Improving Gradient Estimation by Incorporating Sensor Data

Gregory Donnell Lawrence
Stuart J. Russell

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-58
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-58.html

May 15, 2006

Improving Gradient Estimation by Incorporating Sensor Data

Gregory Lawrence
Stuart Russell

May 15, 2006

Abstract

A key step in many policy search algorithms is estimating the gradient of the objective function with respect to a given policy's parameters. The gradient is typically estimated from a set of policy trials. Each of these trials can be very expensive and so we prefer to minimize the total number of trials required to achieve a desired level of performance. In this paper we show that by viewing the task of estimating the gradient as a structured probabilistic inference problem, we can improve the learning performance. We argue that in many instances, reasoning about sensory data obtained during policy execution is beneficial. In other words, in addition to an agent knowing how well it performed during each policy run, it is helpful for it to learn "how it feels" to perform a particular task well. This knowledge is especially useful if we are able to incorporate prior knowledge specific to the given control task. Examples of using prior knowledge include setting the conditional independencies between various sensor variables and choosing the types of conditional probability distributions. In addition, by using hierarchical Bayes methods we are able to efficiently reuse old data from trials of other policies. We demonstrate the effectiveness of this approach by showing an improvement in learning performance on a toy cannon problem and a dart throwing task.

1 INTRODUCTION

Policy search algorithms have been very effective in learning good policies in the reinforcement learning setting. Successful applications include learning controllers for helicopter flight [9] and robot locomotion [5]. These algorithms improve a given policy by estimating the gradient of a policy's value with respect to its parameters from a collection of samples, where each sample is obtained by executing a policy. This gradient is used to improve the performance measure of the current policy by adjusting the parameters in the uphill direction. Given this new policy, the process is repeated by evaluating a new set of policies and taking another hill-climbing step. Each of these samples can be very expensive and so we prefer to

minimize the total number of samples required to achieve a desired level of performance. A number of approaches have been presented to help minimize the number of required samples [1, 2, 8, 11, 13] and we will show that additional improvements can be made by incorporating an agent's sensor data.

During each policy trial, an agent may receive lots of sensory data from its environment. While the agent's controller may use this information in deciding which actions to take, the sensory data is usually ignored in the gradient estimation task [1] and this can result in inefficient learning. As an example, suppose that an agent's goal is to aim a cannon so that when it is fired, the cannon ball hits a target in the distance. Furthermore, suppose that this agent increases the angle of the cannon and notices that the cannon missed too far. Normally, our agent will want to reduce the angle of the cannon so that the next shot moves closer to the target. However suppose that during the previous shot, the agent noticed that the shot sounded louder than usual. This helps to explain the miss and therefore the agent should keep the cannon at the same angle. Observing the sensor value helps to explain what actually happened and consequently, improves the quality of the gradient estimate. These types of inferences have been exploited when the agent is given a sensor model and perfect sensing [6]. In this paper, we remove these unrealistic assumptions and explain how we may construct algorithms that are implementable on real systems.

In addition to ignoring sensor data, many policy search algorithms throw away each batch of samples after taking a hill-climbing step. Importance sampling estimators have been applied to reuse old sample data [10, 11]. The key idea is that although the underlying dynamics of the environment may be unknown and highly nonlinear, we can still compute the probability that a given policy would have generated the same history observed by executing another. One drawback to this approach is that the advantages of reusing old samples may quickly fall off as the current policy moves far away from the generating policy, even if the value is completely linear in the policy parameters. This occurs when the probability of a given policy generating the history of another is close to zero and therefore provides neglible weight in the importance sampling equation.

Our approach will be to view the task of estimating the gradient as a structured probabilistic inference problem. For each task we will present a Bayesian network parameterized by θ that represents the process of generating the observable data, including a measurement of the performance. Each hill-climbing step will be made by first evaluating N policies drawn according to some exploration strategy in order to learn the network parameters θ that best explain the observed data. Given this parameter, we can compute the gradient of the objective function and use it as our estimate. Finally, we adjust the policy in the direction of the gradient in an attempt to increase the objective. We parameterize our Bayesian network so that the probabilistic relationships that occur between the observable data can differ between policies evaluated during different hill-climbing steps. However, we enforce parameter tying by placing a prior distribution over these parameter values. A kernel function measures how

[1] There are other approaches to reinforcement learning, such as temporal-difference learning, that make use of this sensory information by learning a value function for each possible state. However, applying these algorithms can be very difficult in some domains because it may be hard to determine the true state space. Furthermore, most real problems are only partially observable.

