Control Model Learning for Whole-Body Mobile Manipulation

Scott Kuindersma
Computer Science Department
University of Massachusetts Amherst
scottk@cs.umass.edu

Introduction

The ability to discover the effects of actions and apply this knowledge during goal-oriented action selection is a fundamental requirement of embodied intelligent agents. This requirement is most clearly demonstrated when the agent, or robot in our case, must continue to meet objectives in changing environmental contexts. For example, a humanoid robot that encounters a slippery surface might use its arms to stabilize and avoid a fall. In our ongoing work, we hope to demonstrate the utility of learned control models for whole-body mobile manipulation. In what follows we discuss preliminary work on learning a forward model of the dynamics of a balancing robot exploring simple arm movements. This model is then used to construct whole-body control strategies for regulating state variables using arm motion.

Approach

Model Learning

We assume the discrete-time dynamics of the system are captured by the function

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t),$$

where $\mathbf{s}_t$ and $\mathbf{a}_t$ are the state and action vectors at time t. Since we do not in general have access to f, our goal is to learn an approximation, $\hat{f}$, which we represent using a linear-Gaussian basis function model,

$$\mathbf{s}_{t+1} = \hat{f}(\mathbf{s}_t, \mathbf{a}_t) + \epsilon \equiv \mathbf{W}\boldsymbol{\psi}(\mathbf{s}_t, \mathbf{a}_t) + \epsilon,$$

where it is assumed that $\epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma)$ accounts for unmodeled disturbances. The term $\boldsymbol{\psi}(\mathbf{s}_t, \mathbf{a}_t)$ is a k-dimensional feature vector derived from a set of k basis functions.

To completely specify the model, we must provide or learn the weight matrix $\mathbf{W}$ and the ordered set of basis functions Ψ used to compute the feature vector. Although there is much recent work on the automatic discovery of basis functions, e.g. (Mahadevan et al. 2006), in our preliminary experiments we elected to use a fixed set of basis functions. The matrix $\mathbf{W}$ can be learned by applying standard multiple-output linear regression techniques. However, depending on the choice of Ψ and method for discovering $\mathbf{W}$, the model may be prone to overfitting. Since we do not want to make any strong assumptions about Ψ, we focus our attention on algorithms that parsimoniously select basis functions that explain much of the variation in the target variable without overfitting.

To serve this purpose we employed the lasso algorithm (Tibshirani 1996), which is a well-known l_1 regularization technique. In addition to penalizing large weights, the use of the l_1 norm creates a boundary in weight space that tends to produce solutions with multiple zero-valued weights, resulting in basis selection. Using an efficient implementation of the lasso algorithm (Efron et al. 2002), one can generate a *set* of models in the same order of computation as a *single least squares fit*. Intuitively, this set of candidate models is generated by systematically increasing the degree to which large weights are penalized, thus producing models ranging from a single predictor to the full least squares fit using all k features.

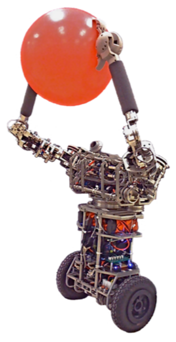
A model is selected automatically from the set of candidates using the Bayesian information criterion (BIC) (Schwarz 1978), BIC $= -2 \cdot \text{loglik} + p \cdot \log(N)$, where N is the number of data points and p is the number of model degrees of freedom. It can be shown that choosing the model with minimum BIC is equivalent to choosing the model with the largest approximate posterior probability (Hastie, Tibshirani, and Friedman 2009). It should be noted that the above combined approach to model learning and selection has the benefit of being completely parameter free.

Control

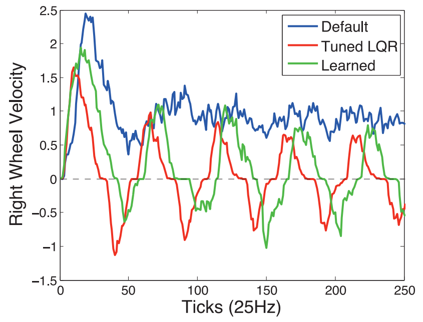
Learned dynamic models $\hat{f}(\mathbf{s}_t, \mathbf{a}_t)$ are used in constructing controllers that descend the gradient of objective functions. An objective function $\phi(\mathbf{s}_t)$ is a convex function that maps each state to a real number that describes the error in the system with respect to a primitive task. To select actions we derive a Jacobian matrix, $\mathbf{J}_t = \frac{\partial \hat{f}(\mathbf{s}_t, \mathbf{a}_t)}{\partial \mathbf{a}_t}$, and use its inverse to map directions of steepest descent of a given objective function to directions in action space (Nakamura 1991),

$$\Delta \mathbf{a}_t = -\eta \, \mathbf{J}_t^{\#} \frac{\partial \phi(\mathbf{s}_t)}{\partial \mathbf{s}_t}^{\top},$$

where $\mathbf{J}_t^{\#}$ is the pseudoinverse of $\mathbf{J}_t$ and η is a gain parameter. The key point is that the robot can choose to descend the gradient of $\phi(\mathbf{s}_t)$ using *any effectors* for which it has learned



