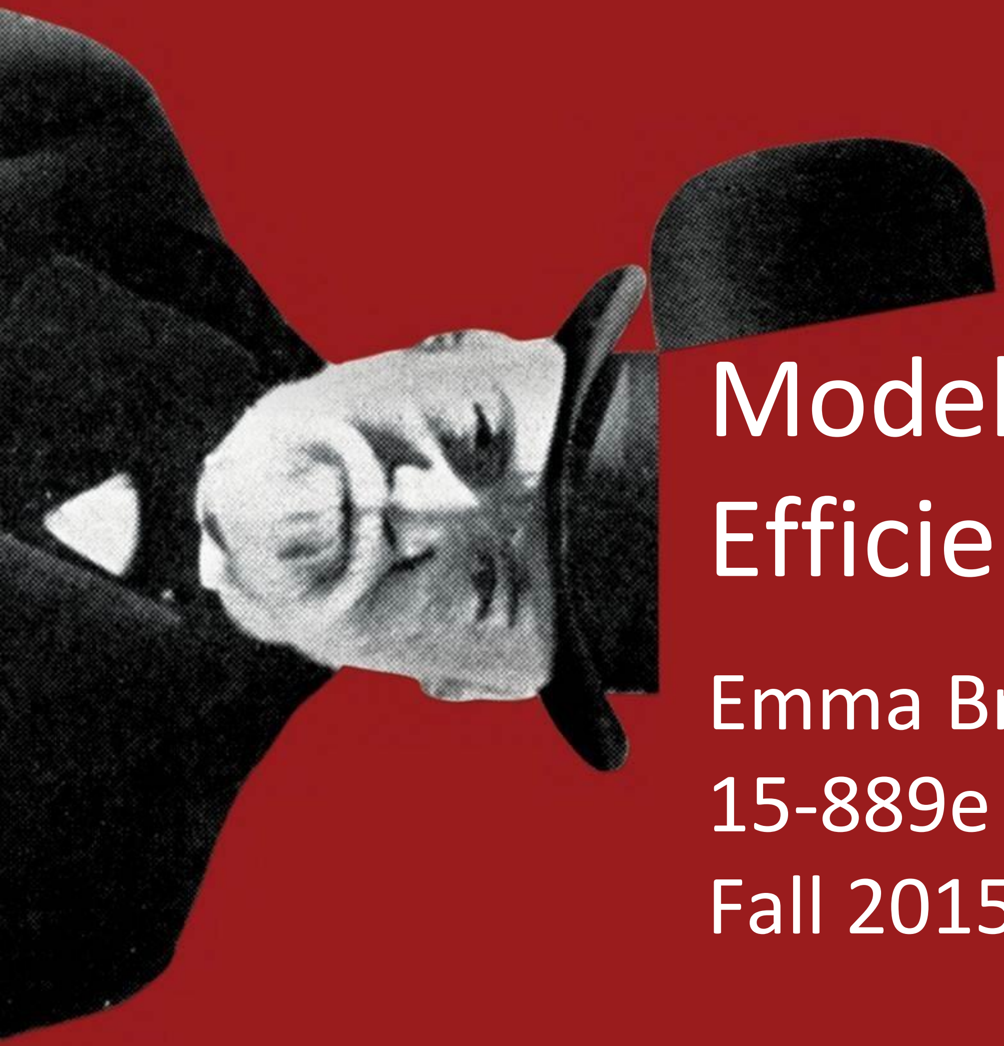


Model-based Sample Efficient RL

Emma Brunskill

15-889e

Fall 2015

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

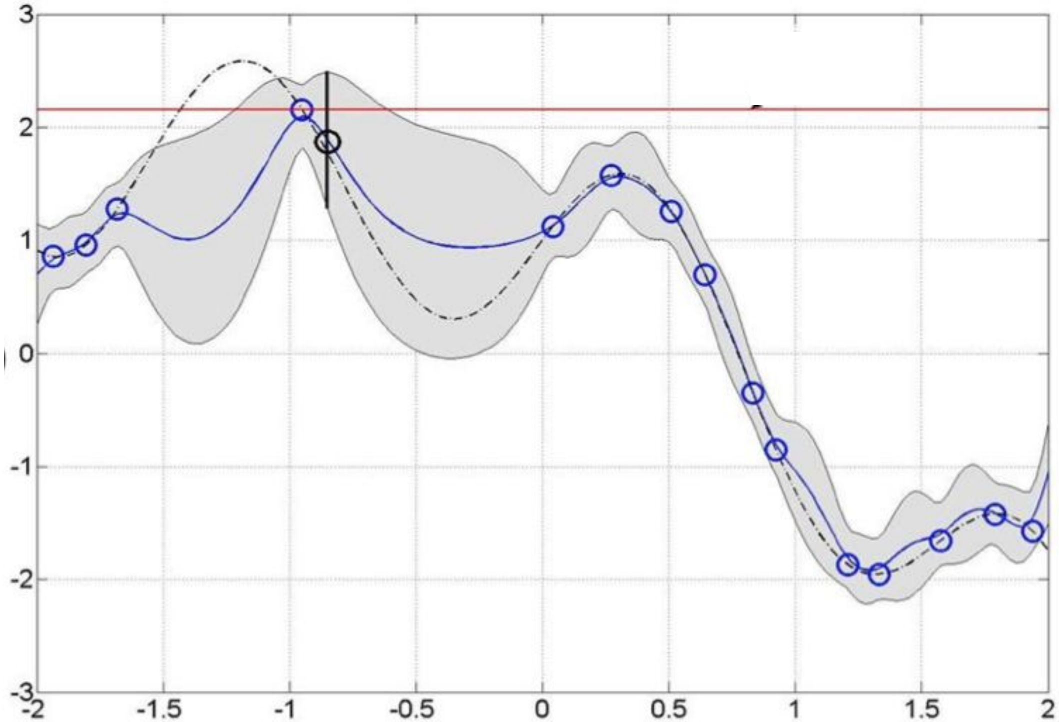
Exploration / Exploitation when Using Function Approximation for Models

- 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' = ***Δ*** + s

Δ



s

Figure adjusted from Wilson et al. JMLR 2014

Carnegie Mellon University

Gaussian Process:

Explicit Representation of Uncertainty Over Model Parameters

$s' = \Delta + s$

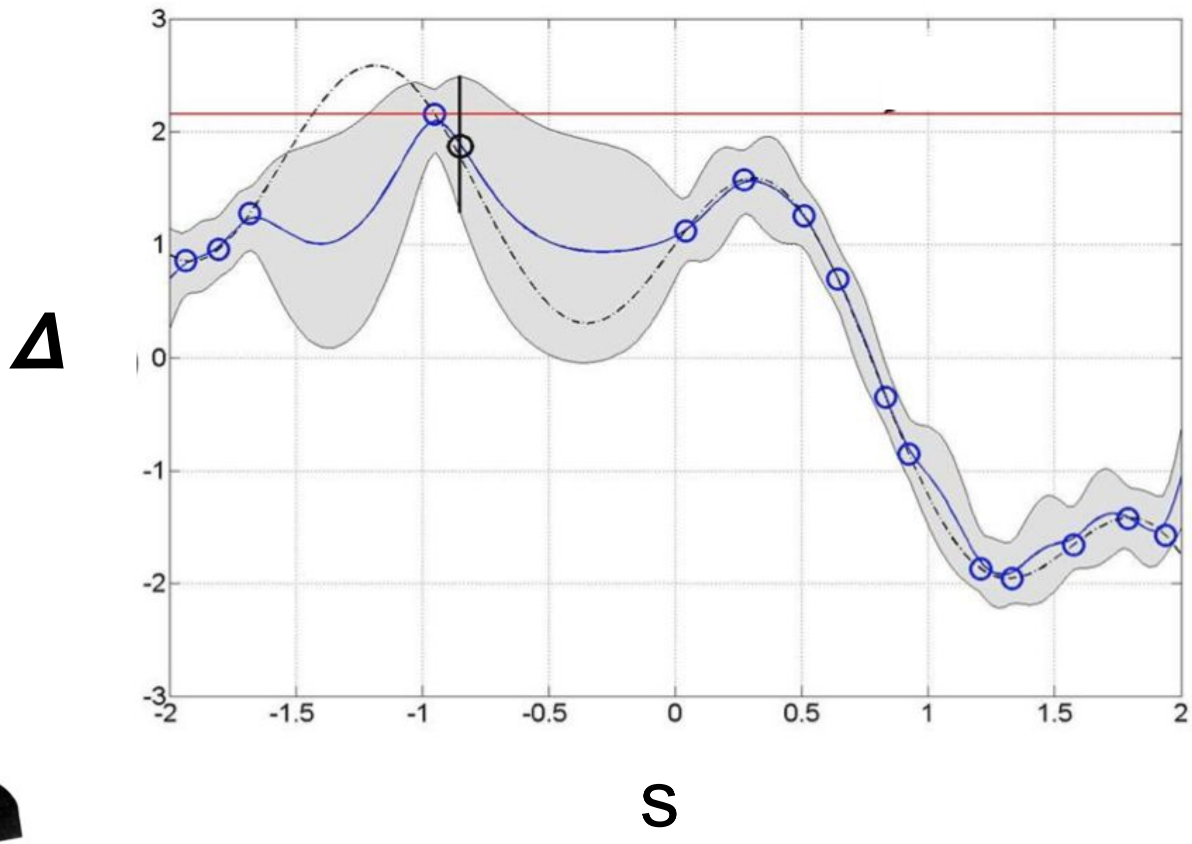


Figure adjusted from Wilson et al. JMLR 2014

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:

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

The parameter r_d is the length scale of the function along input dimension d.