"similar" samples obtained during different hill-climbing steps are and is used to construct an appropriate prior. One advantage of this approach is that different conditional probability distributions in the Bayesian network can be shared at various strengths.

Section 2 of this paper shows how one may use a Bayesian network to represent the probability distribution of the sensor variables and performance measure. Section 3 describes a hierarchical Bayes approach to reuse samples. Section 4 describes how to learn the parameters of the Bayesian network so that we may estimate the gradient (the details are presented in the appendix sections). We will show the effectiveness of this approach by examining a toy cannon problem and a dart throwing problem in Section 5 and discuss possibilities for future improvements in Section 6.

2 INCORPORATING SENSOR INFORMATION

A policy π determines how an agent chooses its actions given its past observations, and the reinforcement learning goal is to find a policy π^* that maximizes the objective function. In this paper we use policy search by hill-climbing through a space of parameterized policies $\pi \in \mathbb{R}^d$. We adopt the standard objective function which is that an agent should maximize the expected sum of future reward values. Each policy execution gives a *history* h, whose value is the sequence of observation-action pairs. The *response function* $F(h)$ evaluates each history and equals the sum of reward values obtained at each time step. Thus the optimal policy maximizes the expected response $\mathrm{E}[F(H)|\pi^*]$ where the histories H are generated from π^*.

We will use subscripts to denote the ith policy execution and superscripts to denote the jth hill-climbing step. Thus in a single run of the algorithm, π_i^j represents the ith policy executed in the jth hill-climbing step.[2] In addition, we write π_0^j to denote the nominal policy used in the jth hill-climbing step (the actual policies that are evaluated will be drawn according to some exploration strategy).

2.1 TOY EXAMPLE

To illustrate the main contributions of this paper we will examine a toy cannon problem (Figure 1). In this problem the goal is to fire a cannon ball towards a distant target. The policy $\pi = (\theta_d, v_d)^T$ consists of a *desired* cannon angle, $0 \leq \theta_d \leq \pi/2$, and *desired* initial velocity, $v_d > 0$. In this single-step problem, the policy is perturbed by noise to give the *actual controls* $x = (\theta_a, v_a)^T$. We assume that the agent has access to a noisy sensor that measures x and let $s = (\theta_s, v_s)$ denote its value. There is additive, zero-mean noise in both the control and sensor values; the control noise has covariance matrix Σ_x and the sensor noise has covariance matrix Σ_s. The history h for this problem contains both the desired action π and the sensor value s. The response function is defined to be $F(h) = -d(h)^2$, where $d(h)$

[2]To ease explanation we assume that, at each hill climbing step, exactly N policies are evaluated and that we are currently in the Mth step of the hill-climbing procedure. It is straightforward to relax this assumption.

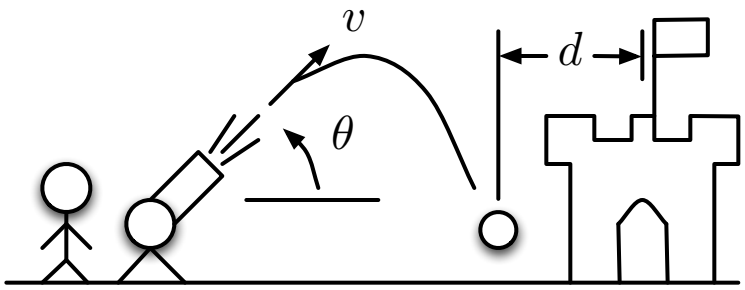

Figure 1: The cannon problem.

is the distance from the target to where the cannon ball lands. Maximizing $\mathrm{E}[F(H)|\pi]$ is equivalent to minimizing the expected squared distance error.

