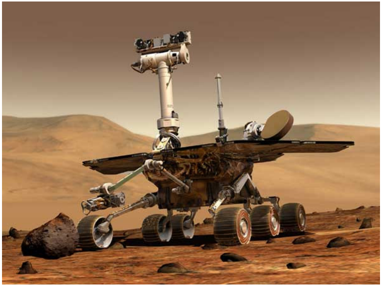
Temporal Difference Reinforcement Learning in Time-Constrained Domains

Todd Hester

Learning Agents Research Group
Department of Computer Science
The University of Texas at Austin

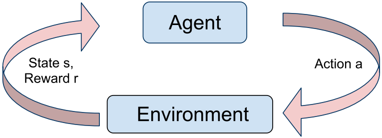
Thesis Proposal
September 29, 2010

Robot Learning



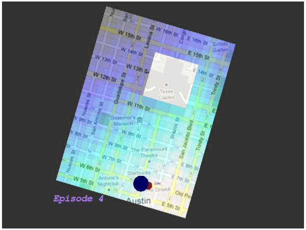
- Robots have the potential to solve many problems
- **But** they are held back by the need to hand-program them
- We need methods for them to learn and adapt to new situations

Reinforcement Learning



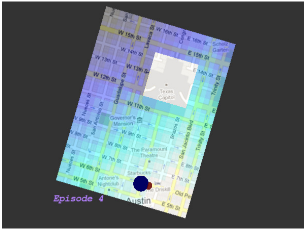
- Value function RL has string of positive theoretical results [Watkins 1989, Brafman and Tennenholtz 2001]
- Could be used for learning and adaptation on robots
- Typically take too many actions to be practical

Reinforcement Learning



Q-Learning

- Theoretically proven to converge
- Only updates VF when taking actions in the world



R-Max

- Learns a tabular model
- State-actions with fewer than m visits are given R_{max} transitions

Sample Complexity of Exploration

Definition: Number of sub-optimal actions the agent must take

- Lower bound is polynomial in N (# of states) and A (# of actions) [Kakade 2003]
- On a very large problem, NA actions is too many
- If actions are expensive, even a few thousand actions may be unacceptable
- What should we do when we do not have enough actions to guarantee convergence to an optimal policy?

Thesis Question

Thesis Question

How should an online reinforcement learning agent act in time-constrained domains?

Thesis Question

Thesis Question

How should an online reinforcement learning agent act in time-constrained domains?

- Takes actions at specified frequency (not batch mode or policy search)
- Concerned with cumulative reward (not final policy)

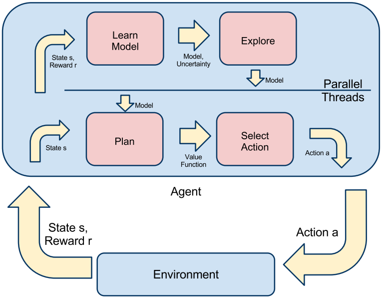
Thesis Question

Thesis Question

How should an online reinforcement learning agent act in time-constrained domains?

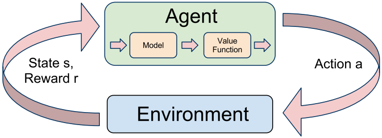
- Agent has a limited number of time steps
- Not enough time steps to learn optimal policy without some assumptions

Proposed Solution



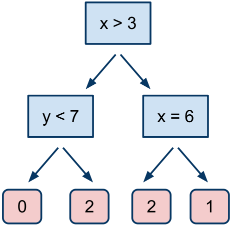
- Develop a model-based algorithm
- Incorporate generalization into the model learning
- Target exploration on specific states to improve model
- Novel architecture for real-time action

Model-Based RL



- Learn transition and reward dynamics, then update VF using model
- Typically more sample-efficient than model-free approaches
- Can update action-values without taking real actions in the world
- Algorithm is constrained by the number of actions it takes to learn an accurate model

Model Generalization

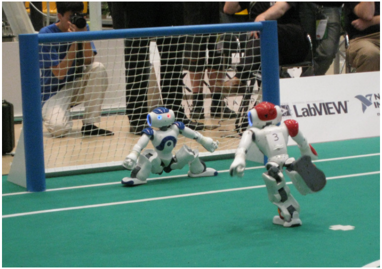


- Do **not** want a tabular model (must visit every state)
- Assume that transition and reward dynamics are similar across states
- Generalize these dynamics across states when learning model
- Can make predictions about states the agent has not visited