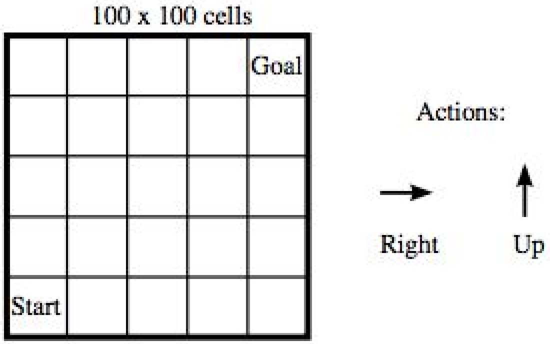
As $r_d \to \infty$ the function f varies less and less as a function of $x^{(d)}$, that is, the dth dimension becomes *irrelevant*.

Given data, by learning the lengthscales $(r_1, \ldots, r_D)$ it is possible to do automatic feature selection.

Slide from Ghahramani

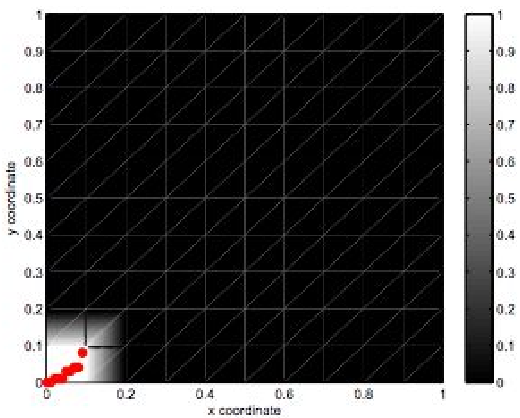
http://www.eurandom.tue.nl/events/workshops/2010/YESIV/Prog-Abstr_files/Ghahramani-lecture2.pdf

Can Exploit Structure in Dynamics

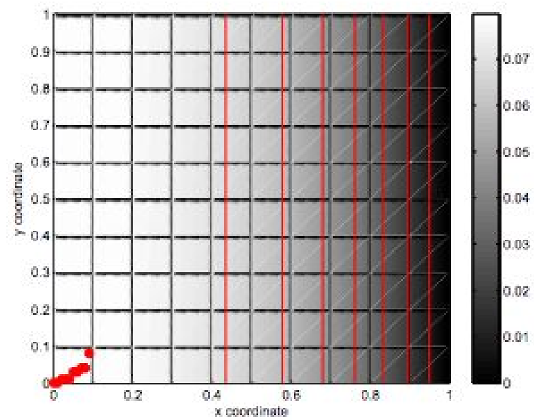


$$
\begin{aligned}
x_{new}^{right} &= x_{old} + 0.01 \\
y_{new}^{right} &= y_{old} \\
x_{new}^{up} &= x_{old} \\
y_{new}^{up} &= y_{old} + 0.01
\end{aligned}
$$

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



10×10 grid

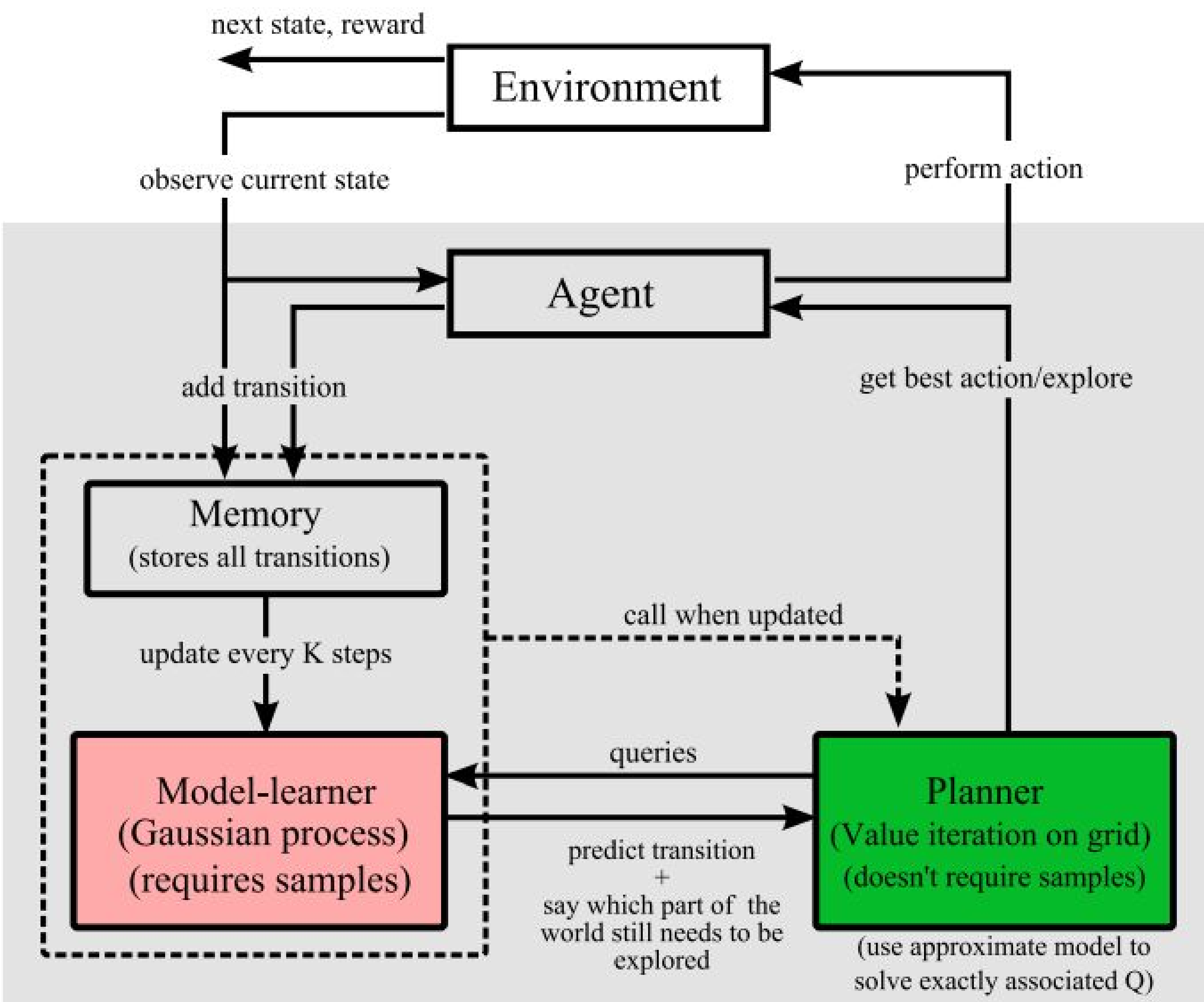


GP with ARD kernel

GP+ARD detects that the y-coordinate is irrelevant $\Longrightarrow$ reduced exploration $\Longrightarrow$ faster learning.

Slide modified from Jung & Stone ECML 2010

Gaussian Processes for Sample Efficient Reinforcement Learning with RMAX-like Exploration (Jung & Stone, ECML 2010)



Slide modified from Jung & Stone ECML 2010

General idea:

- Have to learn D-dim transition function $\mathbf{x}' = f(\mathbf{x}, a)$.
- To do this, we combine multiple univariate GPs.

Training:

- Data consists of transitions $\{(\mathbf{x}_t, a_t, \mathbf{x}'_t)\}_{t=1}^{N}$, where $\mathbf{x}'_t = f(\mathbf{x}_t, a_t)$ and $\mathbf{x}_t, \mathbf{x}'_t \in \mathbb{R}^D$.
- 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 $\vec{\theta}_{ij}$ found from minimizing marginal likelihood

$$\min_{\vec{\theta}_{ij}} \mathcal{L}(\vec{\theta}_{ij}) = -\frac{1}{2}\log\det(\mathbf{K}_{\vec{\theta}_{ij}} + \sigma\mathbf{I}) - \frac{1}{2}\mathbf{y}^{\top}(\mathbf{K}_{\vec{\theta}_{ij}} + \sigma\mathbf{I})^{-1}\mathbf{y} - \frac{n}{2}\log 2\pi$$

  - Once trained, $\mathcal{GP}_{ij}$ produces for any state $\mathbf{x}^*$
    - Prediction $\tilde{f}_i(\mathbf{x}^*, a = j) := \mathbf{k}_{\vec{\theta}_{ij}}(\mathbf{x}^*)^{\top}(\mathbf{K}_{\vec{\theta}_{ij}} + \sigma\mathbf{I})^{-1}\mathbf{y}$.
    - Uncertainty $\tilde{c}_i(\mathbf{x}^*, a = j) := k_{\vec{\theta}_{ij}}(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_{\vec{\theta}_{ij}}(\mathbf{x}^*)^{\top}(\mathbf{K}_{\vec{\theta}_{ij}} + \sigma\mathbf{I})^{-1}\mathbf{k}_{\vec{\theta}_{ij}}(\mathbf{x}^*)$.
- At the end, predictions of individual state variables are stacked together.

