Partial, Noisy and Qualitative Models for Adaptive Critic Based Neurocontrol

Thaddeus T. Shannon,
Portland State University, Systems Science Ph.D. Program
tads@sysc.pdx.edu

Abstract

The roles of plant models in adaptive critic methods for approximate dynamic programming are considered, with primary focus given to the DHP methodology. In place of complete system identification, partial, approximate, and qualitative models of plant dynamics are considered. Such models are found to be sufficient for successful controller design. As classification is in general easier than regression, the results for qualitative models suggest an avenue for simplifying ongoing system identification in adaptive control applications.

Introduction

A variety of Adaptive Critic Design techniques for training neuro-controllers have appeared in the literature in recent years [7], [8], [9], [12] and [13]. These techniques can be divided into model based methods such as Dual Heuristic Programming (DHP), and non-model based methods such as Action Dependent Heuristic Dynamic Programming (ADHDP) or Q-learning. While the DHP method has been shown to be much more efficient for training neuro-controllers and to produce superior designs to the non-model based methods, its implementation relies on having an explicit differentiable model. When such a model is not available, the cost of developing one may well offset the increased speed and accuracy of the model based methods.

An alternative to the less model intensive methods is to use simplified models. The time and effort needed to develop a model may be reduced either by only estimating a partial model, by using a very rough estimation procedure, or by only attempting to capture the qualitative behavior of the system with the model. We seek to better define what is required in a plant model for successful implementation of DHP based controller training. The eventual aim of this exercise is to compare the use of various approximate models in the DHP method, with the use of other adaptive critic methods. Towards this end, this paper outlines some empirical results from our initial investigations.

The Role of Models in Approximate Dynamic Programming

Adaptive critic methods of approximate dynamic programming are all based upon using a "critic" network to provide feedback for training a "controller" network. The standard classification of adaptive critic methods is based on the critic's inputs and outputs. In Heuristic Dynamic Programming (HDP) the critic outputs estimates of the secondary utility function's value. In Dual Heuristic Programming (DHP) the critic outputs estimates of the derivatives of the secondary utility function. In either case, action dependent critics receive both the system states and the controller's action (control) as inputs (thus ADHDP and ADDHP).

The two networks used in these approaches to approximate dynamic programming utilize two distinct training loops, a controller training loop and a critic training loop [5]. The controller training loop adapts a neural network to be an approximately optimal controller. Specifically, the controller network is trained to maximize some secondary utility function J(t). The network outputs control actions u(t), so a gradient based learning algorithm requires estimates of the derivatives $\frac{\partial J(t)}{\partial u_i(t)}$ for network training. The critic network is trained based on the consistency of its estimates through time, the exact implicit relationship being a function of the type of critic used and the structure of the primary utility function.

An alternative way of distinguishing these methods is to consider the role of system models in the training loops of each method:

HDP: The critic estimates J(t) based on the system state R(t). Critic training is based on the identity

$$J(t) = U(t) + g\ J(t+1),$$

which requires no system model to check. Controller training is based on finding the derivatives of J(t) with respect to the control variables. In HDP we obtain these derivatives through the chain rule $\frac{\partial J(t)}{\partial u_i(t)} = \sum_{j=1}^{n} \frac{\partial J(t)}{\partial R_j(t)} \frac{\partial R_j(t)}{\partial u_i(t)}$. We combine estimates of the derivatives of J(t) with respect to the states , obtained via backpropagation through the critic network, with the derivatives of the states with respect to the controls using the chain rule. This final set of derivatives comes from a differentiable model, e.g. an explicit analytic model or a neural model. Thus **HDP uses a model for controller training but not critic training.**

ADHDP: (Q-learning) Training for the critic network is the same as for HDP. Training for the controller is simplified in that the control variables are inputs to the critic, thus derivatives of J(t) with respect to the controls are obtained directly from backpropagation through the critic. Thus **ADHDP uses no models in the training process.**

DHP: Here the critic estimates the derivatives of J(t) with respect to the system states, i.e. $l_i(t) = \frac{\partial J(t)}{\partial R_i(t)}$. The identity used for critic training is (in tensor notation)

$$l_i(t) = \frac{\partial U(t)}{\partial R_i(t)} + \frac{\partial U(t)}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial R_i(t)} + l_k(t+1)\left[\frac{\partial R_k(t+1)}{\partial R_i(t)} + \frac{\partial R_k(t+1)}{\partial u_m(t)} \frac{\partial u_m(t)}{\partial R_i(t)} \right].$$

To evaluate the right hand side of this equation we need a full model of the system dynamics. This includes all the terms from the Jacobian matrix of the coupled plant-controller system, e.g. $\frac{\partial R_j(t+1)}{\partial R_i(t)}$ and $\frac{\partial R_j(t+1)}{\partial u_i(t)}$.