Targeted Exploration

- The agent is not going to visit every state
- Which states to visit and which **not** to visit
- Target exploration on states that we are **uncertain** about
- And states that will be **relevant** to the final policy

Real-Time Action

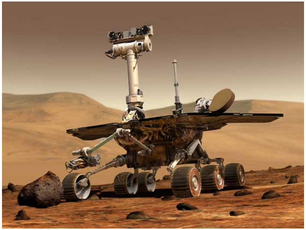


- In many problems, actions must be taken frequently
- Cannot stop and wait for model learning or planning to occur
- Must act in **real-time** at desired frequency

Sample Complexity of Exploration

- Proven lower bound: $O(\frac{NA}{\epsilon(1-\gamma)} log(\frac{1}{\delta}))$
- For deterministic domains: $O(\frac{NA}{(1-\gamma)})$ [Kakade 2003]
- Efficient RL algorithms require a number of actions polynomial in N, A, $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and $\frac{1}{1-\gamma}$.
- Even these algorithms must take at least this many actions to learn an optimal policy
- Look at cases where the agent does not have enough time steps for these algorithms to learn

Limited Time Steps



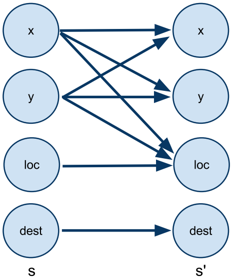
- For many practical problems, we do not have time to take thousands of actions
- Actions may be very time-consuming or expensive
- Need to learn on-line (rewards during learning are important)
- Cannot let the agent break/die during learning

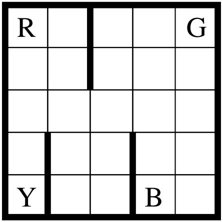
Time-Constrained Problems

- The agent has a lifetime L bounding the number of actions it can take
- Time-Constrained if $L < 2NA$
- Two orders of magnitude less than lower bound
- The agent does **not** have enough time steps to learn the optimal policy without some additional assumptions about the domain
- Evaluate agent on **cumulative** reward over L time steps

Domain	No. States	No. Actions	No. State-Actions	Min Bound Deterministic	Min Bound Stochastic	Maximum L
Taxi	500	6	3,000	300,000	1,050,000	6,000
Four Rooms	100	4	400	40,000	140,000	800
Fuel World	39,711	8	317,688	31,768,800	111,190,800	635,376
Mountain Car	10,000	3	30,000	300,000	10,500,000	60,000
Puddle World	400	4	1,600	160,000	560,000	3,200
Cart Pole	160,000	2	320,000	32,000,000	11,200,000	640,000

Factored Domains





- State is represented by n features: $s = < x_0, x_1, ..., x_n >$
- Transition represented by Dynamic Bayes Network (DBN)
- **Problem**: Learn the structure of the DBN
- Also need to learn the conditional probabilities

Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Expected Contributions

- **Generalized Models** (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

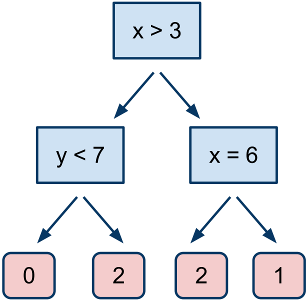
Why generalize the model?

- **Improve Sample efficiency**
- Want to learn a model of a large domain
- Do **not** want to explore every state-action
- 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

Make it a supervised learning problem

- Model learning is a supervised learning problem [AAMAS 2009]
- Input: State and Action
- Output: Next state and reward
- Separate model for each state feature and reward
- Compared Tabular, Decision Trees, Random Forests, SVMs, Neural Networks, and KNN [ICML ARL 2009]
- **Decision Tree** based models were the best

Why decision trees?