Slide modified from Jung & Stone ECML 2010

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 Γ_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

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

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 $\Longrightarrow$ exploration.

Slide modified from Jung & Stone ECML 2010

Computational Cost

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)

This is expensive!

Hard to scale to large dim state spaces

General idea:

- Value iteration on grid Γ_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

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

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 $\Longrightarrow$ exploration.

Slide modified from Jung & Stone ECML 2010

Simulation Experiments

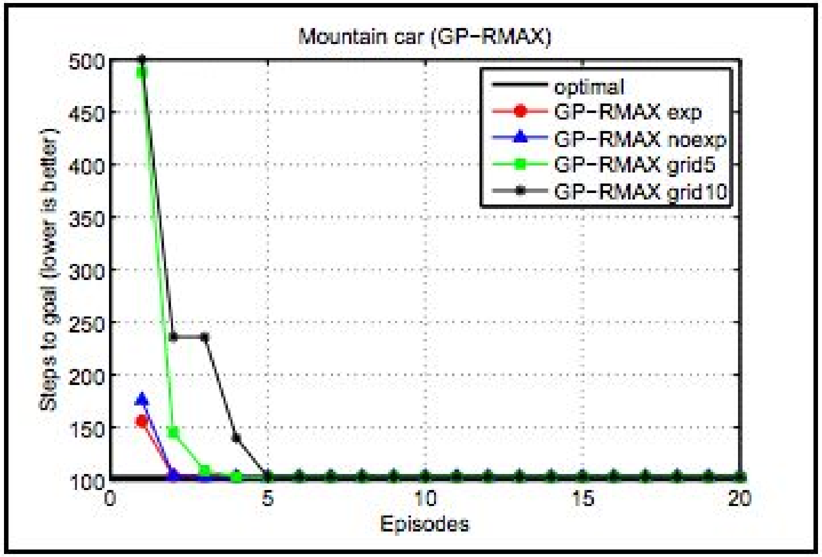
Domains:

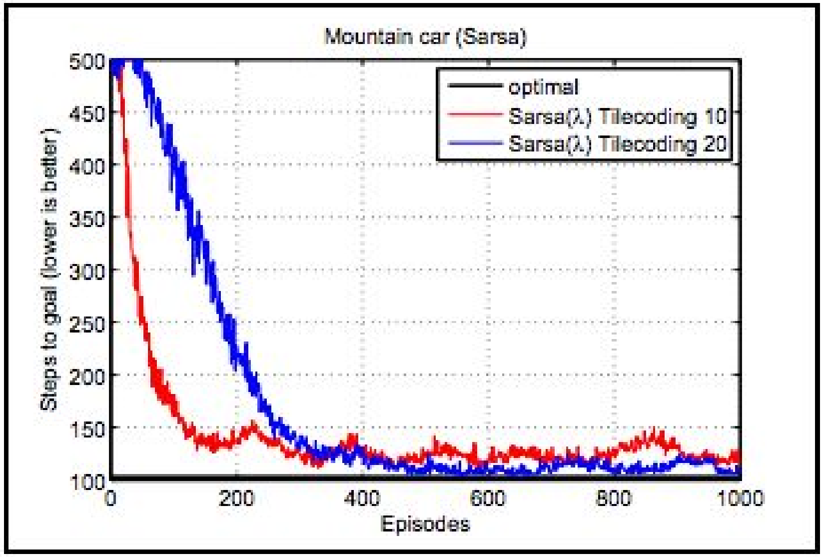
- Mountain car (2D state space)
- Inverted pendulum (2D state space)
- Bicycle balancing (4D state space)
- Acrobot (swing-up) (4D state space)

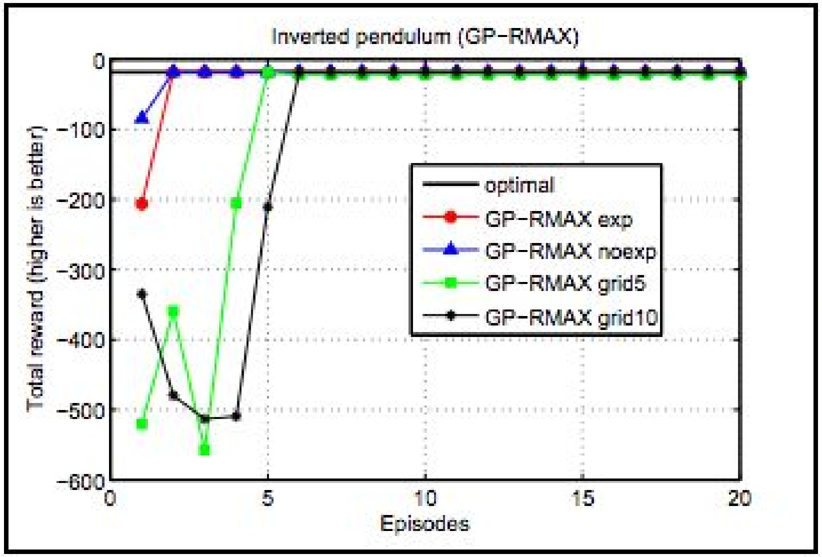
Contestants:

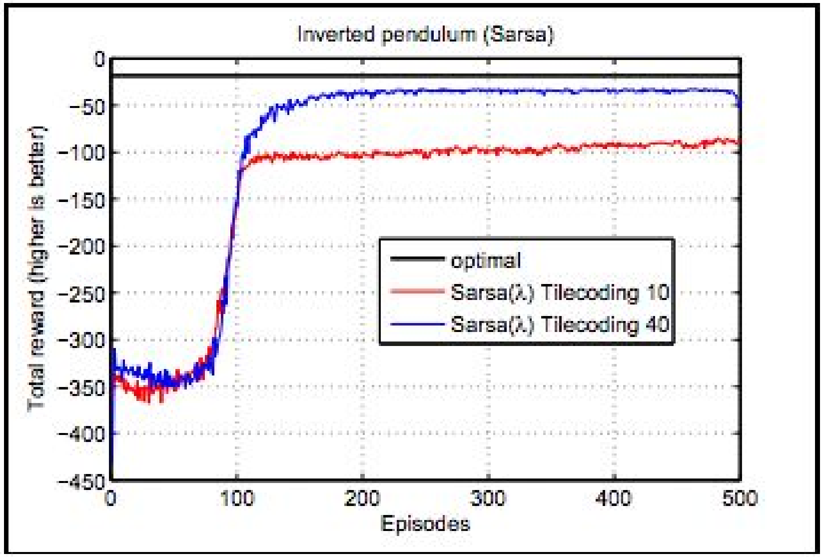
- Sarsa(λ) + tilecoding
- GP-RMAXexp (exploration where uncertainty is determinded from GP)
- GP-RMAXnoexp (no exploration)
- GP-RMAXgrid (exploration where uncertainty is determined from grid)

