a$ by action
- Compute $R^o_a = V^o$ recursively
- Learn $C^o_a := \gamma P^o_a V^o$ directly

*V*^{Drive}(s) = *V*^{Reach}(s) + *C*^{Drive}_{Reach}(s)
**MAXQ Value Function Decomposition**

**High-Level Rewards are Low-Level Values**
- Separate $Q^o$ into components $Q^o_a$ by action
- Compute $R^o_a = V^o$ recursively
- Learn $C^o_a := \gamma P^o_a V^o$ directly

```
V^{Drive}(s) = Q^{Reach}(s) + C^{Drive}(s)
```

---

Nicholas K. Jong
Structured Exploration for Reinforcement Learning
MAXQ Value Function Decomposition

High-Level Rewards are Low-Level Values

- Separate $Q^o$ into components $Q^o_{a}$ by action
- Compute $R^o_{a} = V^o$ recursively
- Learn $C^o_{a} := \gamma P^o_{a} V^o$ directly

\[ V^{\text{Drive}}(s) = V^{\text{Turn}}(s) + C^{\text{Reach}}(s) + C^{\text{Drive}}(s) \]
Hierarchical Model Decomposition

Structured Exploration

Approximate Exploration  Hierarchical Exploration

Function Approximation  Model-Based Exploration  Hierarchical Decomposition

Reinforcement Learning

Nicholas K. Jong  Structured Exploration for Reinforcement Learning
Hierarchical Model Decomposition
(Jong and Stone, 2008)

High-Level **Successors** are Low-Level **Terminals**

- \( P_{\text{Drive}} \cdot \Omega_{\text{Reach}} \)
- \( \Omega^0(s, s') \): Discounted probability that executing \( o \) in \( s \) terminates at \( s' \)
- \( \Omega^0(\cdot, s') \) is a value function!

\[
P_{\text{Drive}} \cdot \Omega_{\text{Reach}} = \Omega^0(s, s') \text{ s.t. } a \in \pi(s) \Rightarrow \Omega^0(s, \cdot) = \text{value function!}
\]
Hierarchical Model Decomposition
(Jong and Stone, 2008)

**High-Level Successors are Low-Level Terminals**

1. $P_{\text{Drive}}^{\text{Reach}} = \Omega_{\text{Reach}}$
2. $\Omega^o(s, s')$: Discounted probability that executing $o$ in $s$ terminates at $s'$
3. $\Omega^o(\cdot, s')$ is a value function!

---

Nicholas K. Jong
Structured Exploration for Reinforcement Learning
Hierarchical Model Decomposition
(Jong and Stone, 2008)

High-Level Successors are Low-Level Terminals

- \( P^{\text{Drive}}_{\text{Reach}} = \Omega^{\text{Reach}} \)
- \( \Omega^o(s, s') \): Discounted probability that executing \( o \) in \( s \) terminates at \( s' \)
- \( \Omega^o(\cdot, s') \) is a value function!
The R-MAXQ Algorithm
(Jong and Stone, 2008)

**Primitive Tasks**
- Learn primitive models from data
- Splice in R-MAX optimistic exploration
- Result: $V^a$ and $\Omega^a$

**Composite Tasks**
- Concatenate subtask $V^a$ and $\Omega^a$ into $R^o$ and $P^o$
- Plan $\pi^o$ using MAXQ goal rewards
- Evaluate $\pi^o$ without goal rewards
- Result: $V^o$ and $\Omega^o$
### The R-MAXQ Algorithm

*(Jong and Stone, 2008)*

#### Primitive Tasks
- Learn primitive models \(V^a, \Omega^a\) from data
- Splice in R-MAX optimistic exploration
- Result: \(V^a\) and \(\Omega^a\)

#### Composite Tasks
- Concatenate subtask \(V^a, \Omega^a\) into \(R^o, P^o\)
- Plan \(\pi^0\) using MAXQ goal rewards
- Evaluate \(\pi^0\) without goal rewards
- Result: \(V^o\) and \(\Omega^o\)
The R-MAXQ Algorithm
(Jong and Stone, 2008)

**Primitive Tasks**
- Learn primitive models from data
- Splice in R-MAX optimistic exploration
- Result: $V^a$ and $\Omega^a$

**Composite Tasks**
- Concatenate subtask $V^a$ and $\Omega^a$ into $R^o$ and $P^o$
- Plan $\pi^o$ using MAXQ goal rewards
- Evaluate $\pi^o$ without goal rewards
- Result: $V^o$ and $\Omega^o$
The Fitted R-maxq Algorithm

Structured Exploration

Approximate Exploration

Hierarchical Exploration

Function Approximation

Model-Based Exploration

Hierarchical Decomposition

Reinforcement Learning
The Fitted R-maxq Algorithm
(Jong and Stone, 2009)

Prediction
Solve $V^o = \pi^o(R^o + \gamma P^o(I - T^o)V^o)$
Solve $\Omega^o = \pi^o(P^o T^o + \gamma P^o(I - T^o)\Omega^o)$

Planning
Optimize $\tilde{V}^o = T^o G^o + (I - T^o)\pi^o(R^o + \gamma P^o \tilde{V}^o)$

Value Approximation
Define $R^o[sa] = V^a[s]$ 
Define $P^o[sa, x] = \Omega^a[s, s']\Phi^o[s', x]$
The Software Architecture

**Algorithm**
1. Execute $\pi_{\text{Root}}$ hierarchically
2. Update data: $R_D$ and $P_D$
3. Propagate changes to $\pi_{\text{Root}}$
4. Repeat