2.2 BAYESIAN NETWORK REPRESENTATION

Figure 2a shows a Bayesian network that represents a single trial in the cannon problem. The joint distribution of a trial is written as $P(\pi_e, x, f, s) = P(\pi_e)P(x|\pi_e)P(f|x)P(s|x)$. An exploration policy π_e is drawn from a neighborhood centered around the nominal policy for the current hill-climbing step. Given this policy, the hidden random variable X represents the actual control, F denotes the response value, and S denotes the sensor value. We assume that the agent is given a suitable network structure and an appropriate parameterization of the conditional probability distributions. For the cannon problem, the agent assumes that each node, except for π_e [3], is drawn from a linear-Gaussian distribution conditioned on the values of its parents. If we assume that for each hill-climbing step the policies are drawn from a small neighborhood and that the noise is relatively small, then the linear-Gaussian assumption is accurate. We will assume that the learning algorithm does not know that there is additive, zero-mean noise and that it does not know the covariance matrices in advance. If we were to include these constraints, then the learning performance will improve (e.g., section 5 shows an improvement in learning performance in the case of a known sensor model).

The advantage of incorporating the sensor variables is in explaining the variance of the response F. If the agent had access to the hidden variable X, then it could better predict what the response should be for a particular trial. In general, the agent will never be able to determine this exactly but some information can be inferred from the sensors. Learning the relationship between what an agent believes about the actual control X and the response F is relatively easy, compared to learning the relationship between π_e and the response F, because the noise has already been partially accounted for.

Let $\theta = \{A_x, b_x, \Sigma_x, A_f, b_f, \Sigma_f, A_s, b_s, \Sigma_s\}$ represent all of the parameters in the Bayesian

[3]Since the agent always knows the policies it chooses to execute, we do not need to learn the distribution of π_e.

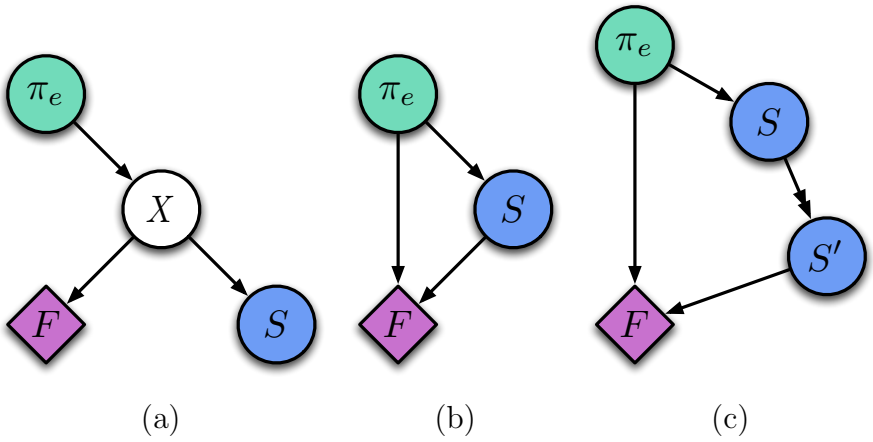

Figure 2: (a) A Bayesian network that represents the process of executing a policy in the cannon problem in which the hidden random variable X represents the actual control, F denotes the response value, and S denotes the sensor value, (b) an alternative network without any latent variables, and (c) a network that includes quadratic sensor terms $S' = (\theta_s, v_s, \theta_s^2, v_s^2, \theta_s v_s)^T$.

network.[4] The conditional distribution of a node is given by its corresponding parameters (denoted by subscripts). For example, the mean of random variable X is given by the following linear relationship $\mu_x = A_x \pi_e + b_x$ and the variance is given by Σ_x. Given that we have a suitable network structure for a given problem, we would like to find a parameter θ^* that best captures the probabilistic relationships among a policy, the sensory information obtained during policy execution, and the response value. We can learn θ^* using standard methods (e.g., maximum likelihood estimation). It may be easier to learn the model parameters in a Bayesian network that is free of latent variables (Figure 2b). In this case, the distributions captured by this alternative representation form a superset of those captured in Figure 2a, where the latent variable constrains the response and sensor nodes to be drawn from a distribution conditioned on a two-dimensional intermediate value.[5] Given θ^*, we can compute the gradient of the objective function with respect to changes in the policy node. For the Bayesian network given in Figure 2a the gradient of the objective is written as $\nabla_\pi \mathrm{E}[F(H)|\pi] = (A_f A_x)^T$. A basic algorithm is to learn θ^* from the data obtained during the current hill-climbing step and follow the gradient provided by the network parameters θ^*.

