Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning

Jeff Schneider

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Jeff.Schneider@cs.cmu.edu

Abstract

Model learning combined with dynamic programming has been shown to be effective for learning control of continuous state dynamic systems. The simplest method assumes the learned model is correct and applies dynamic programming to it, but many approximators provide uncertainty estimates on the fit. How can they be exploited? This paper addresses the case where the system must be prevented from having catastrophic failures during learning.

We propose a new algorithm adapted from the dual control literature and use Bayesian locally weighted regresssion models with dynamic programming. A common reinforcement learning assumption is that aggressive exploration should be encouraged. This paper addresses the converse case in which the system has to reign in exploration. The algorithm is illustrated on a 4 dimensional simulated control problem.

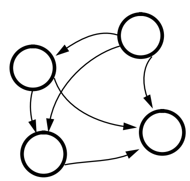
Safety in MDPs and Mazes?

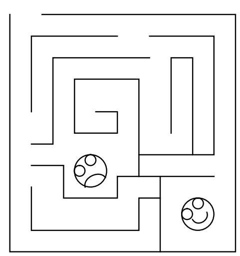
The problem formulation usually directly addresses the stochastic nature of the outcomes of actions. Therefore, policies are "safe" in the sense that they optimize expected values given their model.

Unfortunately , acquisition of the model still isn't safe. Disasters have to happen before they can be known about, and thus avoided. Exploration mechanisms tend to be random and often have an "exploration must be encouraged" philosophy .

In problems that really have continuous dynamics, is it possible to avoid catastrophes without ever experiencing them?

And what would that mean for exploration strategies?





How can we learn dynamic control of non-linear systems safely?

Use Reinforcement Learning?

Pro: Good at handling non-linear models and reward functions

Con: Often, exploration / exploitation issues are handled with random algorithms. This is particularly dangerous in real, physical systems.

Use traditional adaptive/dual control methods?

Pro: Explicitly addresses the need to acquire new data for learning while not incurring large penalties

Con: Restricted to linear or other easily parameterizable systems.

Can we put the best parts of each together?

A Description of the Problem

Given a system with unknown dynamics:

$$x^{k+1} = f\left(x^k, u^k\right) + \text{noise}$$

Given a known additive cost function for a single trial:

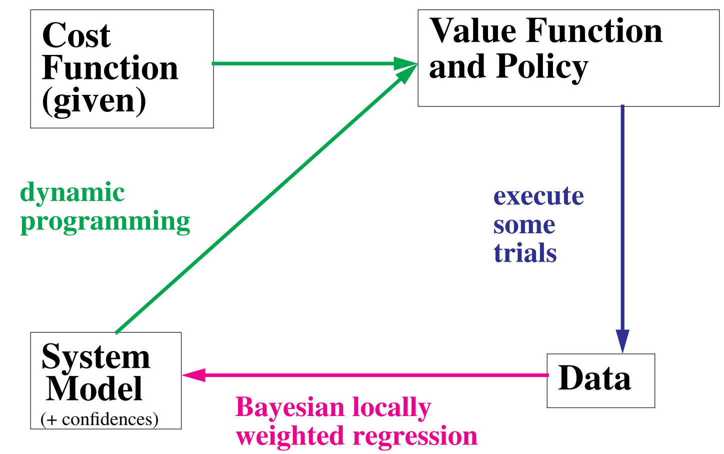
$$J = \sum_{k=0}^{N} L\left(x^k, u^k, k\right)$$

Possibly given a nominal controller:

$$u^k = g\left(x^k, k\right)$$

Control the system over M trials and minimize the total cost incurred over those trials.

The T op Level Algorithm



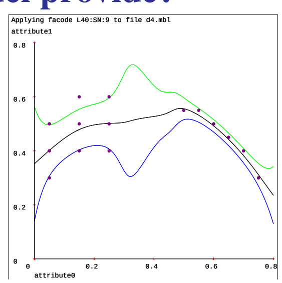
What does the learned model provide?