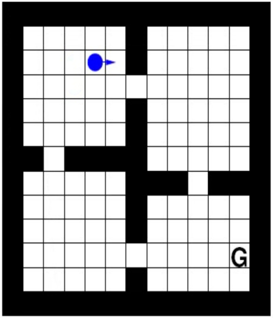


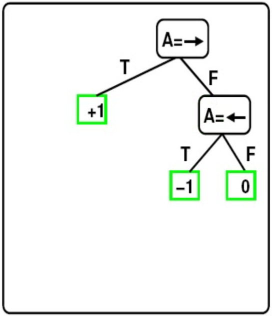
- Incremental and fast
- Generalize broadly at first, refine over time
- Can learn the structure of DBN

Relative Effects

- Predict the change in state: $s^r = s' - s$ rather than absolute next state s'
- Often actions have the same effect across states
- Previous work predicts relative effects [Jong and Stone 2007] [Leffler et al. 2007]

How the Decision Tree Model works





- Build one tree to predict each state feature and reward independently
- Combine their predictions: $P(s^r|s, a) = \Pi_{i=0}^{n} P(x_i^r|s, a)$
- Update trees on-line during learning

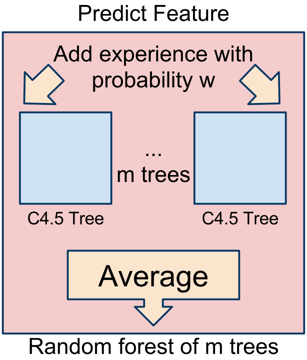
Expected Contributions

- Generalized Models (C)
- **Model Uncertainty** (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Model Uncertainty

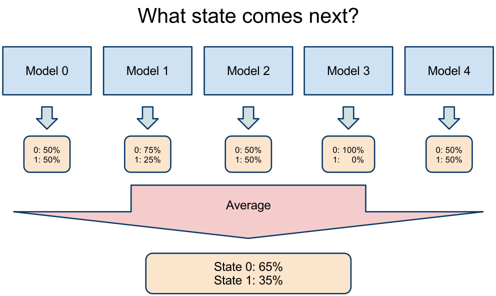
- **Improve Sample efficiency**
- Want some way to measure **uncertainty** of model
- Can use uncertainty to drive exploration and improve model
- Idea: Learn **multiple** possible models and compare them [ICDL 2010]

Random Forest Model



- Build *m* decision trees per forest
- Each tree gets each training experience with probability *w*
- When splitting, each feature is removed from potential split set with probability *f*

Random Forest Benefits

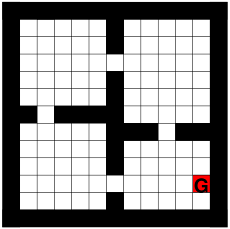


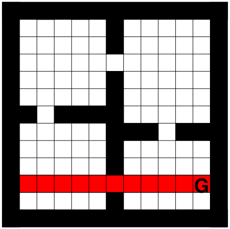
- Each tree represents a possible model of the domain
- Averaging the models inherently incorporates possibilities
- Can use the variance of model's predictions as an uncertainty measure

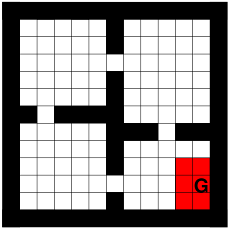
Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- **Targeted Exploration** (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Targeted Exploration







- **Improve Sample efficiency**
- Want to **target** exploration on uncertain states that will be relevant to final policy
- Hypothesize that acting greedily with average model will work well

Use prediction variance

- Can add exploration bonus reward based on variance of model's predictions [ICDL 2010]
- $R(s,a) = R_o(s,a) + b\sigma^2(s,a)$
- $\sigma^2(s,a) = \frac{1}{n+1}[\sigma^2 R(s,a) + \sum_{i=1}^{n} \sigma^2 P(x_i^e|s,a)]$
- Use average variance from each random forest model (n features + reward)

Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- **RL Method for Time-Constrained Domains** (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

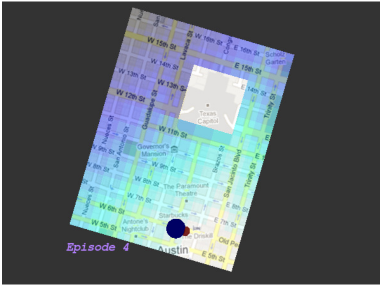
TEXPLORE algorithm [ICDL 2010]

- Combine this model learning method and exploration approach with a planner
- Use UCT as the planning algorithm
- **Seed** the model with a few experiences
- Seed experiences are a natural way to inject human knowledge into the agent

UCT algorithm [Kocsis and Szepesvári 2006]