Prior knowledge can be used to improved the hill-climbing learning performance. For example, we may already have access to an accurate sensor model, thereby reducing the the number of model parameters that need to be learned. An alternative approach is to place

[4]This parameterization is more general than necessary, according to our problem description, because it does not exploit certain facts about the cannon problem (e.g., the noise has zero-mean).

[5]By using the network structure presented in Figure 2b we lose some of the structure inherent in the problem while making the inference task easier.

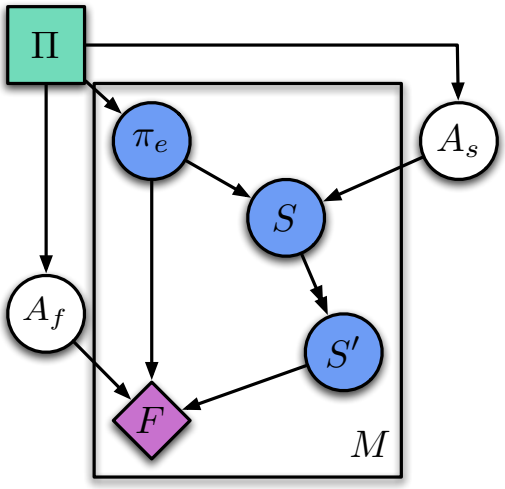

Figure 3: A hierarchical Bayesian network based on Figure 2c that represents the process of executing policies in the cannon problem. Information is shared between the different steps via the linear regression coefficients contained within nodes A_s and A_f.

a prior probability distribution on the model parameters using knowledge about the specific task. Another type of prior knowledge is structural. Suppose that we know that the noise added to the desired angle θ_d is independent of the noise in velocity v_d. This can be enforced by constraining the covariance matrices to be diagonal. We can also use prior knowledge in choosing the kind of conditional probability distributions. For example, in this toy problem, the mapping between sensor values and the response function (Figure 2b) is better captured by a quadratic function of the sensor variable. We may add quadratic sensor terms to the parents of the response node F by adding a deterministic mapping between the observed sensor value and a node $S' = (\theta_s, v_s, \theta_s^2, v_s^2, \theta_s v_s)^T$.

In Section 5 we show improvements in the learning performance of an agent that incorporates its sensor data. Adding the quadratic function of the sensor variables provides additional improvements and so we will use the Bayesian network shown in Figure 2c from now on.

3 SAMPLE REUSE

Many policy search algorithms throw away old sample data after taking each hill-climbing step. However, if the data was obtained from a policy in a different hill-climbing step that is similar (according to some metric) to the current policy, the data may still be useful in reasoning about the current policy. Our approach to incorporating sample data from prior hill-climbing steps will be to assume that the regression coefficients $\{A_s, b_s, A_f, b_f\}$ that are used to parameterize Figure 2c are tied via a hierarchical Bayes model. Figure 3 shows a

Bayesian network for the cannon problem where we use plate notation to show that the nodes contained within the plate are to be copied M times. Decision node $\Pi := \{\pi_0^1, \pi_0^2, \ldots, \pi_0^M\}$ contains all of the nominal policies used in the M hill-climbing steps and is used to determine the level of parameter tying.

Each hill-climbing step has its own regression coefficients, but some of these coefficients are assumed to be drawn from a joint Gaussian distribution. The covariance of the Gaussian determines the strength of the parameter tying. Coefficients that are constrained to be equal will have correlations equal to 1 and those that are independent will have zero correlation. We consider correlation values that are between these two extremes to give varying levels of soft parameter tying. These correlations cause the parameter learning to favor solutions that are similar across the different hill-climbing steps.

The regression coefficients are now considered random and are part of the Bayesian network where we group the offset terms $\{b_s, b_f\}$ with their corresponding linear terms $\{A_s, A_f\}$. As an example, the A_s node in Figure 3 contains all of the linear regression coefficients (including the offset terms) used to predict the sensor value given the exploration policy π_e concatenated together and it is written as follows:

$$A_s \;:=\; \left[\; b_s^1 \quad {A_s}^1 \quad b_s^2 \quad {A_s}^2 \quad \cdots \quad b_s^M \quad {A_s}^M \;\right].$$

In general, the regression coefficients are contained within a single $c \times m$ matrix where c is the dimension of the child node and $m = M(r+1)$ where r is the sum of the dimensions of its parents.