Controller training is much like in HDP, except that the controller training loop directly utilizes the critic outputs along with the system model. So **DHP uses models for both critic and controller training.**

ADDHP: This methods utilizes the DHP critic training process, but gets the derivatives needed for controller training directly from the critic's output. Therefore **ADDHP uses a model for critic training but not for controller training.**

One of the promising applications for adaptive critic methodologies is in adaptive control contexts for non-stationary plants. In these contexts there will necessarily be a third training loop updating a differentiable model in an ongoing plant identification process. The limitations of the overall adaptation context may often come from our ability to track the changing system with our model, rather than from our ability to continuously solve the approximate dynamic programming problem. Thus understanding the effects of model error on controller training is important.

Benchmark Problems

Two benchmark problems are considered in our preliminary work on the role of model quality in the DHP training process: the inverted pendulum or pole-cart problem, and a nonlinear multivariate discrete time plant proposed by Narendra and Mukhopadhyay [6]. Both of these plants have been extensively explored using adaptive critic methodologies [11], [7].

The Pole-Cart System

The pole-cart problem is described by:

$$\ddot{q} = \left[\cos q \left[\frac{-u - ml\dot{q}^2 \sin q + m_c \operatorname{sgn}(\dot{x})}{m_c + m} \right] + g \sin q - \frac{m_p \dot{q}}{ml} \right] \left[\frac{4l}{3} - \frac{ml \cos^2 q}{m_c + m} \right]^{-1},$$

$$\ddot{x} = \frac{u + ml\left[\dot{q}^2 \sin q - \ddot{q} \cos q\right] - m_c \operatorname{sgn}(\dot{x})}{m_c + m},$$

where q is the angle of the pole's deflection from vertical, x is the position of the cart on the track, m is the mass of the pole, l is the length of the pole, g is the gravitational constant, m_c is the mass of the cart, and m_c and m_p are the coefficients of friction of the cart on the track and the pole on the cart. The objective is to minimize the pole's angle of deflection from vertical (q = 0), while keeping the cart as close as possible to a pre-specified location on the track (x = 0). In our example here we use the primary utility function

$$U(t) = -0.25\left(q^2(t) + x^2(t)\right).$$

Our baseline solution consists of a two layer controller network with 6 inputs (position, velocity and acceleration for both cart position and pole angle), 1 hidden layer element, and 1 output layer element. Both processing elements use hyperbolic tangent activation functions and contain no bias term. The critic network uses the same 6 inputs, has 1 hidden layer element and 6 outputs. Again the processing elements have no bias term, but while the hidden element uses a hyperbolic tangent function, the output elements are linear.

Training of the networks is conducted simultaneously using a plain vanilla gradient descent method with moderate sized learning rates and no momentum terms. In this case training stimulus is provided by setting the pole at small deflections away from vertical. The resulting controllers vary in performance, but are uniformly able to both keep the pole vertical and control the position of the cart on the track. As was reported in [4], initial angles of over 50° can be handled, and the cart made to track desired locations. For prior treatments of this problem see [1], [10], [2], [3] and [4].

The Narendra System

The Narendra system is defined by the state equations:

$$x_1(t+1) = 0.9x_1(t)\sin[x_2(t)] + \left[2 + 1.5\frac{x_1(t)u_1(t)}{1+x_1^2(t)u_1^2(t)}\right]u_1(t) + \left[x_1(t) + \frac{2x_1(t)}{1+x_1^2(t)}\right]u_2(t),$$

$$x_2(t+1) = x_3(t)\left[1+\sin[4x_3(t)]\right] + \frac{x_3(t)}{1+x_3^2(t)},$$

$$x_3(t+1) = \left[3+\sin[2x_1(t)]\right]u_2(t).$$

Often the observable states are taken to be $x_1(t)$ and $x_2(t)$. The plant is stable at the origin with constant control values. Linearized around the origin, it is controllable, observable and of minimum phase. The standard reference signal for evaluating controller performance is:

$$\tilde{x}_1(t) = 0.75\sin\left[\frac{2p\,t}{50}\right] + 0.75\sin\left[\frac{2p\,t}{10}\right],$$

$$\tilde{x}_2(t) = 0.75\sin\left[\frac{2p\,t}{30}\right] + 0.75\sin\left[\frac{2p\,t}{20}\right].$$