Predictions: Covered in Part I

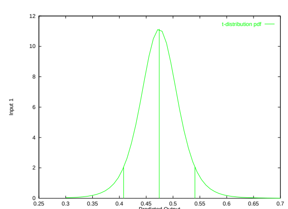
Confidence Intervals on Predictions

Use a localized version of classical regression analysis methods. It requires the X'X, X'Y , and Y'Y matrices. The distribution of the model coefficients is a multi-d t distribution.

The distribution of the true output is a t distribution computed by a projection of the model coefficients distribution. The mean of that distribution is the prediction. The distribution is used to obtain prediction confidence intervals.

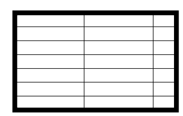


Distribution at query 0.15



Bayesian Regression

Here is some data:



Give me all my favorite estimates.

HELP!

Does this ever really happen?

YES!

With local weighting the query may be outside the influence of all the data.

It happens when making decisions about collecting data.

It happens when considering possible control decisions that have never been tried before.

Bayesian Regression

- allows easy specification of priors that take over when little or no data is present
- keeps numeric computation stable
- ridge regression is a special case

Bayesian Regression

A convenient prior on the model coefficients, M, and the precision, R (noise variance inverse), is a *normal-gamma* distribution with parameters:

- τ - the variance (uncertainty) of the normal prior on the coefficients, M
- μ - the mean of the normal prior on the coefficients, M
- α, β - the parameters of a gamma distribution on the precision, R

The data is n data points. For locally weighted learning, use n = Σw_i, the sum of the weights on the data points.

The posterior distribution of R is a gamma distribution with parameters α' and β'.

(see [DeGroot 1970] for details).

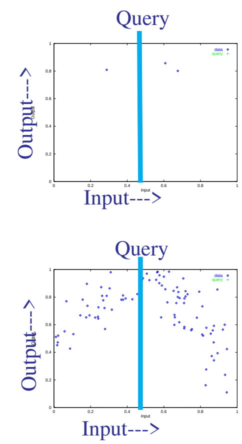
$$\alpha' = \alpha + \frac{n}{2},\ \beta' = \beta + \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \frac{\tau\mu + nm}{\tau + n}\right)y_i + \tau\mu\left(\mu - \frac{\tau\mu + nm}{\tau + n}\right)$$

The posterior distribution of M is a t distribution with parameters:

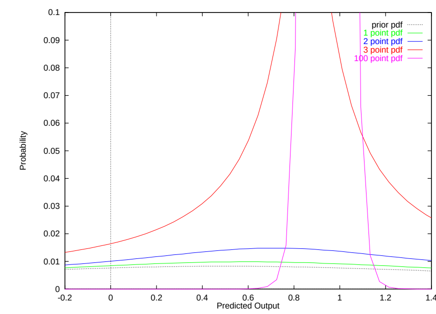
mean = $(\tau\mu + nm)/(\tau + n)$ precision = $\alpha'\tau/\beta'$ degrees of freedom = $2\alpha'$

The posterior distribution of the predicted output for a given input is a projection of the model coefficients t distribution (also a t distribution)

Locally Weighted Bayesian Regression: An Example

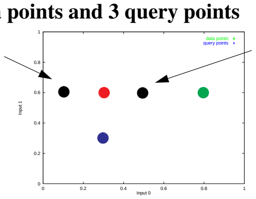


PDFs on predicted output at input 0.47 using local kernel regression with various amounts of data

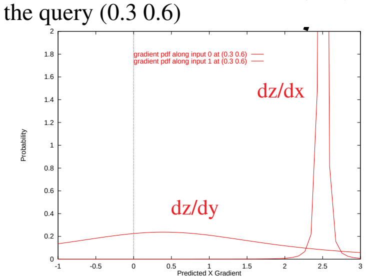


Locally Weighted Bayesian Regression: Another Example

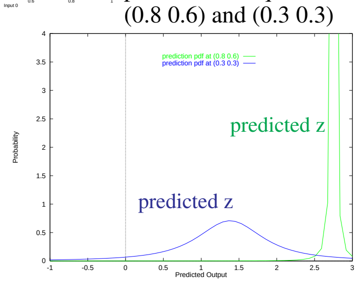
2 data points and 3 query points



Distribution of X and Y predicted gradients at the query (0.3 0.6)