The strength of the parameter tying depends on the nominal policies used for each hill-climbing step and a set of hyper-parameters determines this relationship. For a particular Π we have a prior belief on what the regression coefficients for each hill-climbing step should be before we obtain any data. For example, if Π contains two nominal policies that are exactly the same, then we should believe that the corresponding coefficients must be equal. Likewise, if Π contains two nominal policies that are far away from each other then we may believe that the corresponding coefficients are completely uncorrelated.

We use the matrix normal distribution to represent the prior and perform Bayesian linear regression [7] for the shared nodes. The density of a matrix A drawn from the matrix normal distribution is writen as

$$\begin{aligned} A &\sim \mathcal{N}_{c\times m}(0, V, K) \\ P(A) &= (2\pi)^{-\frac{1}{2}cm} |V|^{-\frac{m}{2}} |K|^{-\frac{c}{2}} \\ &\quad \exp\left\{ -\frac{1}{2} \mathrm{tr}(V^{-1} A K^{-1} A^T) \right\}, \end{aligned}$$

where V and K are two covariance matrices.

Each node contained within the plate can have regression coefficients drawn from different prior probabilities which allows us to share different parts of the model at various strength levels. The Bayesian network in Figure 3 has two sets of regression coefficients where one set determines the linear relationship between the exploration policy π_e and the observed

sensor value s and the other determines how the exploration policy π_e and quadratic sensor value s' relates to the response f. In this problem, the true sensor mapping $P(s|\pi_e)$ is given by a linear Gaussian distribution which allows us to learn its parameters using samples from any hill-climbing step (i.e., these parameters can be fully tied). However, the true response function mapping $P(f|\pi_e, s)$ is only locally linear (Figure 2b) or locally quadratic (Figure 2c) and therefore the parameter tying strength should depend on the corresponding policies.

The covariance matrix K determines the strength of the parameter tying. The value of this matrix is determined using a similarity metric defined between the M hill-climbing steps. We use the following weights to determine the similarity between hill-climbing steps i and j:

$$w_{ij} \;:=\; \exp\left(-\lambda(\pi_0^i - \pi_0^j)^T S(\pi_0^i - \pi_0^j)\right) + \epsilon\delta_{ij},$$

where λ is a scaling term, S is a known matrix that scales each dimension, and ϵ is a small constant added to the diagonal to ensure a non-singular weight matrix. Given these weights we consider covariance matrices of the following form:

$$K \;:=\; \beta \begin{bmatrix} C^{-1}w_{11} & \dots & C^{-1}w_{1M} \\ \vdots & \ddots & \vdots \\ C^{-1}w_{M1} & \dots & C^{-1}w_{MM} \end{bmatrix},$$

where $C := \frac{1}{MN} XX^T$ in order to make parameter learning invariant to changes in scale and rotation and β is a scaling term. The matrix X contains all of a node's parent values stored as a column vector and replicated for each sample. The covariance matrix K determines the relationship between the different columns of A and therefore gives the correlations between the same regression coefficients in different hill-climbing steps. If two sample batches are similar according to the weight matrix, then the corresponding regression coefficients will be highly correlated. The covariance matrix V determines the relationship between the different rows of A and we place no restrictions on its value.

4 PARAMETER ESTIMATION

We will generally not know the parameters θ of the Bayesian network that best fit the observed data and so we must learn them from experience. The parameterization of covariance matrix K prevents us from finding an analytical solution to this problem. We use empirical Bayes to find an appropriate covariance matrix K by searching over a set of hyper-parameters to best explain the observed data. Then, given these values, we compute the remaining elements of θ that maximize the a posteriori estimate of the regression coefficient matrices. The details of this process are described in the appendix. Finally, we can extract the gradient of the response with respect to the current policy $\nabla_{\pi_0^M} \mathrm{E}[F(H)|\pi_0^M]$ where the histories are drawn according to the network. Given the network parameters θ^* we can either compute the gradient analytically or estimate it by drawing samples from the network. Depending on the structure of the network it may be difficult to find an analytical solution to the gradient

