Sample Efficient RL

- Objectives
  - Probably Approximately Correct
  - Minimizing regret
  - Bayes-optimal RL
- Today: Model-based data efficient RL
Model-based Sample Efficient RL

- What objective is algorithm optimizing?
- Using function approximation for the model
- Planning with complex models
- Computational constraints
Model Based Approaches

- Linear representations are fairly limited
- Lots of powerful function approximators, e.g.
  - Gaussian processes
  - Random forests
  - Neural networks
Exploration / Exploitation when Using Function Approximation for Models

- When learning a single policy from a batch of data, we didn’t have to address exploration vs exploitation
- Now we are doing online RL
- If using function approximator to represent the transition/reward models, how should we address exploration/exploitation?
Gaussian Process to Model MDP

\[ s' = \Delta + s \]
Gaussian Process:
Explicit Representation of Uncertainty Over Model Parameters

\[ s' = \Delta + s \]
Feature Selection using ARD in GPs

**Problem:** Often there are *many* possible inputs that might be relevant to predicting a particular output. We need algorithms that automatically decide which inputs are relevant.

**Automatic Relevance Determination:**

Consider this covariance function:

\[
K_{nn'} = v \exp \left[ -\frac{1}{2} \sum_{d=1}^{D} \left( \frac{x_n^{(d)} - x_{n'}^{(d)}}{r_d} \right)^2 \right]
\]

The parameter \( r_d \) is the **length scale** of the function along input dimension \( d \).

As \( r_d \to \infty \) the function \( f \) varies less and less as a function of \( x^{(d)} \), that is, the \( d \)th dimension becomes *irrelevant*.

Given data, by learning the lengthscales \( (r_1, \ldots, r_D) \) it is possible to do automatic feature selection.
Can Exploit Structure in Dynamics

After observing 20 transitions, we plot how certain each model is about its predictions for “right”:

GP + ARD detects that the y-coordinate is irrelevant $\implies$ reduced exploration $\implies$ faster learning.
Gaussian Processes for Sample Efficient Reinforcement Learning with RMAX-like Exploration (Jung & Stone, ECML 2010)
General idea:
- Have to learn $D$-dim transition function $x' = f(x, a)$.
- To do this, we combine multiple univariate GPs.

Training:
- Data consists of transitions $\{(x_t, a_t, x'_t)\}_{t=1}^N$, where $x'_t = f(x_t, a_t)$ and $x_t, x'_t \in \mathbb{R}^D$.
- Train independently one GP for each state variable, action.
  - $\mathcal{GP}_{ij}$ models $i$-th state variable under action $a = j$
  - $\mathcal{GP}_{ij}$ has hyperparameters $\tilde{\theta}_{ij}$ found from minimizing marginal likelihood

$$
\min_{\tilde{\theta}_{ij}} \mathcal{L}(\tilde{\theta}_{ij}) = -\frac{1}{2} \log \det (K_{\tilde{\theta}_{ij}} + \sigma I) - \frac{1}{2} y^T (K_{\tilde{\theta}_{ij}} + \sigma I)^{-1} y - \frac{n}{2} \log 2\pi
$$

- Once trained, $\mathcal{GP}_{ij}$ produces for any state $x^*$
  - Prediction $\tilde{f}_i(x^*, a = j) := k_{\tilde{\theta}_{ij}} (x^*)^T (K_{\tilde{\theta}_{ij}} + \sigma I)^{-1} y$.
  - Uncertainty $\tilde{c}_i(x^*, a = j) := k_{\tilde{\theta}_{ij}} (x^*, x^*) - k_{\tilde{\theta}_{ij}} (x^*)^T (K_{\tilde{\theta}_{ij}} + \sigma I)^{-1} k_{\tilde{\theta}_{ij}} (x^*)$.