Distribution of output predictions at queries (0.8 0.6) and (0.3 0.3)



Dynamic Programming in Continuous State Spaces with Discretization

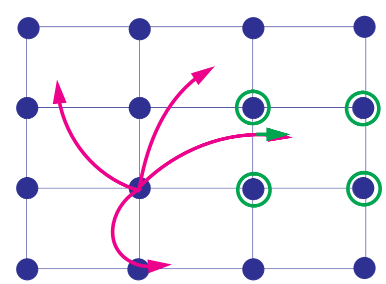
Use value iteration to solve the Bellman equation.

$$V^*\left(x^k\right) = \min_{u \in U} L\left(x^k, u^k\right) + V^*\left(f\left(x^k, u^k\right)\right)$$

Discretize the state space. Compute and store the value function at each grid point.

Perform the min operation by discretizing the action space and considering each one.

Estimate the value function at the next state with interpolation in the grid.

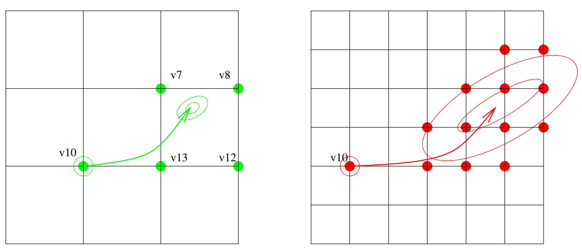


DP with model uncertainty

$$V^*\left(x^k\right) = \min_{u \in U} L\left(x^k, u^k\right) + E\left(V^*\left(f\left(x^k, u^k\right)\right) \middle| \hat{f}\right)$$

High Confidence Next State

Low Confidence Next State



$$V_{10} = L(x, u) + 0.4V_7 + 0.3V_8 + 0.2V_{11} + 0.1V_{12}$$

$$V_{10} = L(x, u) + \frac{\sum_{\{x' | p(x') > \varepsilon\}} p(x' | \hat{f}, x, u)\, V(x')}{\sum_{\{x' | p(x') > \varepsilon\}} p(x' | \hat{f}, x, u)}$$

Experiments: Fast Cart-Pole Maneuvers

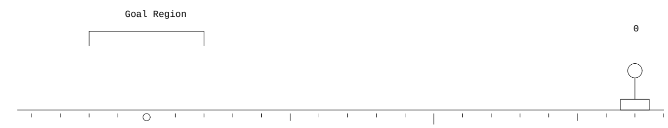
$$\mathbf{x} = \begin{bmatrix} \text{pole angle } (\theta) \\ \text{pole angular velocity} \\ \text{cart position } (\rho) \\ \text{cart velocity} \end{bmatrix} \qquad \mathbf{u} = \text{force applied to cart}$$

The state equations are taken from [Barto *et al*, 1993].

$$J = \sum_{i=0}^{x(i) = \text{goal}} x_i^T x_i + 0.01 u_i^T u_i$$

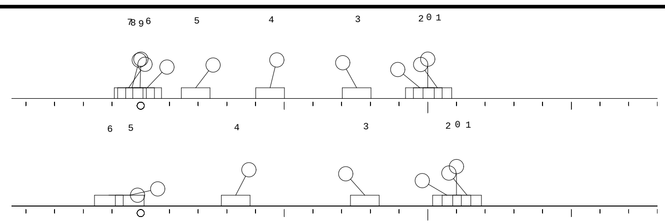
Additional penalty of 1,000,000 if the pole is dropped. This penalty is also incurred in the DP solutions if the state goes outside the given bounds on it.