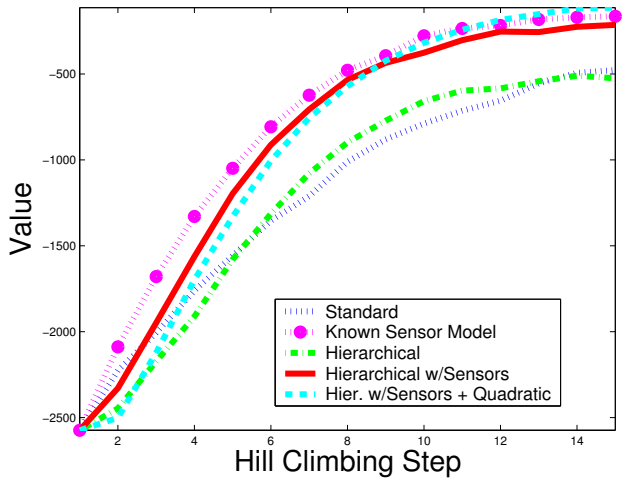

Figure 4: The learning curve performance of five different policy search strategies in the cannon problem. The five curves include a standard method, one that uses the true sensor model, one that uses hierarchical Bayes without any sensor values, one that uses hierarchical Bayes with sensor variables, and one that adds the quadratic terms. The learning curves show that including sensory information and quadratic terms are both useful.

(e.g., we may have added non-linear nodes). Instead, we just draw samples in a top-down fashion.[6]

5 RESULTS

We ran policy search on the toy cannon problem by learning the Bayesian network parameters in a manner described by the previous section and then estimating the gradient of the Bayesian network. The policy parameters were adjusted in the direction of the gradient and the magnitude of this change was bounded by a constant. At each hill-climbing step we drew a single sample from 10 different policies and we averaged over 100 hill-climbing runs. The policies were drawn, according to our exploration strategy, from a Gaussian distribution centered around the nominal policy. Figure 4 shows the learning curve performance of five different policy search strategies. The five curves include a standard method, one in which the sensor model is known, one that uses hierarchical Bayes without any sensor variables, one that uses hierarchical Bayes with sensor variables, and one that adds the quadratic terms. The standard method includes using a least squares fit between the policy π and the response function, ignoring any sensor information. The learning curves show that sensory information and quadratic terms are both useful. Note that applying hierarchical Bayes alone does not help as much as it does if we incorporate the sensor data. The case in which the agent is given the sensor model appears to perform the best.

[6]We assume that the cost of obtaining these samples is negligible when compared to executing a policy.

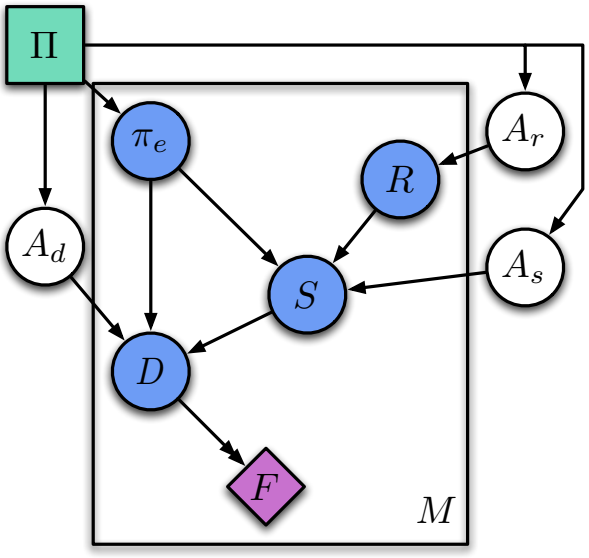

Figure 5: A hierarchical Bayesian network that represents the process of executing policies in the dart thrower problem. The dart release time R is independent of the exploration policy π_e. The sensor node S measures the state of the arm $s \in \mathbb{R}^6$ right before the dart is released. The distance node D measures the distance that the dart lands from the bulls eye and the response function deterministically squares this value to give the appropriate response.

We also applied this technique to a dart throwing task [6]. The objective is to throw a dart with minimal mean squared error (measured from where the dart hits the wall to the center of the dart board). The arm is modelled as a three-link rigid body with dimensions based on biological measurements [3]. The links correspond to the upper arm, forearm, and hand and are connected using a single degree of freedom rotational joint. The upper arm is connected to the shoulder at a fixed location. We generated code to simulate the dynamics of this system using the Open Dynamics Engine.