Slide modified from Jung & Stone
ECML 2010



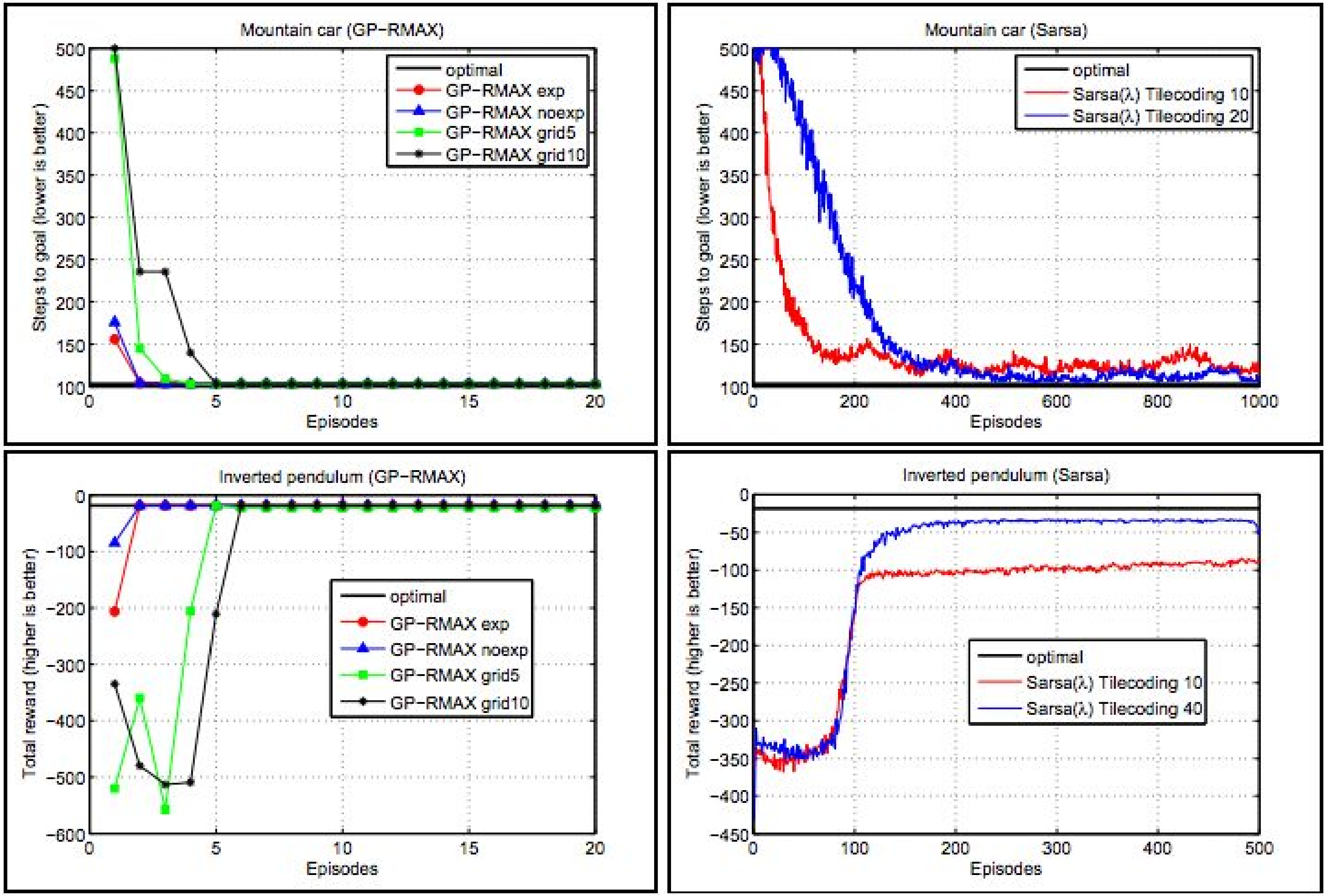






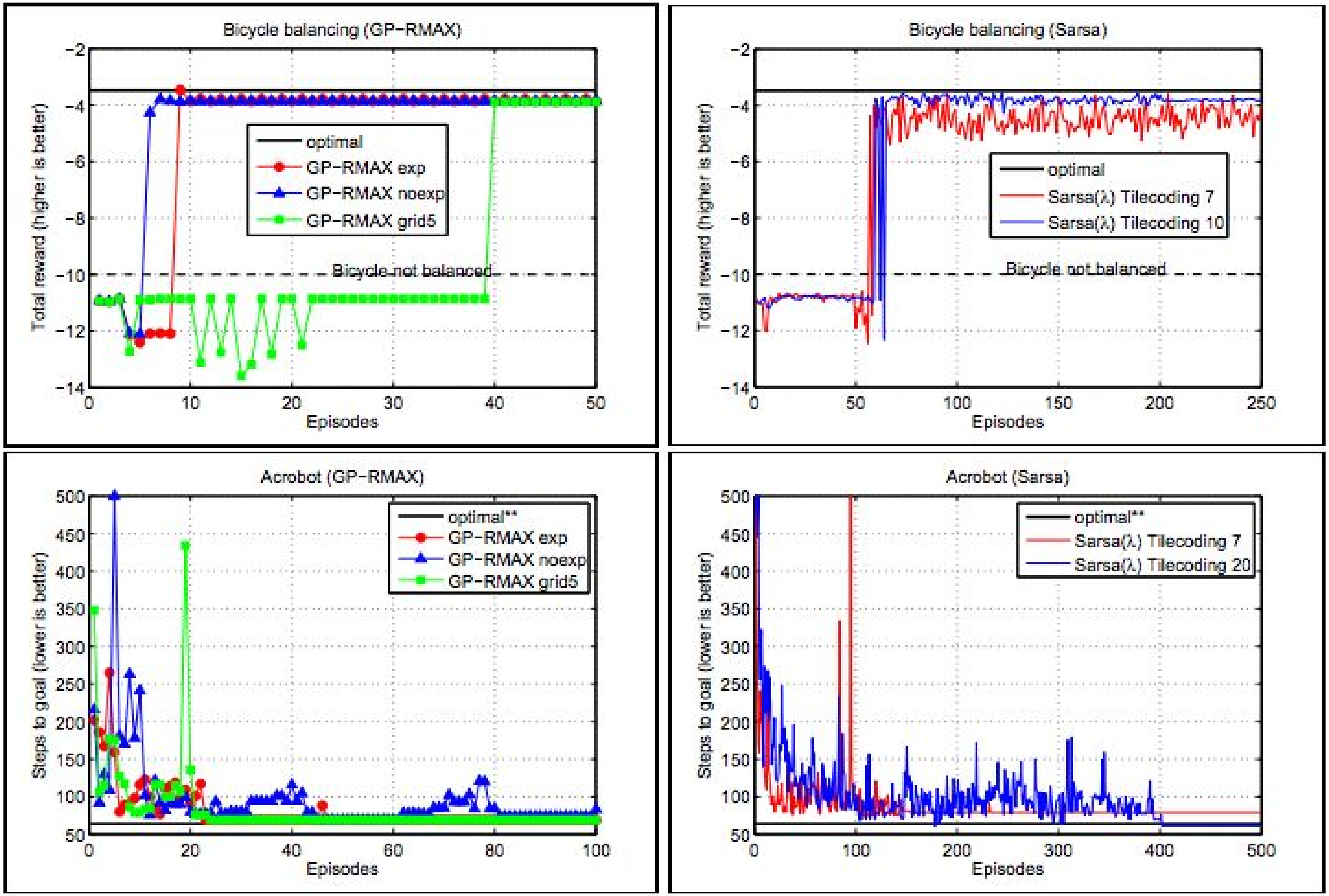
Slide modified from Jung & Stone ECML 2010

GP model with **no exploration** doing best



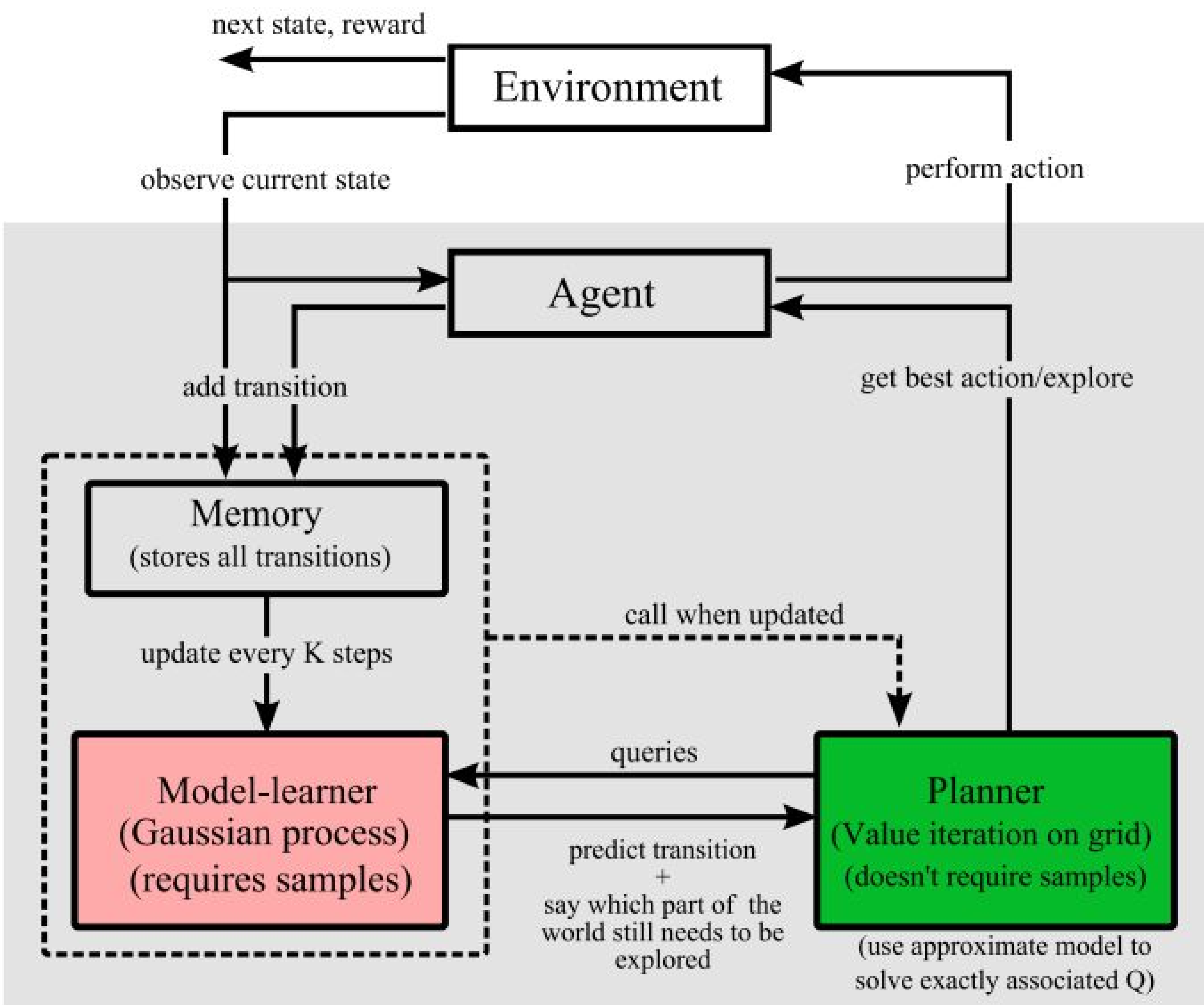
Slide modified from Jung & Stone ECML 2010