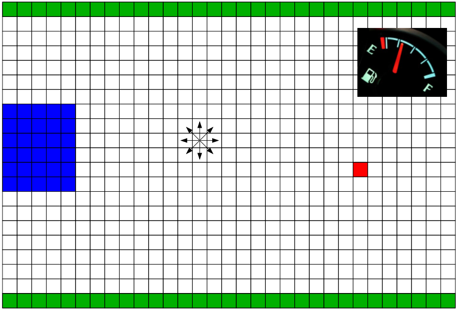
- Update value of a given state by sampling forward many times and updating towards the average return
- Choose actions at each state based on Upper Confidence Bounds
- $a = \mathrm{argmax}_a Q^d(s,a) + \sqrt{2\log(C(s,d))/C(s,a,d)}$
- Concentrates updates on parts of the state space agent is likely to visit soon
- Anytime algorithm

Results



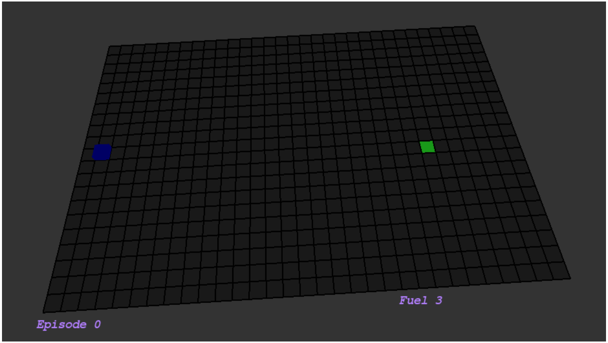
- Learns much faster than R-Max or Q-Learning

Fuel World



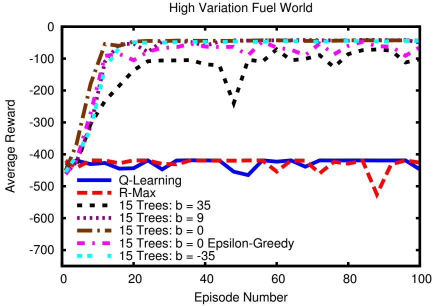
- Most of state space is very predictable
- But fuel stations have varying costs
- Want to explore mainly fuel stations, and particularly ones on short path to goal

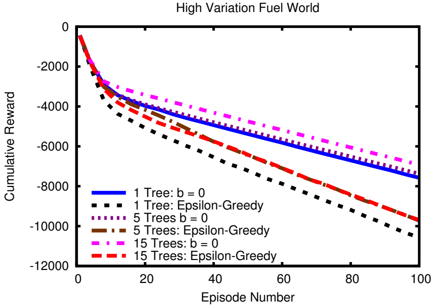
Fuel World Behavior



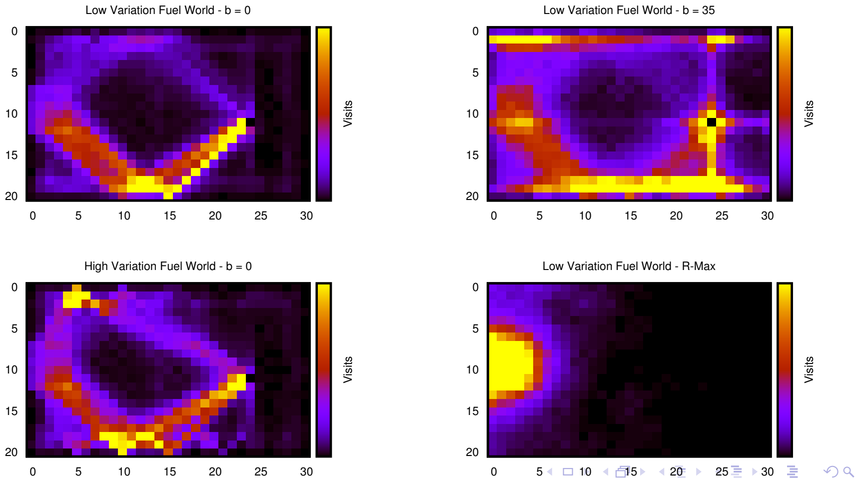
- Agent **focuses its exploration** on fuel stations near the shortest path to the goal.
- Agent **finds near-optimal policies**.