(a)



(b)

Figure 1: (a) The uBot-5. (b) Learned arm controller performance vs. tuned wheel controller on ramp.

a model $\hat{f}(\mathbf{s}_t, \cdot)$, thus making this an attractive approach to doing whole-body control.[1]

Preliminary Results

As a first experiment in learning whole-body control strategies, we collected data from the uBot-5 (Figure 1a) in a balancing configuration with its two wheels controlled by a linear-quadratic regulator (LQR). During a 4 minute exploratory phase, the robot executed random actions in the shoulder joints and data were collected at 25 Hz. The changes in shoulder velocity affected a subset of the robot's state variables. For example, setting a positive velocity reference in the right shoulder would affect the position and angular velocity of the right shoulder, body tilt, and left and right wheels due to moments generated about the shoulder and wheels, changes in the mass distribution of the robot, and reactions of the unmodeled LQR controller.

By ignoring state variables that remain constant, the robot is left with a 10-dimensional continuous state space and a 2-dimensional continuous action space (one velocity reference for each shoulder joint). The basis functions included all degree 1 and 2 polynomials of the state and action variables, resulting in 90 total basis functions. We focused on the submodels that predict the left and right wheel velocities. The lasso algorithm and BIC selected 62 basis functions for the left wheel model and 59 for the right wheel model.

To test the suitability of the learned models for control, we used a simple quadratic error objective function which penalized wheel velocities deviating from 0. The controller was evaluated by placing the robot on a 10° inclined plane and comparing its behavior with and without the learned controller. In the latter case, the robot was placed at the top of ramp in the balancing configuration and it slowly drifted down the ramp due to gravitational acceleration eventually achieving a constant down-ramp velocity. However, when the learned controller is activated the robot quickly extends its arms out to counteract the acceleration due to gravity. The robot reaches a steady dynamic equilibrium after rolling backward approximately 25 cm.[2] Figure 1b shows the wheel velocities over time for the default and learned behaviors as well as a modified wheel LQR tuned to keep position on the ramp.

[1]Of course, this is not to say that all effectors will be equally well suited for meeting particular objectives.

[2]See video at: `http://www.cs.umass.edu/~scottk/videos`

Related and Future Work

Many impressive examples of control model learning can be found in the robotics literature, see e.g., (Schaal, Atkeson, and Vijayakumar 2002). However, these examples typically involve a robot learning a model for a particular task rather than demonstrating the generalizability of models for whole-body control. The locally-weighted projection regression algorithm (Vijayakumar and Schaal 2000) has many of the same benefits as the approach described here, but builds models with local, rather than global support, thus removing the need for basis functions. Making empirical comparisons between these different approaches with respect to whole-body control is a goal of future research.

We also intend to demonstrate the reuse of learned models across several tasks and investigate ways of bootstrapping using previously learned models. To scale up to high-dimensional action spaces, we anticipate the need to intelligently generate data rather than relying on random walks. Specifically, we are interested in how a developmental exploratory approach may facilitate learning in high-dimensional spaces while preserving basic safety guarantees (Grupen and Huber 2005).

Acknowledgments

Thanks to my advisors Rod Grupen and Andy Barto for their valuable feedback and guidance. Scott is supported by a NASA GSRP Fellowship from Johnson Space Center.

References

Efron, B.; Hastie, T.; Johnstone, L.; and Tibshirani, R. 2002. Least angle regression. *Annals of Statistics* 32:407–499.

Grupen, R. A., and Huber, M. 2005. A framework for the development of robot behavior. In *AAAI Spring Symposium Series: Developmental Robotics*.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *The Elements of Statistical Learning, 2nd Edition*. Springer.

Mahadevan, S.; Maggioni, M.; Ferguson, K.; and Osentoski, S. 2006. Learning representation and control in continuous markov decision processes. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*. AAAI Press.

Nakamura, Y. 1991. *Advanced Robotics: Redundancy and Optimization*. pub-AW:adr: Addison-Wesley.

Schaal, S.; Atkeson, C. G.; and Vijayakumar, S. 2002. Scalable techniques from nonparameteric statistics for real-time robot learning. *Applied Intelligence* 17(1):49–60.

Schwarz, G. E. 1978. Estimating the dimension of a model. *Annals of Statistics* 6:461–464.

Tibshirani, R. J. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B* 58(1):267–288.

Vijayakumar, S., and Schaal, S. 2000. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 1079–1086.