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

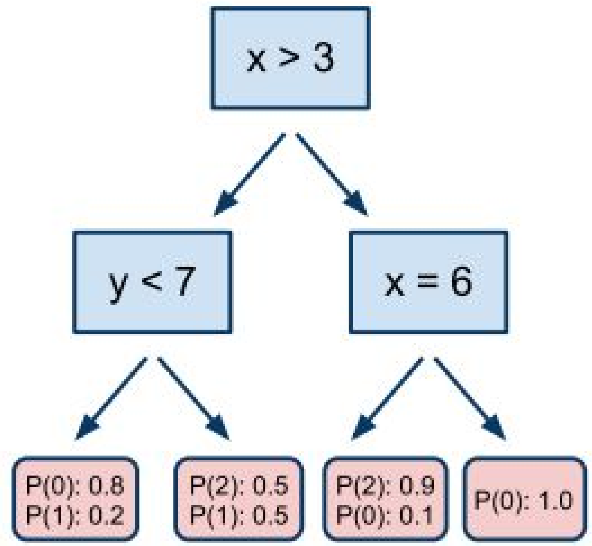
TEXPLORE

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

Algorithm	Citation	Sample Efficient	Real Time	Continuous	Delay
R-MAX	Brafman and Tennenholtz, 2001	Yes	No	No	No
Q-LEARNING	Watkins, 1989	No	Yes	No	No
with F.A.	Sutton and Barto, 1998	No	Yes	Yes	No
SARSA	Rummery and Niranjan, 1994	No	Yes	No	No
PILCO	Deisenroth and Rasmussen, 2011	Yes	No	Yes	No
NAC	Peters and Schaal 2008	Yes	No	Yes	No
BOSS	Asmuth et al., 2009	Yes	No	No	No
Bayesian DP	Strens, 2000	Yes	No	No	No
MBBE	Dearden et al., 2009	Yes	No	No	No
SPITI	Degris et al., 2006	Yes	No	No	No
MBS	Walsh et al., 2009	Yes	No	No	Yes
U-TREE	McCallum, 1996	Yes	No	No	Yes
DYNA	Sutton, 1990	Yes	Yes	No	No
DYNA-2	Silver et al., 2008	Yes	Yes	Yes	No
KWIK-LR	Strehl and Littman, 2007	Yes	No	Partial	No
FITTED R-MAX	Jong and Stone, 2007	Yes	No	Yes	No
DRE	Nouri and Littman 2010	Yes	No	Yes	No
TEXPLORE	This thesis	Yes	Yes	Yes	Yes

Decision Trees for MDP Model

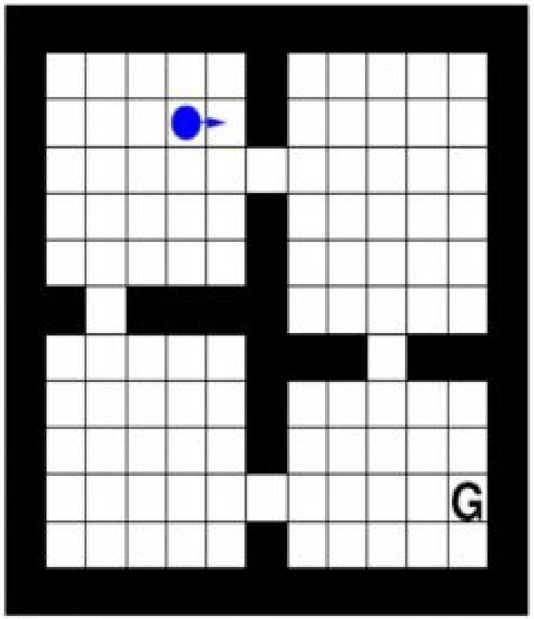


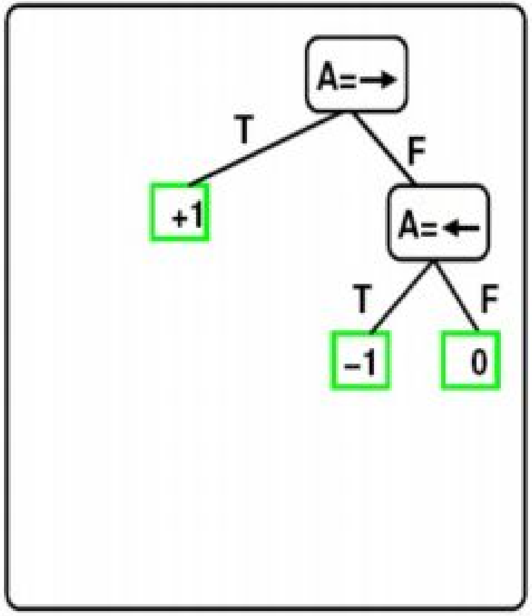
- Incremental and fast
- Generalize broadly at first, refine over time
- Split state space into regions with similar dynamics
- Good at selecting relevant state features to split on

Assumption: Relative Effects

- Assume actions have similar effect across states
- $s^{rel} = \Delta = s' - s$
- Δ in some cases may be independent of s (or be shared by many states)
  - Brunskill et al. 2008, Leffler et al. 2007, Jong & Stone 2007

Using Decision Trees for Dynamics Model





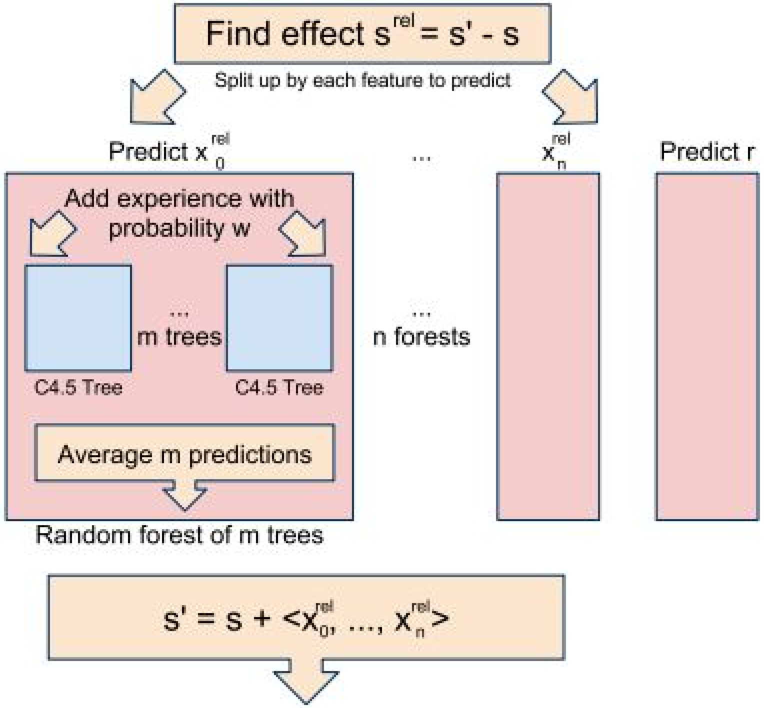
- Build one tree to predict each state feature and reward
- Combine their predictions: $P(s^{rel}|s,a) = \Pi_{i=0}^{n} P(s_i^{rel}|s,a)$
- Update trees on-line during learning

Slide modified from Todd Hester

Carnegie Mellon University

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?



Exploration/Exploitation with Random Forest Model of MDP

Bayesian Approaches

- BOSS: Plan over most optimistic model at each action
- MBBE: Solve each model and use distribution of q-values

TEXPLORE

- Desiderata: Explore less, be greedier
- Plan on average of the predicted distributions
- Balance models that are optimistic with ones that are pessimistic

Slide modified from Todd Hester

TEXPLORE

Planning Using Random Forest of Models

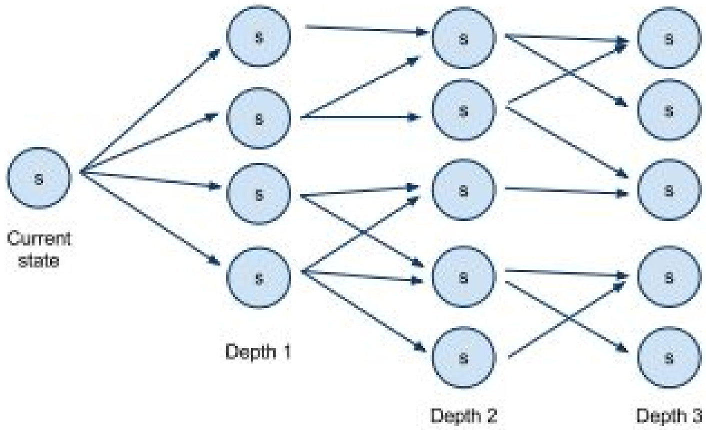
$$Q(s,a) = \frac{1}{m}\sum_{i=1}^{m} R_i(s,a) + \gamma \frac{1}{m}\sum_{i=1}^{m}\sum_{s'} P_i(s'|s,a)\max_{a'} Q(s',a')$$