- At the end, predictions of individual state variables are stacked together.
Remember:
- Input to the planner is the current model.
- The current model "produces" for any \((x, a)\)
  - \(\tilde{f}(x, a)\), the predicted successor state
  - \(\tilde{c}(x, a)\), the associated uncertainty (0=certain, 1=uncertain)

General idea:
- Value iteration on grid \(\Gamma_h\) + multidimensional interpolation.
- Instead of true transition function, simulate transitions with current model.
- As in RMAX integrate "exploration" into value updates.  \(\text{(Nouri & Littman 2009)}\)

Algorithm: iterate \(k = 1, 2, \ldots\): \forall\ node \(\xi_i \in \Gamma_h\), action \(a\)

\[
Q_{k+1}(\xi_i, a) = (1 - \tilde{c}(\xi_i, a)) \cdot \left[ \underbrace{r(\xi_i, a)}_{\text{given a priori}} + \gamma \max_{a'} \underbrace{Q_k(\tilde{f}(\xi_i, a), a')}_{\text{interpolation in } \mathbb{R}^D} \right] + \tilde{c}(\xi_i, a) \cdot V_{\text{MAX}}
\]

Note:
- If \(\tilde{c}(\xi_i, a) \approx 0\), no exploration.
- If \(\tilde{c}(\xi_i, a) \approx 1\), state is artificially made more attractive \(\Rightarrow\) exploration.
Remember:

- Input to the planner is the current model.
- The current model “produces” for any \((x, a)\)
  - \(\tilde{f}(x, a)\), the predicted successor state
  - \(c(x, a)\), the associated uncertainty (0=certainty, 1=uncertain)

General idea:

- Value iteration on grid \(\Gamma_h\) + multidimensional interpolation.
- Instead of true transition function, simulate transitions with current model.
- As in RMAX integrate “exploration” into value updates. (Nouri & Littman 2009)

Algorithm: iterate \(k = 1, 2, \ldots\): \(\forall\) node \(\xi_i \in \Gamma_h\), action \(a\)

\[
Q_{k+1}(\xi_i, a) = (1 - c(\xi_i, a)) \cdot \left[ \underbrace{r(\xi_i, a) + \gamma \max_{a'} Q_k(\tilde{f}(\xi_i, a), a')}_{\text{given a priori}} + \underbrace{c(\xi_i, a) \cdot V_{\text{MAX}}}_{\text{interpolation in } \mathbb{R}^D} \right]
\]

Note:

- If \(c(\xi_i, a) \approx 0\), no exploration.
- If \(c(\xi_i, a) \approx 1\), state is artificially made more attractive \(\implies\) exploration.
Simulation Experiments

Domains:
- Mountain car (2D state space)
- Inverted pendulum (2D state space)
- Bicycle balancing (4D state space)
- Acrobot (swing-up) (4D state space)

Contestants:
- Sarsa(λ) + tilecoding
- GP-RMAXexp (exploration where uncertainty is determined from GP)
- GP-RMAXnoexp (no exploration)
- GP-RMAXgrid (exploration where uncertainty is determined from grid)
GP model with **no exploration** doing best
GP model + no exploration also best in larger domains
Summary

• GP models can be very useful for quickly learning a good dynamics model, especially if there’s structure in the domain
• Planning can be computationally expensive
• In domains considered here, leveraging GP’s representation of model parameter uncertainty not needed

Slide modified from Jung & Stone
ECML 2010
Model Based Approaches

- Linear representations are fairly limited
- Lots of powerful function approximators, e.g.
  - Gaussian processes
  - Random forests
  - Neural networks
TEXPLORE

1. Model generalization for **sample efficiency**
2. Handles **continuous** state
3. Handles actuator **delays**
4. Selects actions continually in **real-time**

Slide modified from Todd Hester
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Citation</th>
<th>Sample Efficient</th>
<th>Real Time</th>
<th>Continuous</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-MAX</td>
<td>Brafman and Tennenholtz, 2001</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Q-LEARNING with F.A.</td>
<td>Watkins, 1989</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SARSA</td>
<td>Sutton and Barto, 1998</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PILCO</td>
<td>Rummery and Niranjan, 1994</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NAC</td>
<td>Peters and Schaal 2008</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>BOSS</td>
<td>Asmuth et al., 2009</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Bayesian DP</td>
<td>Strens, 2000</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MBBE</td>
<td>Dearden et al., 2009</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SPITI</td>
<td>Degris et al., 2006</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MBS</td>
<td>Walsh et al., 2009</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>U-TREE</td>
<td>McCallum, 1996</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DYNAX</td>
<td>Sutton, 1990</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DYNAX-2</td>
<td>Silver et al., 2008</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>KWIK-LR</td>
<td>Strehl and Littman, 2007</td>
<td>Yes</td>
<td>No</td>
<td>Partial</td>
<td>No</td>
</tr>
<tr>
<td>FITTED R-MAX</td>
<td>Jong and Stone, 2007</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DRE</td>
<td>Nouri and Littman 2010</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TEXPLORE</td>
<td>This thesis</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Decision Trees for MDP Model