Fuel World Rewards

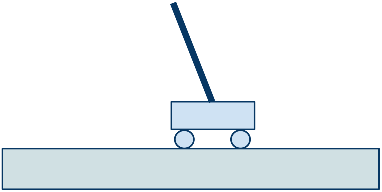




Where did the agent explore?

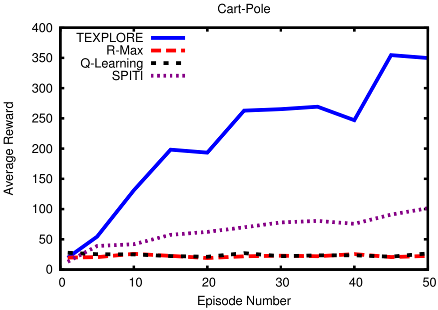


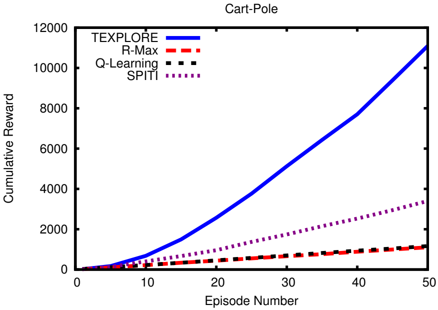
Results: Cart-Pole



- State Features: Pole Angle, Pole Vel, Cart Pos, Cart Vel
- Two Actions: -Force, +Force
- Reward +1 until pole falls or cart moves too far
- Discretized each dimension into 20 values

Cart-Pole Rewards

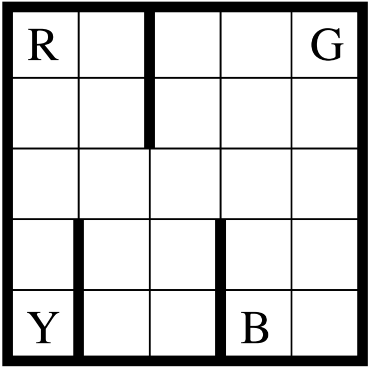




Additional Results

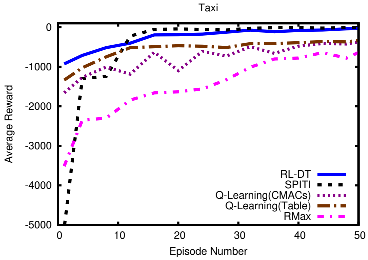
- From Previous Algorithm: RL-DT [AAMAS 2009]
- Used single decision tree model rather than random forest
- No measure of model uncertainty, so no targeted exploration
- Exploration heuristic: Until agent sees reward near R_{max}, it explores unvisited states

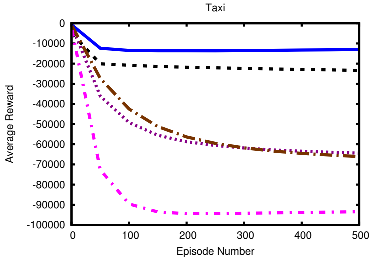
Results: Taxi [Dietterich 1998]



- State Features: x, y, passenger, destination
- Six Actions: East, West, North, South, PickUp, PutDown
- Stochastic: Move in intended direction 80% of time

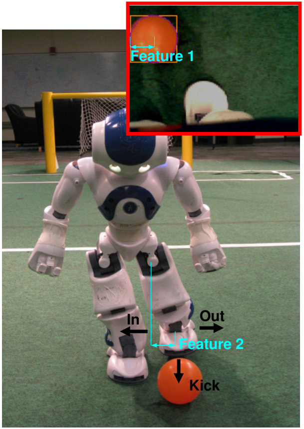
Results: Taxi



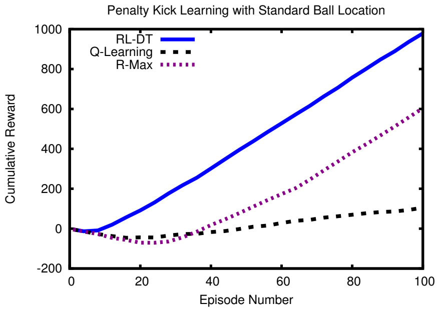


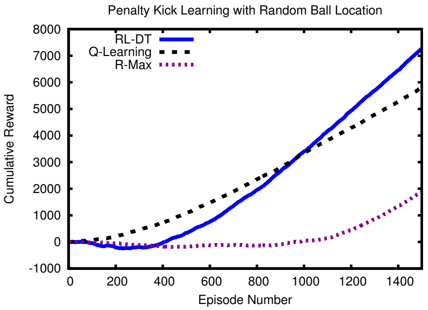
- Performs better on first episode
- Converged in fewer steps (more episodes) than SPITI
- Greater cumulative rewards