- 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



Slide modified from Todd Hester

TEXPLORE: Planning using UCT

```
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
```

Figure from Hester & Stone JMLR 2013

TEXPLORE: Reuse Tree Across Time Steps

```
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-RESET()                ▷ Lower confidence in v.f. since model changed
    for all s_disc ∈ S_disc do                          ▷ For all discretized states
        if c(s_disc) > resetCount · |A| then
            c(s_disc) ← resetCount · |A|                ▷ resetCount per action
        end if
        for all a ∈ A do
            if c(s_disc, a) > resetCount then
                c(s_disc, a) ← resetCount
            end if
        end for
    end for
end procedure
```

Figure from Hester & Stone JMLR 2013

TEXPLORE: UCT + lambda-returns

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
  $s_{disc} \leftarrow$ DISCRETIZE($s, nBins, minVals, maxVals$) ▷ Get discretized version of state s
  $a \leftarrow \text{argmax}_{a'} \left(Q(s_{disc}, a') + 2 \cdot \frac{r_{max} - r_{min}}{1-\gamma} \cdot \sqrt{\frac{\log c(s_{disc})}{c(s_{disc}, a')}} \right)$ ▷ Note: Ties broken randomly
  $(s', r) \leftarrow M \Rightarrow$ QUERY-MODEL(s, a) ▷ Algorithm 4
  $sampleReturn \leftarrow r + \gamma$UCT-SEARCH($M, s', d+1$) ▷ Continue rollout from state s'
  $c(s_{disc}) \leftarrow c(s_{disc}) + 1$ ▷ Update counts
  $c(s_{disc}, a) \leftarrow c(s_{disc}, a) + 1$
  $Q(s_{disc}, a') \leftarrow \alpha \cdot sampleReturn + (1 - \alpha) \cdot Q(s_{disc}, a')$
  return $\lambda \cdot sampleReturn + (1 - \lambda) \cdot \max_{a'} Q(s_{disc}, a')$ ▷ Use λ-returns
end procedure

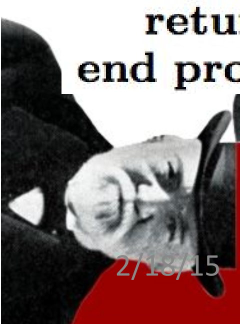
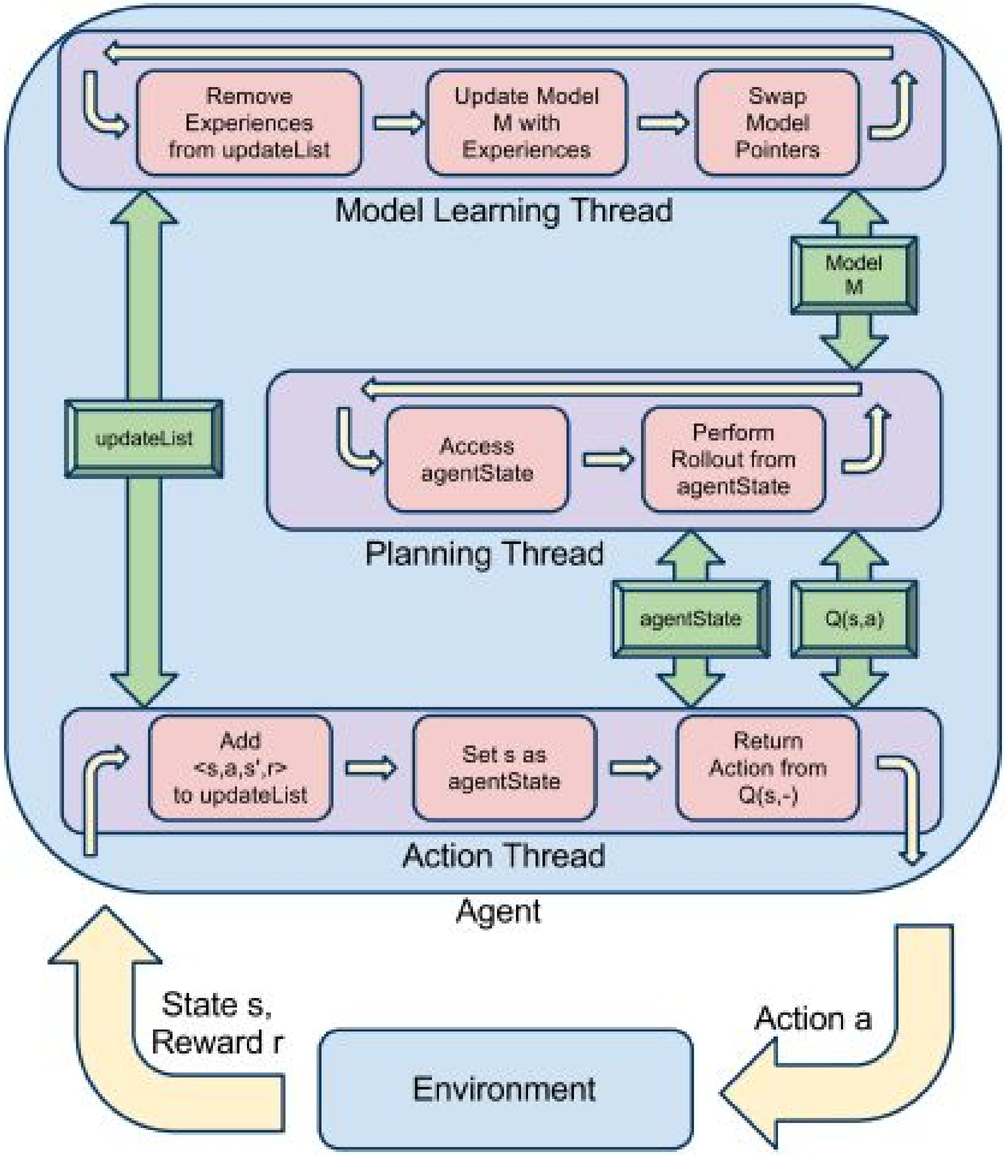


Figure from Hester & Stone JMLR 2013

TEXPLORE

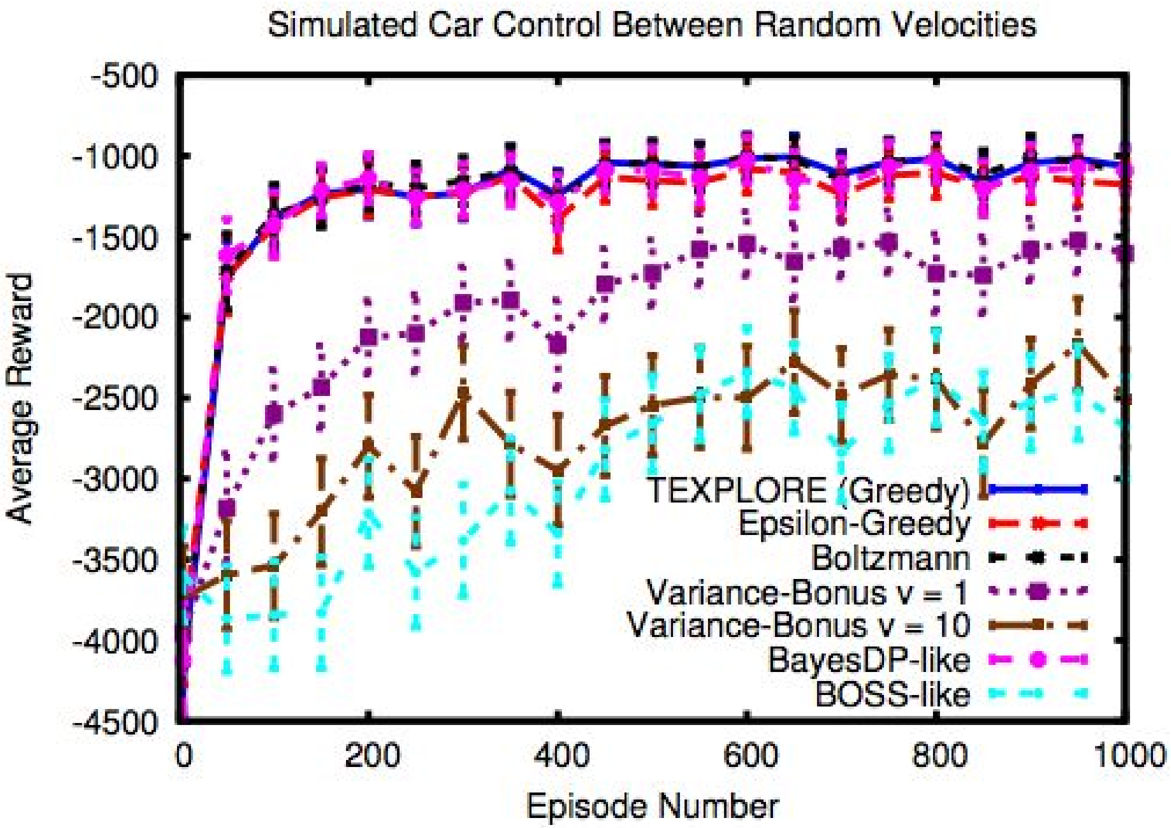


Slide modified from Todd Hester

Simulations on Car Driving

1. TEXPLORE
2. ϵ-greedy exploration ($\epsilon = 0.1$)
3. Boltzmann exploration ($\tau = 0.2$)
4. VARIANCE-BONUS Approach $v = 1$ [Deisenroth & Rasmussen 2011]
5. VARIANCE-BONUS Approach $v = 10$
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**