The arm is controlled by applying torques at each joint. These torques are generated by a PD-controller that attempts to move the arm through a desired trajectory, specified by a cubic spline for each joint angle. The starting posture of the arm is fixed in advanced and the path is determined by interpolating between three other knot positions. These three knots per joint give us a compact policy representation of 9 parameters. The controller is simulated for approximately 0.2 seconds and then the dart is released (there is Gaussian noise added to the release time with $\sigma = 0.01$). Additional noise enters the system by perturbing the torques given by the PD-controller by additive and multiplicative noise. Multiplicative noise has been shown to explain biological motion [4, 12].

Figure 5 shows the Bayesian network used in the dart throwing task. The sensor node S measures the state of the arm $s \in \mathbb{R}^6$ right before the dart is released and its value is dependent on the given exploration policy π_e and the release time of the dart. Since the

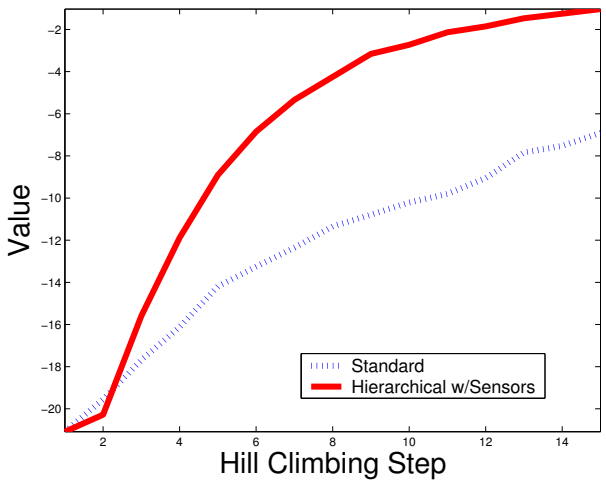

Figure 6: The learning curve performance of two different policy search strategies in the dart throwing domain. One curve shows the performance of a standard method and the other shows the performance when using the Bayesian network shown in figure 5. Incorporating sensor data gives a substantial improvement in the learning performance when compared to the standard method.

noise in the release time is independent of the policy we can exploit this in the learning process (i.e., the release node R has no parents other than its regression coefficients). The distance node D measures the distance that the dart lands from the bulls eye and the response function deterministically squares this value to give the appropriate response. Figure 6 shows a substantial improvement in the learning performance of an algorithm that incorporates sensor data when compared to the standard method. At each hill-climbing step we drew a single sample from 20 different policies and we averaged over 100 hill-climbing runs. Like the cannon problem, these policies were drawn from a Gaussian distribution centered around the nominal policy. The standard method includes using a least squares fit between the policy π and the response function, ignoring any sensor information.

6 DISCUSSION

We demonstrated how one may incorporate sensor data into the gradient estimation task to improve the performance of policy search. For the learning problems considered in this paper, we presented a Bayesian network that represents the probabilistic relationships between the executed policy, the sensor variables, and the corresponding response. Hierarchical Bayes methods were used to tie parameters across different hill-climbing steps at various levels of strength. We presented learning curves that show improvements in the learning performance of a toy cannon problem and dart throwing task. We feel that additional improvements can be made in the learning performance by incorporating different kinds of prior knowledge into the learning task.

This paper considers exploiting the conditional independencies present in each learning task. In addition, we showed that the gradient estimates could be improved by using the right type of conditional probability distribution for each variable (e.g., a quadratic versus linear relationship). One area for improvement involves incorporating our knowledge of the physics behind each task. For example, in the cannon problem we already know the equations of projectile motion. Thus, given the actual controls, we should be able to accurately predict the response. Even in cases in which we do not know the equations of motion, we often know qualitative information about the motion, such as increasing the desired velocity causes the cannon ball to fly further (i.e., the distance travelled is monotonically increasing as a function of the desired velocity). One possible approach to incorporating this information is to place constraints on the signs of the regression coefficients.