- Incremental and fast
- Generalize broadly at first, refine over time
- Split state space into regions with similar dynamics
- Good at selecting relevant state features to split on

Slide modified from Todd Hester
Assumption: Relative Effects

- Assume actions have similar effect across states
  - $s^{rel} = \Delta = s' - s$
- $\Delta$ in some cases may be independent of $s$ (or be shared by many states)
Using Decision Trees for Dynamics Model

- Build one tree to predict each state feature and reward
- Combine their predictions: $P(s_{rel} | s, a) = \prod_{i=0}^{n} P(s_{rel} | s, a)$
- Update trees on-line during learning
Representing Uncertainty Over Model: Random Forest

- Create a random forest of $m$ different decision trees [Breiman 2001]
- Each tree is trained on a **random subset** of the agent’s experiences
- Each tree represents a **hypothesis** of the true dynamics of the domain
- How best to use these different hypotheses?

Find effect $s'_{\text{rel}} = s' - s$

Predict $x_{0,\text{rel}}$

Split up by each feature to predict

Add experience with probability $w$

Average $m$ predictions

Random forest of $m$ trees

$s' = s + <x_{0,\text{rel}}, \ldots, x_{n,\text{rel}}>$

Predict $r$
Exploration/Exploitation with Random Forest Model of MDP

<table>
<thead>
<tr>
<th>Bayesian Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BOSS</strong>: Plan over most optimistic model at each action</td>
</tr>
<tr>
<td><strong>MBBE</strong>: Solve each model and use distribution of q-values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEXPLORE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Desiderata</strong>: Explore less, be greedier</td>
</tr>
<tr>
<td>Plan on average of the predicted distributions</td>
</tr>
<tr>
<td>Balance models that are optimistic with ones that are pessimistic</td>
</tr>
</tbody>
</table>
TEXPLORE
Planning Using Random Forest of Models

\[ Q(s, a) = \frac{1}{m} \sum_{i=1}^{m} R_i(s, a) + \gamma \frac{1}{m} \sum_{i=1}^{m} \sum_{s'} P_i(s'|s, a) \max_{a'} Q(s', a') \]

- Essentially, compute an average model (from random forest)
- Then use that for planning
- Some computational advantages too

Equation from Hester & Stone JMLR 2013
MCTS for TEXPLORE Planning

- **Simulate trajectory** from current state using model (rollout)
- Use upper confidence bounds to select actions (UCT [Kocsis and Szepesvári 2006])
- Focus computation on states the agent is most likely to visit
- **Anytime**—more rollouts, more accurate value estimates
- Update value function at each state in rollout
TEXPLORE: Planning using UCT

```
procedure PLAN-POLICY(M, s)  
  UCT-RESET()  
  while time available do  
    UCT-SEARCH(M, s, 0)  
  end while  
end procedure

▷ Approximate planning from state s using model M
```
TEXPLORE: Reuse Tree Across Time Steps

\begin{verbatim}
procedure PLAN-POLICY\(M, s\)  
  \(\triangleright\) Approximate planning from state \(s\) using model \(M\)
  UCT-RESET()
  while time available do
    UCT-SEARCH\(M, s, 0\)
  end while
end procedure

procedure UCT-RESET()
  \(\triangleright\) Lower confidence in v.f. since model changed
  for all \(s_{\text{disc}} \in S_{\text{disc}}\) do
    if \(c(s_{\text{disc}}) > \text{resetCount} \cdot |A|\) then
      \(c(s_{\text{disc}}) \leftarrow \text{resetCount} \cdot |A|\)
    end if
  end for
  for all \(a \in A\) do
    if \(c(s_{\text{disc}}, a) > \text{resetCount}\) then
      \(c(s_{\text{disc}}, a) \leftarrow \text{resetCount}\)
    end if
  end for
end procedure
\end{verbatim}
**TEXPLORE: UCT + lambda-returns**

```plaintext
procedure PLAN-POLICY(M, s)  ▷ Approximate planning from state s using model M
    UCT-RESET()
    while time available do
        UCT-SEARCH(M, s, 0)
    end while
end procedure

procedure UCT-SEARCH(M, s, d)  ▷ Rollout from state s at depth d using model M
    if TERMINAL or d = maxDepth then
        return 0
    end if
    s\_disc ← DISCRETIZE(s, nBins, minVals, maxVals)  ▷ Get discretized version of state s
    a ← argmax\_a′ (Q(s\_disc, a′) + 2 \cdot \frac{r\_max - r\_min}{1-\gamma} \cdot \sqrt{\frac{\log c(s\_disc)}{c(s\_disc, a′)}})  ▷ Note: Ties broken randomly
    (s′, r) ← M \rightarrow QUERY-MODEL(s, a)  ▷ Algorithm 4
    sampleReturn ← r + γUCT-SEARCH(M, s′, d + 1)  ▷ Continue rollout from state s′
    c(s\_disc) ← c(s\_disc) + 1
    c(s\_disc, a) ← c(s\_disc, a) + 1
    Q(s\_disc, a′) ← α \cdot sampleReturn + (1 - α) \cdot Q(s\_disc, a′)
    return λ \cdot sampleReturn + (1 - λ) \cdot \max\_a′ Q(s\_disc, a′)  ▷ Use λ-returns
end procedure
```