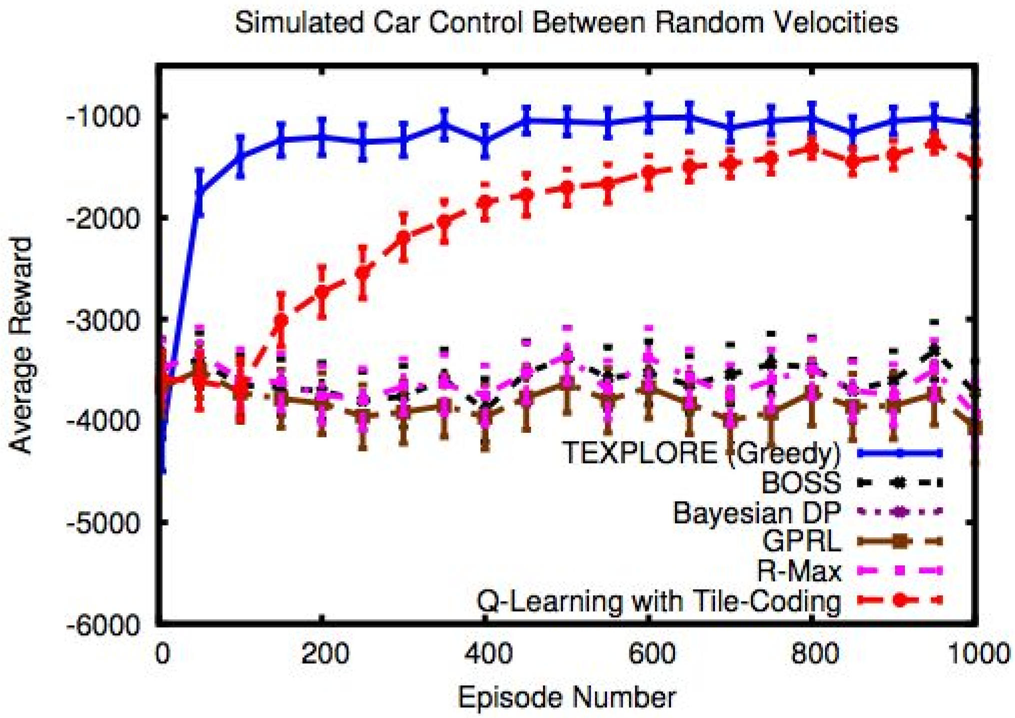
- Adding ϵ-greedy, Boltzmann, or Bayesian DP-like exploration **does not** improve performance

Slide modified from Todd Hester

Comparing to Other Approaches

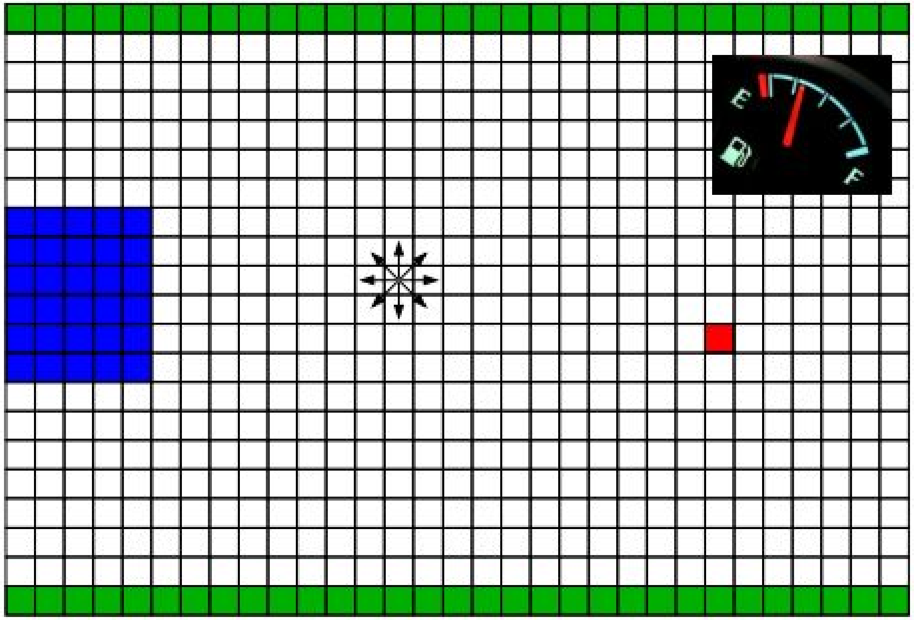
1. BOSS (Sparse Dirichlet prior) [Asmuth et al. 2009]
2. Bayesian DP (Sparse Dirichlet prior) [Strens 2000]
3. PILCO (Gaussian Process Regression model) [Deisenroth & Rasmussen 2011]
4. R-MAX (Tabular model) [Brafman & Tennenholtz 2001]
5. Q-LEARNING using tile-coding [Watkins 1989]

Slide modified from Todd Hester



- TEXPLORE accrues **significantly more rewards** than all the other methods after episode 24 ($p < 0.01$).

Fuel World

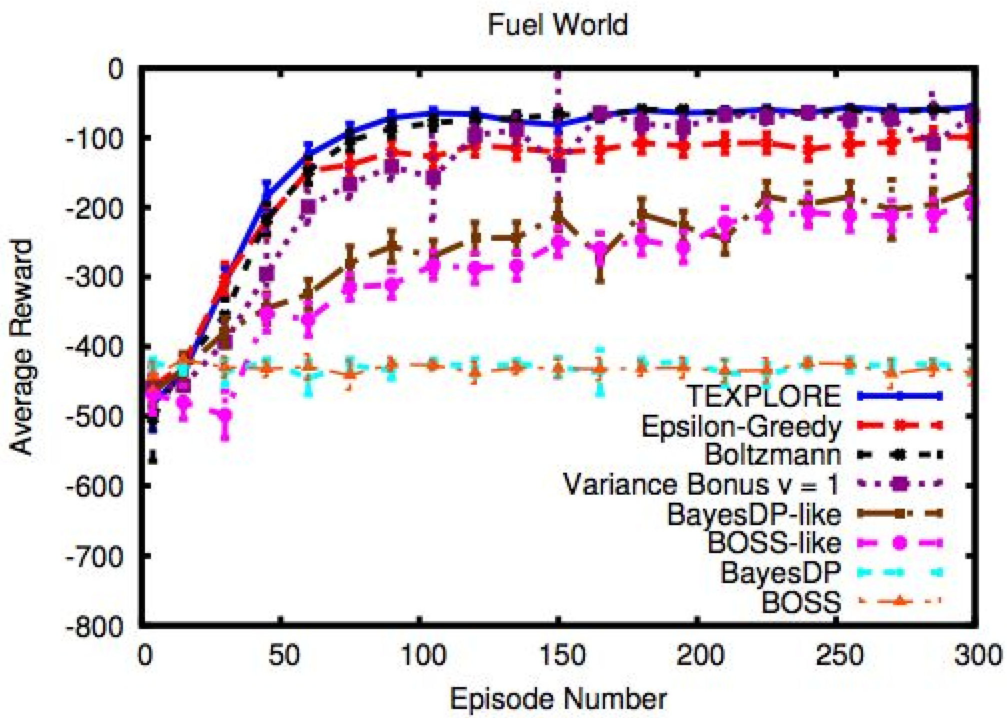


- Most of state space is very **predictable**
- But fuel stations have **varying costs**
- 317, 688 State-Actions, Time-Constrained Lifetime: 635, 376 actions
- Seed experiences of goal, fuel station, and running out of fuel

Slide modified from Todd Hester

1. TEXPLORE (Greedy w.r.t. aggregate model)
2. ϵ-greedy exploration ($\epsilon = 0.1$)
3. Boltzmann exploration ($\tau = 0.2$)
4. VARIANCE-BONUS Approach $v = 10$ [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]