References

[1] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[2] Jonathan Baxter E. Greensmith, Peter L. Bartlett. Variance reduction techniques for gradient estimates in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1507–1514, 2001.

[3] B. Garner and M. Pandy. A kinematic model of the upper limb based on the visible human project (vhp) image dataset. *Computer Methods in Biomechanics and Biomedical Engineering*, 2:107–124, 1999.

[4] Christopher Harris and Daniel Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394:780–784, 1998.

[5] N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 2004.

[6] Gregory Lawrence, Noah Cowan, and Stuart Russell. Efficient gradient estimation for motor control learning. In *Proceedings of the Nineteenth International Conference on Uncertainty in Artificial Intelligence*, 2003.

[7] Thomas Minka. Bayesian linear regression. Technical report, MIT, 2000.

[8] Andrew Y. Ng and Michael Jordan. Pegasus: A policy search method for large mdps and pomdps. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

[9] Andrew Y. Ng, H. Jin Kim, Michael Jordan, and Shankar Sastry. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems*, 2003.

[10] Leon Peshkin and Christian R. Shelton. Learning from scarce experience. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

[11] Christian R. Shelton. Policy improvement for POMDPs using normalized importance sampling. In *Proceedings of the Seventeenth International Conference on Uncertainty in Artificial Intelligence*, pages 496–503, 2001.

[12] Emanuel Todorov and Michael I. Jordan. Optimal control as a theory of motor coordination. *Nature Neuroscience*, 2002. submitted.

[13] Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 538–545, 2001.

A EMPIRICAL BAYES

We choose to consider only Bayesian networks in which all of the nodes within the plate are observed. As a result, the task of finding the best hyper-parameters for each set of regression coefficients can be done independently. The covariance matrix for a particular set of regression coefficients is parameterized by two hyper-parameters: λ and β. We use empirical Bayes to fix these values so that the likelihood of seeing the observed data is maximized. The log-likelihood of observing the data given a particular covariance matrix K is written as follows:

$$\begin{aligned}\ell(\lambda,\beta) &= k - \frac{MN}{2}\log|V| - \frac{c}{2}\log|\bar{X}^T K\bar{X} + I| - \\ &\quad \frac{1}{2}\mathrm{tr}\left(V^{-1}Y(\bar{X}^T K\bar{X} + I)^{-1}Y^T\right),\end{aligned}$$

where k is a constant, Y is a $c \times m$ matrix containing the observed child node values, and $\bar{X}$ is a specially formatted matrix containing the observed parent values. $\bar{X}$ is written as

$$\bar{X} = \begin{bmatrix} X^1 & 0 & 0 & 0 \\ 0 & X^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & X^M \end{bmatrix}$$

to ensure that the linear relationship ($Y = A\bar{X}$) for the data of each hill-climbing step is given by the correct entries in the A matrix.

For a given covariance matrix K, the matrix V^* that maximizes the log likelihood is given by

$$V^* = \frac{1}{N}Y(\bar{X}^T K\bar{X} + I)^{-1}Y^T.$$

We search over the hyperparameters (λ, β) using hill-climbing to find a local maximum of the log likelihood function. These values are then fixed while computing the maximum a posteriori estimates of the regression coefficient matrices.

B MAXIMUM A POSTERIORI ESTIMATION

Given an appropriate setting of the hyper-parameters, we can find the maximum a posteriori estimate for the regression coefficients. The log-likelihood of observing the data drawn from A can be written as follows:

$$\begin{aligned} \ell(\lambda, \beta, A) &= c - \frac{m}{2}\log|V| - \frac{d}{2}\log|K| - \\ &\quad \frac{1}{2}\mathrm{tr}(V^{-1}AK^{-1}A^T) - \frac{MN}{2}\log|V| - \\ &\quad \frac{1}{2}\mathrm{tr}\left((Y - A\bar{X})^T V^{-1}(Y - A\bar{X})\right). \end{aligned}$$

Maximizing the above expression with respect to matrix A gives us the maximum a posteriori estimate as follows:

$$A^* = Y\bar{X}^T(K^{-1} + \bar{X}\bar{X}^T)^{-1}.$$

This estimate is similar to the least squares fit except the addition of a penalty term K^{-1} that favors solutions that are similar across different hill-climbing steps. There is also a general shrinkage towards the origin as found in ridge regression.