The controller chooses constant controls for 0.1 second intervals



Experiments: Learning LQR Control

1. Start the system from [0 0 0 0]
2. For 10 time steps choose controls randomly from N(0,0.25)
3. Repeat a second time to get a total of 20 data points
4. Use the data to fit a global linear model of the state equations
5. Build an LQR controller



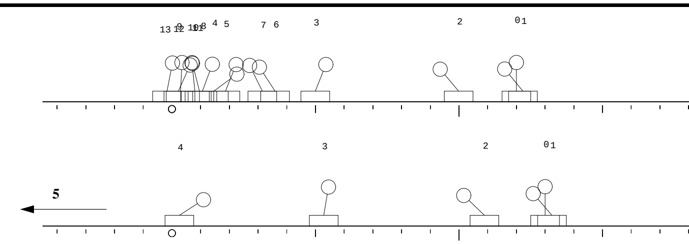
With only 20 data points to build a globally linear model, LQR easily stabilizes the system.

It is able to successfully bring the system back to the goal from as far away as an initial state of [0 0 10 0].

If it is started any further away , it crashes the system.

Experiments: DP assuming a correct Bayesian L WR model

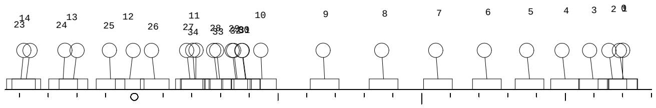
1. Begin with the LQR controller from before. Generate 50 data points by running it from an initial state of [0 0 1 0] with control noise as before.
2. Build an L WR model mapping: $[\theta, \dot{\theta}, u] \rightarrow [\ddot{\theta}, \ddot{\rho}]$
3. Discretize the states to [1 1 9 1 1 9] levels within the ranges [+-1.0 +-4.0 +-21.0 +-20.0] and the control to 15 levels in the range +-30.0.
4. Perform deterministic DP and record the policy at each grid node.
5. Interpolate the policy between grid nodes during execution.



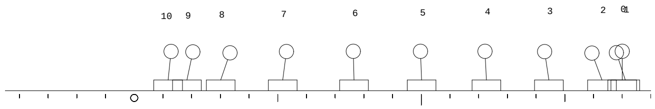
The controller is able to successfully bring the system back from an initial state of [0 0 12 0], but crashes if started farther away .

3 Iterations of DP with Bayesian L WR

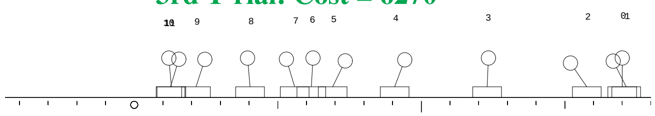
1st T rial: Cost = 12393



2nd T rial: Cost = 71 14

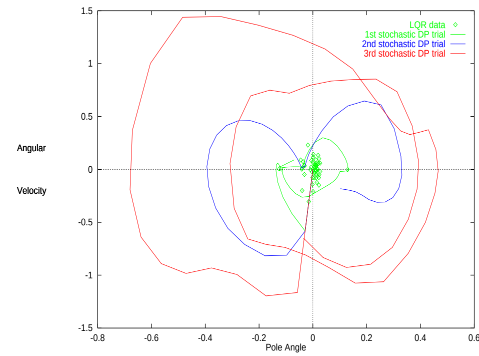


3rd T rial: Cost = 6270



Later T rials: No Further Improvement

Phase Plot of Executions



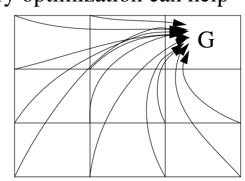
At each iteration, the controller gets to use the data from all previous trials and gets more aggressive.

Future W ork

What if the state space is just too large to complete dynamic programming on it?

Local T rajectory Optimization

[Atkeson, 1993] showed how local trajectory optimization can help dynamic programming on grids.



Rather than building a full optimal policy for the whole state space, compute a single trajectory from the initial conditions and optimize it locally including model uncertainty considerations. Then build a controller that attempts to stay on the trajectory .

Future W ork

Dual control also addresses the issue of probing (encouraging exploration that might not otherwise happen).

Can this be done with locally weighted regression and dynamic programming?

Dual control handles probing by considering the variances in its parameter estimates as additional state variables. It increases the size of the state space, but can be done with linear , parametric models.

But L WR is non-parametric and any extra variables are very expensive for dynamic programming!

HELP!