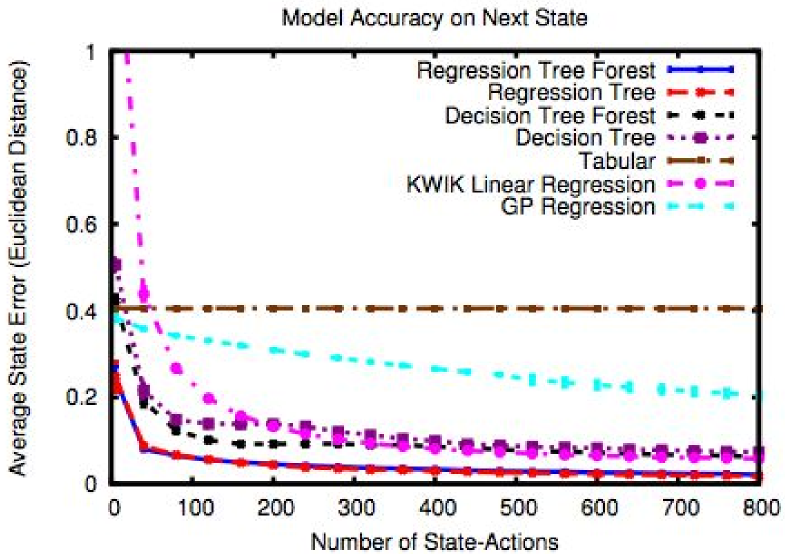
Slide modified from Todd Hester

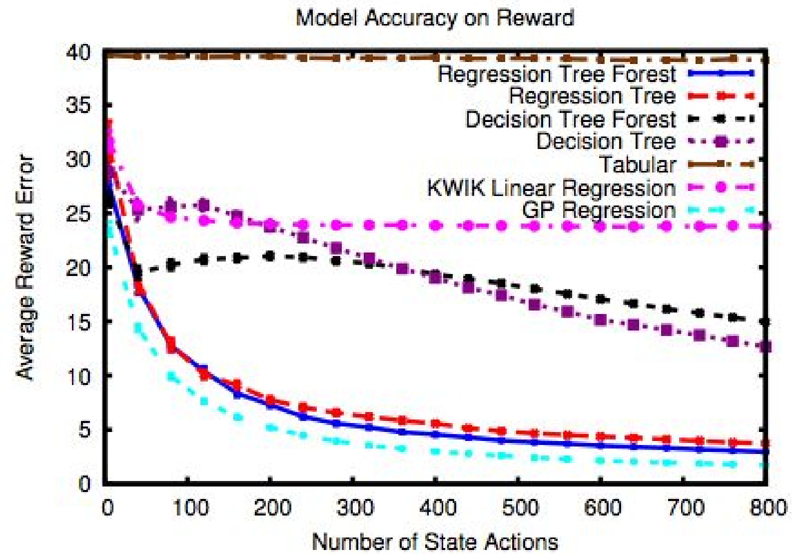


- TEXPLORE learns the **fastest** and **accrues the most cumulative reward** of any of the methods.
- TEXPLORE learns the task **within the time-constrained lifetime** of 635, 376 steps.

Slide modified from Todd Hester

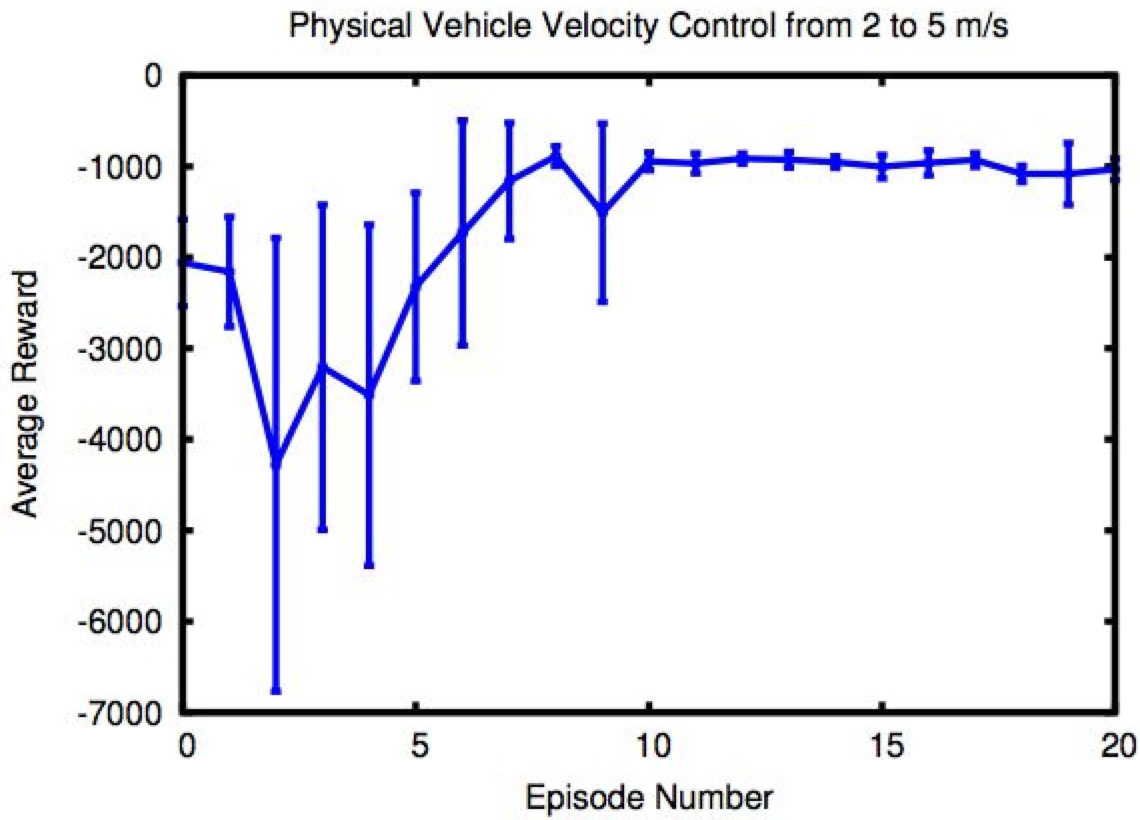
Model Accuracy





- **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?

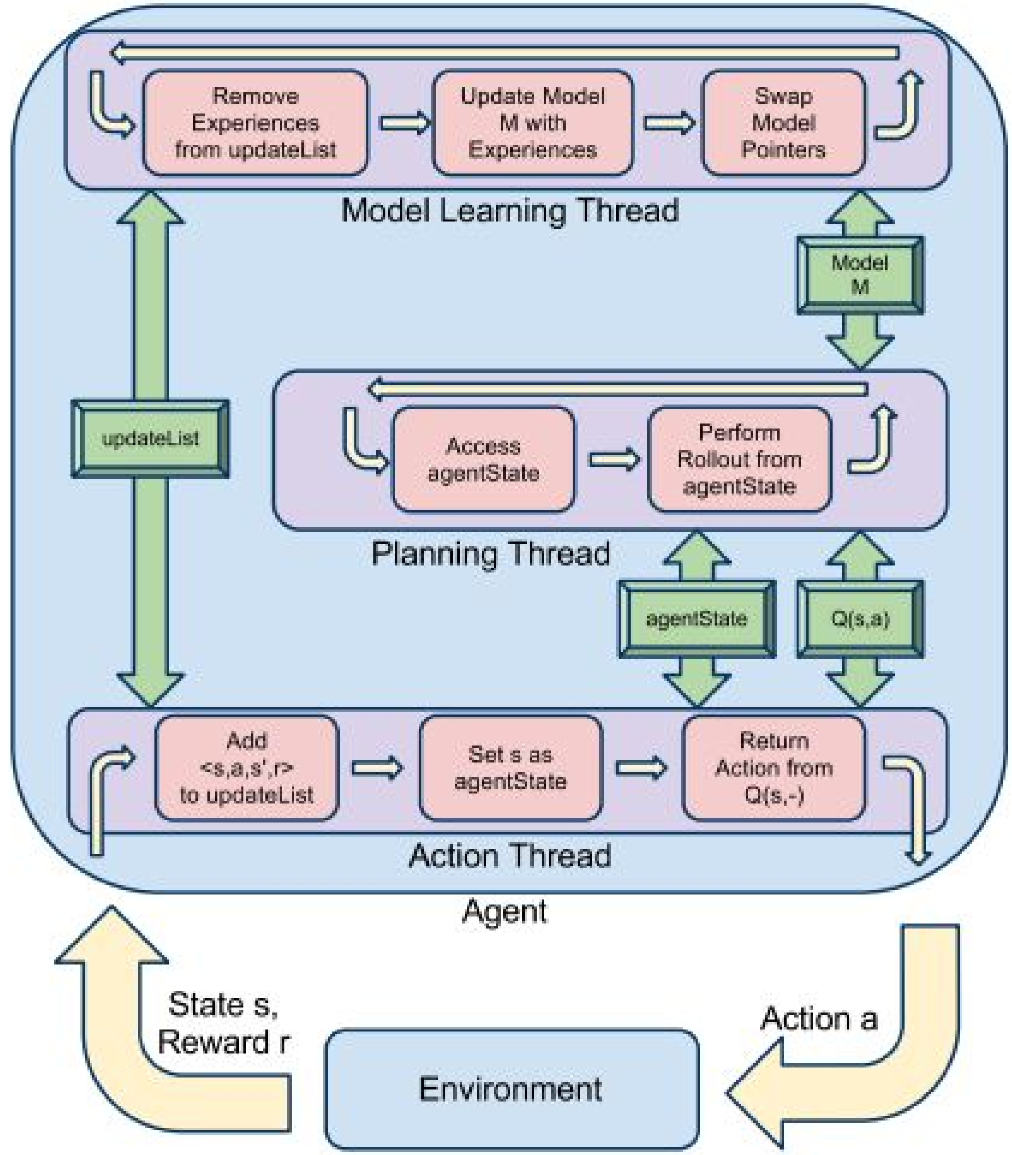


- **Yes!** It learns the task in **2 minutes** (< 11 episodes)

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