Robot Experiments [ICRA 2010]

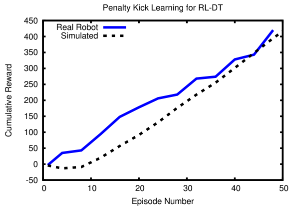


Simulated Results





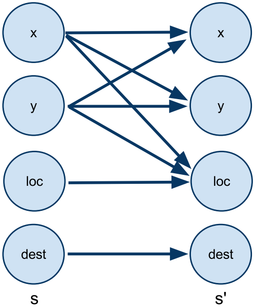
Physical Robot Results

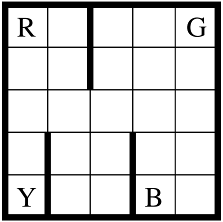


Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- **Model Learning with Dependent Feature Transitions** (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

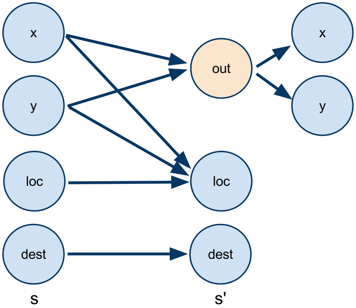
Dependent Feature Transitions

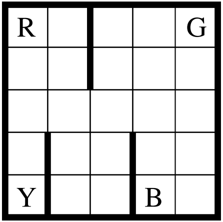




- Make model more accurate and robust
- Algorithm applies to more domains
- Features sometimes transition together

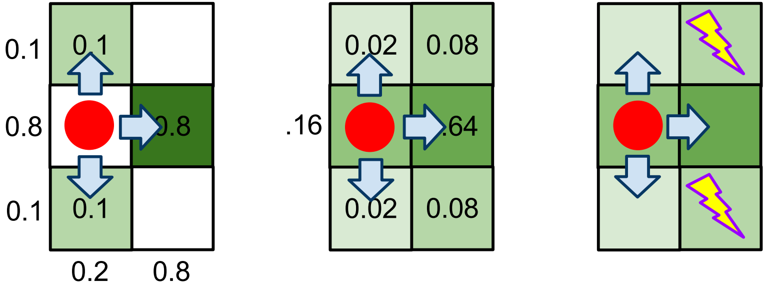
Dependent Feature Transitions



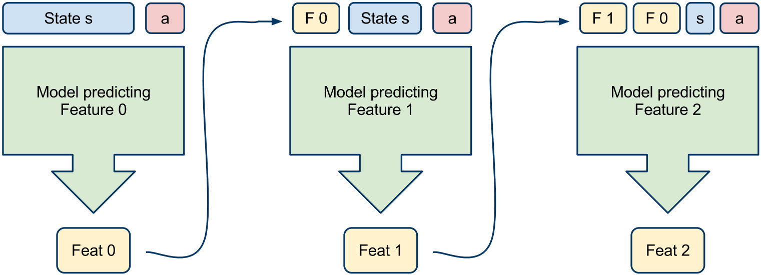


- Make model more accurate and robust
- Algorithm applies to more domains
- Features sometimes transition together