**Averagers**
- $\Phi$: Interpolation over uniform grid
- $\Psi$: Radial basis functions

**Optimizations**
- Memoization and DP
- Prioritized sweeping
- Sparse representations
- Cover trees for online nearest neighbors

**Hierarchical Decomposition**
- The R-MAXQ and Fitted R-MAXQ Algorithms
- The Utility of Hierarchy
Example: A Resource Gathering Simulation

Simulated Robot’s Task
- **Gather** each of $n$ resources
- **Navigate** around danger zones

$n + 2$ State Variables
- Boolean flag for each resource: $A, B, \ldots$
- $x$ and $y$ coordinates

$n + 4$ Actions
- **north, south, east, west** change $x$ and $y$
- **pickupA** sets flag $A$ if near resource $A$, etc.
- Actions cost $-1$ generally but up to $-40$ in “puddles”
The Utility of Hierarchy and Model Generalization

Model generalization allows Fitted R-MAX to outperform R-MAX.

Hierarchical decomposition allows R-MAXQ to outperform R-MAX.
The Utility of Hierarchy and Model Generalization

Model generalization allows Fitted R-MAX to outperform R-MAX.

Hierarchical decomposition allows R-MAXQ to outperform R-MAX.
Model generalization allows Fitted R-MAX to outperform R-MAX.
Hierarchical decomposition allows R-MAXQ to outperform R-MAX.

These two ideas synergize in Fitted R-MAXQ!
Task Hierarchies as Domain Knowledge

Flat hierarchy only knows model averagers.
Shallow hierarchy also knows that gathering each resource is independent.
Deep hierarchy also knows the set of resource locations (but must still associate resource with location).

Nicholas K. Jong
Structured Exploration for Reinforcement Learning
From \( s \), explore unknown state in puddle or exploit known solution?

**Flat Hierarchy**

Optimism about the unknown effects of \( \text{pickupD} \) at \( x \) outweighs value of known solution, \( V^\pi(s) > V^\pi(s) \).

**Shallow Hierarchy**

Value of \( \text{pickupD} \) at \( x \) less than value of known solution in the context of \( \text{GatherD} \), \( V^\pi_{\text{GatherD}}(s) < V^\pi_{\text{GatherD}}(s) \).
Soft Inductive Bias in Hierarchy

From $s$, explore unknown state in puddle or exploit known solution?

**Flat Hierarchy**

Optimism about the unknown effects of $\text{pickupD}$ at $x$ outweighs value of known solution, $V^\pi(s) > V^\pi(s)$.

**Shallow Hierarchy**

Value of $\text{pickupD}$ at $x$ less than value of known solution in the context of $\text{GatherD}$, $V^\pi_{\text{GatherD}}(s) < V^\pi_{\text{GatherD}}(s)$. 
From $s$, explore unknown state in puddle or exploit known solution?

**Flat Hierarchy**

Optimism about the unknown effects of pickupD at $x$ outweighs value of known solution, $V^\pi(s) > V^\pi(s)$.

**Shallow Hierarchy**

Value of pickupD at $x$ less than value of known solution in the context of GatherD, $V^\pi_{\text{GatherD}}(s) < V^\pi_{\text{GatherD}}(s)$.
Hierarchical Constraint and Reformulation

- Hierarchies can find embedded structure.
- Hierarchies can constrain policies and therefore exploration.

Deep Hierarchy

**pickup** actions only possible before or after **Navigate** tasks.
Hierarchical Constraint and Reformulation

- Hierarchies can find **embedded** structure.
- Hierarchies can **constrain** policies and therefore **exploration**.

**Deep Hierarchy**

**pickup actions** only possible **before or after** **Navigate** tasks.
Hierarchical Constraint and Reformulation

- Hierarchies can find embedded structure.
- Hierarchies can constrain policies and therefore exploration.

Deep Hierarchy

pickup actions only possible before or after Navigate tasks.
Discovering Abstractions

- We can now use task hierarchies to efficiently explore continuous environments.
- Can we discover composite tasks automatically?

What makes a good subtask?
- Other research: “bottleneck states”
- My conjecture: “sets of relevant features”
  (Jong and Stone, 2005)
No Free Lunch
No algorithm can learn or discover efficiently in all possible worlds!

Bayesian Reinforcement Learning
- Begin with a prior distribution over environments
- Plan over “belief states”
- Update belief distribution given data

Key question
What is the right prior distribution?

Conjecture
Distributions over task hierarchies

Goal
Efficient approximation of optimal Bayesian solution
Natural Knowledge Representations

- Model-free methods learn a monolithic value function.
- Models are a **natural form** of domain knowledge.
- Models are **modular**: piecewise independent.

**Don’t Reinvent the Wheel**

- Exploit known reward function
- Exploit known dynamics of some actions
- Exploit known dynamics of some state variables
Connections to Other Fields of Artificial Intelligence

- Higher-level actions are more deterministic and discrete.
- Abstract actions could help define abstract state variables.
- Example: A Boolean feature predicting that a task will reach a “good” terminal state.
- Possibly define tasks with postconditions that achieve other tasks’ preconditions

Recognize Familiar Problems Emerging from Data

- Classical planning, scheduling, constraint satisfaction
- Object recognition and multi-agent learning
Summary

- Agents that apply principled exploration to structured environments
- Exploration in continuous domains that require generalization
- Exploration in domains with hierarchical structure
- Publicly available implementation

The bigger picture
  - Extend the reach of RL closer to the real world
  - Build a foundation for work in structure discovery