Overall Goal

Apply RL to real-world problems

- Problems where actions are expensive
- Limited time for exploration
Reinforcement Learning

- Set of states $S$, actions $A$
- Transition function $P(s'|s, a)$
- Reward function $R(s, a)$
- Factored state $s = \langle x_1, x_2, ..., x_n \rangle$
- Find policy $\pi$ mapping states to actions to maximize reward
  - $Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a')$
  - $\pi(s) = \arg\max_a Q^*(s, a)$
Example: Finding your favorite restaurant

- Want to find our favorite restaurant
- Could try a new restaurant every night
- At some point, we have to stop exploring
- Can we eliminate some possibilities?
Example: Finding your favorite restaurant
Example: Finding your favorite restaurant
Example: Finding your favorite restaurant
Example: Finding your favorite restaurant
Model-Free Methods

Q-Learning

- Table of Q-values: $Q(s, a)$
- Loop (state $s$):
  - Take action $a = \operatorname{argmax}_a Q(s, a)$
  - Receive reward $r$, next state $s'$
  - Update $Q(s, a)$ using Bellman equations
  - $s \leftarrow s'$

$\epsilon$-greedy exploration

- Take greedy action most of the time
- Take random action $\epsilon$ of the time
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning

Todd Hester and Peter Stone – UT Austin

An Empirical Comparison of Abstraction in Models of MDPs
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning

Todd Hester and Peter Stone – UT Austin

An Empirical Comparison of Abstraction in Models of MDPs
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Finding your favorite restaurant: Q-Learning
Model-Based Methods

- Learn model $P(s'|s, a)$ and $R(s, a)$ through experiences
- Perform value iteration in model to learn Q-values

**R-Max**

- Track number of visits to each state-action pair
- State-actions with fewer than $m$ visits are unknown
- Agent is driven to explore unknown state-actions through bonus reward

### Value Function
Finding your favorite restaurant: R-Max

Todd Hester and Peter Stone – UT Austin
An Empirical Comparison of Abstraction in Models of MDPs
Finding your favorite restaurant: R-Max
Finding your favorite restaurant: R-Max
Finding your favorite restaurant: R-Max

Todd Hester and Peter Stone – UT Austin
An Empirical Comparison of Abstraction in Models of MDPs
Finding your favorite restaurant: R-Max

<table>
<thead>
<tr>
<th>P</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Finding your favorite restaurant: R-Max
Finding your favorite restaurant: R-Max
Finding your favorite restaurant: R-Max
Finding your favorite restaurant: R-Max

<table>
<thead>
<tr>
<th>P</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>63</td>
</tr>
<tr>
<td>51</td>
<td>42</td>
</tr>
<tr>
<td>20</td>
<td>38</td>
</tr>
<tr>
<td>56</td>
<td>31</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>
Finding your favorite restaurant: R-Max

Todd Hester and Peter Stone – UT Austin
An Empirical Comparison of Abstraction in Models of MDPs
Finding your favorite restaurant: R-Max
Finding your favorite restaurant: R-Max
Finding your favorite restaurant: R-Max
Have to learn separate model for every state-action
Need more efficient exploration

**Q-Learning**
- Only updates values when taking an action
- Random exploration

**R-Max**
- Must visit each state-action $m$ times
- Infeasible in larger domains
Our approach: Function approximation in the model

Model-Based methods
- Faster value backups
- Targeted exploration

Want to learn a model of a large domain
- Incorporate function approximation into the model learning
- Generalize the transition and reward effects in the model
- Not the same as generalizing Q-values in a model-free method
Finding your favorite restaurant: Desired Behavior

Todd Hester and Peter Stone – UT Austin
An Empirical Comparison of Abstraction in Models of MDPs
Finding your favorite restaurant: Desired Behavior
Finding your favorite restaurant: Desired Behavior
Finding your favorite restaurant: Desired Behavior
Finding your favorite restaurant: Desired Behavior
Finding your favorite restaurant: Desired Behavior

Todd Hester and Peter Stone – UT Austin

An Empirical Comparison of Abstraction in Models of MDPs
Finding your favorite restaurant: Desired Behavior

Todd Hester and Peter Stone – UT Austin

An Empirical Comparison of Abstraction in Models of MDPs
Finding your favorite restaurant: Desired Behavior

Todd Hester and Peter Stone – UT Austin
An Empirical Comparison of Abstraction in Models of MDPs
What we really want from our model

- Generalize transitions and rewards well across states
- Knows what it knows - KWIK (Li et al. ICML 2008)
- Agent explores state-actions that model does not know

- Can we use existing supervised learning techniques?
Idea: Make it a supervised learning problem

- Model learning is a supervised learning problem
- **Input:** State and Action
- **Output:** Change in state and reward
- Separate model for each state feature and reward
- Also produce a *confidence* measure to drive *exploration*
Methods

- **Tabular** (number of visits)
- **C4.5 Decision Tree** (number of instances in leaf)
- **Committee of Trees** (disagreement between tree’s predictions)
- **Random Forest** (disagreement between tree’s predictions)
- **Support Vector Machine** (distance from decision boundary)
- **Neural Network** (error in output sums)
- **K Nearest Neighbor** (average distance to K neighbors)
Related Work

SLF-RMax
- Strehl et al. (AAAI 2007)
- Make predictions based on subsets of features
- Computationally expensive

Tree based method of Degris et al.
- Degris et al. (ICML 2006)
- Model MDP with decision trees
- Model absolute transitions
**Example: Decision tree model for \( \Delta x \)**

![Decision tree model with examples](image)

**Todd Hester and Peter Stone – UT Austin**

*An Empirical Comparison of Abstraction in Models of MDPs*
Experiments

- Experiments in three domains to compare:
  - How well models **generalize** transition and reward effects across states
  - How well models **know what they know**

- **Tree-based** methods perform very well on a small number of training samples

- **Tabular** and **SVM** models perform better after large number of samples
Supervised learning techniques can be used for model learning in an RL setting
These techniques can be used to drive exploration
Real Test: Use them in an RL algorithm
Supervised learning techniques can be used to build models and drive exploration in large domains.
Experiments

- Train the models on randomly sampled transitions
- Test the model on every state-action in the domain
- Threshold the confidence measure to classify transitions as known or unknown
- Vary the threshold over the full range to plot operating characteristics
- Do this with different domains and amount of training samples
Results: Taxi

- Dietterich (ICML 1998)
- State Features: x, y, passenger, destination
- Six Actions: East, West, North, South, PickUp, PutDown
- Stochastic: Move in intended direction 80% of time
Results: Taxi

- After 50 random actions
Results: Taxi

Taxi with 6400 samples

% Classified Unknown

# Correct / (Correct + Incorrect)

Tree Committee Random Forest SVM KNN Neural Network Tabular

After 6400 random actions
Results: Castle

- Gridworld with 8 5x5 rooms
- Three state features: room, x and y relative to room
Results: Castle

- After 50 random actions
Results: Castle

- After 1600 random actions
Results: Lights Domain

- Light switch with different modes: random, locked, and unlocked
- Three state features: mode, level, and an extra random variable
- Three actions to switch to each mode
- Two actions to move level up and down (only work in unlocked mode)
Results: Lights

- After 50 random actions
Results: Lights

- After 12800 random actions