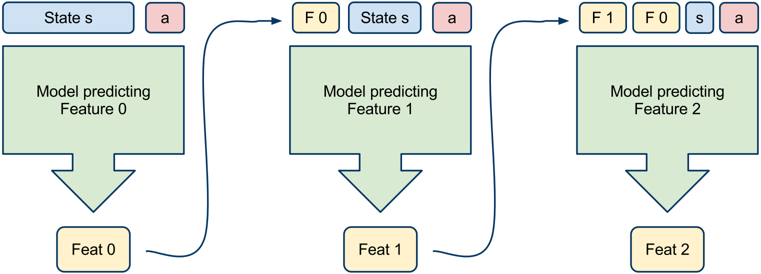
Dependent Feature Transitions



Proposed Dependent Feature Modeling



Proposed Dependent Feature Modeling

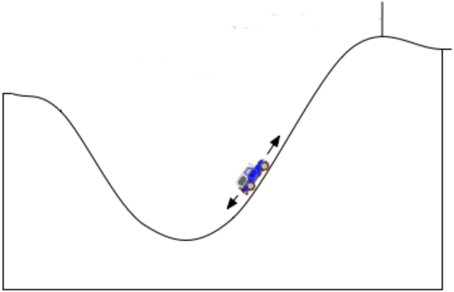


- What if predicting one feature is harder than predicting the other?
- What if its easier to predict x_1 from x_0 rather than x_0 from x_1?

Expected Contributions

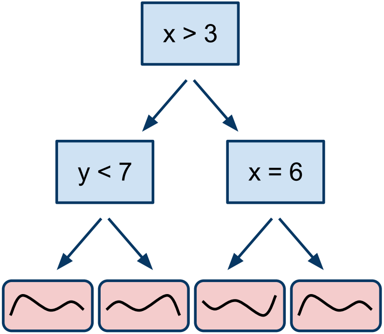
- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- **Extensions to Continuous Domains** (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Continuous Problems



- Most real-world problems are continuous
- First step: Quantize state space

Continuous Models



- Regression trees: More computation?
- Gaussian Process Regression

Continuous Planning

- Fitted Value Iteration [Gordon 1995]
  - Update values for sampled set of states
  - Use function approximator to fit value function
  - Probably computationally slow like VI
- Fitted UCT
  - Can we fit a value function here?
  - Also must maintain visit counts

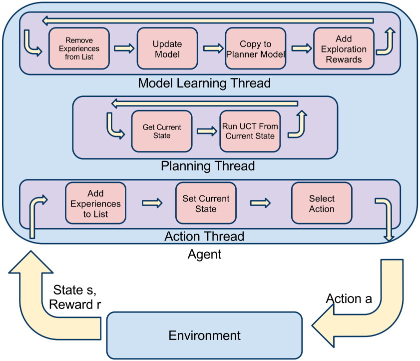
Expected Contributions

- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- **Real-Time Architecture** (P)
- Empirical Evaluation (P)
- Curious Agents (P)

Real-Time Need

- Sometimes planning takes too long
- Sometimes model learning takes too long

Proposed Real-Time Architecture



- Model learning and planning on background threads
- Threads interact through mutex locked data structures
- Can operate at specified action frequency

Expected Contributions

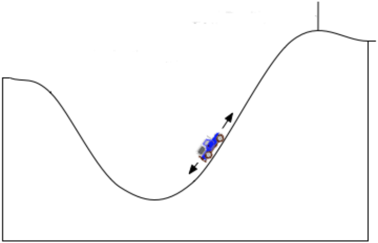
- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- **Empirical Evaluation** (P)
- Curious Agents (P)

Empirical Evaluation

Evaluation Criteria

- Compare **cumulative** rewards because we are interested in **online** learning
- Look at sum of rewards over the L time steps given to the agent
- Evaluate on tasks that require real-time actions (robots)

Empirical Evaluation



- Typical RL Benchmarks (Mountain Car, Cart Pole, Acrobot)
- Robot Tasks: Nao robot, Autonomous vehicle
- Compare with PAC MDP efficient algorithms (MET-RMAX)
- Try to compare with Bayesian RL on **small** problem

Expected Contributions

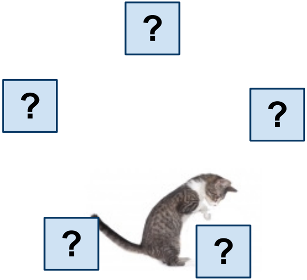
- Generalized Models (C)
- Model Uncertainty (C)
- Targeted Exploration (C)
- RL Method for Time-Constrained Domains (C)
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- **Curious Agents** (P)

Curious Agents

Alternate Evaluation Criteria

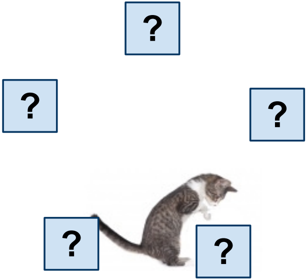
- What does the agent do without external rewards?
- How does the agent explore given a distribution of possible future tasks?