Figure from Hester & Stone JMLR 2013
TEXPLORE

Slide modified from Todd Hester
Simulations on Car Driving

1. **TEXPLORE**
2. $\epsilon$-greedy exploration ($\epsilon = 0.1$)
3. **Boltzmann** exploration ($\tau = 0.2$)
4. **VARIANCE-BONUS** Approach $\nu = 1$ [Deisenroth & Rasmussen 2011]
5. **VARIANCE-BONUS** Approach $\nu = 10$
6. Bayesian DP-like Approach (use sampled model for 1 episode) [Strens 2000]
7. **BOSS-like** Approach (use optimistic model) [Asmuth et al. 2009]

First five approaches use **TEXPLORE’s model**
Adding $\epsilon$-greedy, Boltzmann, or Bayesian DP-like exploration does not improve performance.
Comparing to Other Approaches

1. **BOSS** (Sparse Dirichlet prior) [Asmuth et al. 2009]
2. **Bayesian DP** (Sparse Dirichlet prior) [Strens 2000]
3. **PILCO** (Gaussian Process Regression model) [Deisenroth & Rasmussen 2011]
4. **R-MAX** (Tabular model) [Brafman & Tennenholtz 2001]
5. **Q-LEARNING** using tile-coding [Watkins 1989]
TEXPLORE accrues **significantly more rewards** than all the other methods after episode 24 ($p < 0.01$).
Most of state space is very **predictable**
- But fuel stations have **varying costs**
- 317,688 State-Actions, Time-Constrained Lifetime: 635,376 actions
- Seed experiences of goal, fuel station, and running out of fuel

Slide modified from Todd Hester
1. TEXPLORE (Greedy w.r.t. aggregate model)
2. $\epsilon$-greedy exploration ($\epsilon = 0.1$)
3. Boltzmann exploration ($\tau = 0.2$)
4. VARIANCE-BONUS Approach $\nu = 10$ [Deisenroth & Rasmussen 2011]
5. Bayesian DP-like Approach (use sampled model for 1 episode) [Strens 2000]
6. BOSS-like Approach (use optimistic model) [Asmuth et al. 2009]
7. BOSS (Sparse Dirichlet prior) [Asmuth et al. 2009]
8. Bayesian DP (Sparse Dirichlet prior) [Strens 2000]
TEXPLORE learns the **fastest** and **accumulates the most cumulative reward** of any of the methods.

TEXPLORE learns the task **within the time-constrained lifetime** of 635,376 steps.
Model Accuracy

- **Regression tree forest and single regression tree** have significantly less error than all the other models in predicting the next state ($p < 0.001$).
- For reward, regression tree is significantly better than all models but GP regression after 205 state-actions ($p < 0.001$).
Does it work on real car?

Yes! It learns the task in 2 minutes (< 11 episodes)

Slide modified from Todd Hester
TEXPLORE

• Uses random forests to represent MDP dynamics & rewards
• Uses MCTS for planning
• In domains presented, little explicit exploration needed
Summary: Model-based Sample Efficient RL

- What objective is algorithm optimizing?
  - Today, empirical performance. No formal guarantees
- Using function approximation for the model can greatly speed learning (can exploit structure in dynamics model)
- Exploration / exploitation
  - Do we need to explicitly explore?
  - We’ll always explore things that look promising
  - In results saw today, didn’t need much explicit exploration
- Planning with complex models
  - Can be computationally prohibitive
  - Approximate approaches, like MCTS, useful