InfoMax [Fasel et al. 2010]



- Partially observable state space
- Agent receives internal rewards proportional to negative entropy of agent's belief distribution
- Learns to take actions to maximize the information it knows about the world

Curious Agents



- Does $b > 0$ with no external rewards compare with InfoMax?
- Can we learn to explore for a distribution of possible future tasks?
- Our agent should focus exploration on parts that are relevant to future tasks, rather than exploring fully

Related Work

- Bayesian RL
- PAC MDP Efficient algorithms
- Intrinsic Motivation
- Generalized Models
- Real Time Architectures

Bayesian RL

- Offers optimal solution to exploration problem [Duff 2003]
- Computationally intractable
- Many approximate solutions:
  - Tie model parameters together [Poupart et al. 2006]
  - Sample from model distributions [Strens 2000, Asmuth et al. 2009]
  - Learn Bayesian optimal policy over time [Kolter and Ng 2009]

Value of Information Approaches

- Model-Based Bayesian Exploration [Dearden et al. 1999]
  - Maintain belief over models
  - Sample and plan on k models
  - Utilize distribution over q-values to calculate VPI: improvement in policy value · probability
  - Add this onto average value from value functions

PAC MDP Efficient Algorithms

- MET-RMAX [Diuk et al. 2009]
  - Assume Transition dynamics are represented by DBN with n binary factors and in-degree D
  - Consider all possible parent combinations $\binom{n}{D}$
  - Separate meteorologist predicts based on each possible parent
  - If any meteorologist does not know the answer, use R_{max}
  - If meteorologists disagree, use R_{max}
  - Remove meteorologists with significantly more error
  - $\binom{n}{D}$ can be very large, D provided

Intrinsic Motivation

- Simsek and Barto [2006]
  - Model-free approach
  - Intrinsic rewards for where value function improves the most
- Intelligent Adaptive Curiosity [Oudeyer et al. 2007]
  - Learn separate dynamics models for different regions of statespace
  - Provide intrinsic rewards based on slope of error curve in each region
  - Only one-step planning, does not use RL/MDP framework

Supervised Learning of Models

- SPITI [Degris et al. 2006]
  - Learn decision tree models for each feature
  - Used ϵ-greedy exploration
- AMBI [Jong and Stone 2007]
  - Instance-based model with relative effects
  - R_{max} bonus for state regions with few visits
- GPRL [Deisenroth and Rasmussen 2009]
  - Use Gaussian Process regression to model dynamics
  - Exploration based on variance of GP predictions
  - Batch mode, agent is provided reward model

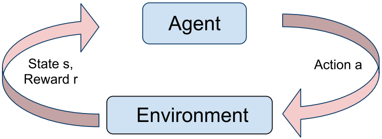
Real-Time Methods

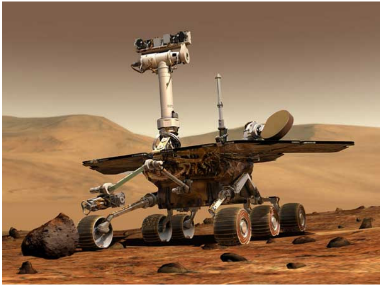
- Dyna Framework [Sutton 1990, 1991]
  - Do Bellman updates on random states using model when not action
  - Still uses tabular model, assumes model update takes insignificant time
- Combining sample-based planning with model-based method
  - With UCT [Silver et al. 2008]
  - With new FSSS [Walsh et al. 2010]
  - Neither places a time restriction on model update or planning

Where will this apply?

- Assumes domains have similar transition and reward effects across states
- Requires factored domains
- Can run in real-time at specified frequency
- Can learn in a limited number of time steps in domain
- Applicable to robots and other real-world problems

Robot Learning





- Real-time sample-efficient reinforcement learning on domains with a limited number of time-steps

Thank You

- Generalized Models (C) [AAMAS, ICML ARL 2009]
- Model Uncertainty (C) [ICDL 2010]
- Targeted Exploration (C) [ICDL 2010]
- RL Method for Time-Constrained Domains (C) [ICRA, ICDL 2010]
- Model Learning with Dependent Feature Transitions (P)
- Extensions to Continuous Domains (P)
- Real-Time Architecture (P)
- Empirical Evaluation (P)
- Curious Agents (P)