Utilizing the reference signal, and recognizing the time delays in the system, we use the primary utility function

$$U(t) = \left(x_1(t) - \tilde{x}_1(t)\right)^2 + \left(x_2(t+1) - \tilde{x}_2(t+1)\right)^2.$$

Our basic controller has 5 inputs, $x_1(t)$, $x_2(t)$, $x_3(t)$, $\tilde{x}_1(t+1)$, and $\tilde{x}_2(t+2)$, 6 hidden layer elements and 2 outputs, $u_1(t)$ and $u_2(t)$. The critic network has 4 inputs, $x_1(t)$, $x_2(t)$, $\tilde{x}_1(t)$, and $\tilde{x}_2(t)$, 6 hidden layer elements, and 2 outputs, $\frac{\partial J(t)}{\partial x_1(t)}$ and $\frac{\partial J(t)}{x_2(t)}$. All the processing elements in both networks use hyperbolic tangent activation functions and have bias terms. Training of both networks is performed simultaneously using the same generic method employed for the pole-cart problem, the only difference being the size of the learning rates and the inclusion of a small momentum term in this case.

Baseline training is carried out using a random reference signal generated by selecting a value from the interval [-1.5, 1.5] via a uniform distribution every four time steps. This procedure generates a random, stair-step signal that provides persistent excitation for training. The performance of the trained controller is evaluated after 40,000 training steps using the sinusoidal reference trajectory given above. This evaluation is a generalization test as the controller never sees a non-random reference trajectory during the training process. Controllers trained by this method vary in performance, with an average RMS error of 0.30 and with the better controllers producing an RMS error of about 0.23.

Partial Models

By a partial model we mean a model which only explains some of the causal interactions between state variables. A general hypothesis is that a differentiable model that includes any subset of the state variables that would constitute an observable system should be sufficient for DHP training. Such a *model* should be useable for carrying out the DHP process, even in those situations where additional plant states might be observable, and even when such additional state variables are used as inputs to the controller and critic networks.

In the case of the pole-cart problem a variety of partial models can be successfully used. Successful controller training can be carried out with any single state or pair of states left out of the model. In fact, any model that includes at least one angle state variable will work. Note that these statements are based on using all state variables as controller and critic network inputs; the state variables are only omitted from the model used in the training update process.

For Narendra's problem, a model including only the first and second state variables (still an observable system) works as well as the full model. As with the pole-cart problem, this test is again intermediate between using full state information with a complete model, and using the reduced model with a controller and critic receiving only the first two states. Controller performance was almost identical to that obtained using the full analytic model with an average RMS error of 0.30.

Noisy Models

Noisy models are ubiquitous in real life. A model based on regression analysis or any other statistical estimation technique is approximate at every operating point. The numerical values derived from such models may be considered "noisy" values in that they are (hopefully) close to the "true" values and if properly estimated, randomly distributed around the true values. We can perform controlled experiments to yield these kinds of approximations by mixing noise with the "true" values derived from a known analytic model. Two kinds of mixing are possible, additive mixing, and multiplicative mixing.

Most of our experiments have been conducted with models unbiased from the true analytic models used to simulate the plants. The exception to this has been a few experiments with uniformly biased models for the Narendra plant mentioned below.

Results for the pole-cart problem show that relatively large amounts of multiplicative noise or scaling can be introduced into a model and still obtain good results from the DHP method. On the other hand, relatively small amounts of additive noise in the model can cause the DHP method to fail. It should be remembered that both these statements are based on introducing noise into analytic models, hence the resulting noisy models are unbiased. Obviously if a model was biased the situation is even worse.

Results for Narendra's problem are similar, though in this case there is a much greater tolerance to additive noise than in the cart-pole example. Controller performance ranged from 0.37 RMS error for unbiased scaling factors in the interval (0, 2), to 0.53 RMS error for additive noise from the interval (-1, 1). Additionally, even with the incorporation of a consistent positive or negative bias away from the true derivative values, DHP training is still possible. This result for biased models is based on adding non-zero mean, uniformly distributed noise to all the analytic model values. Controller performance in these cases was significantly degraded with RMS errors ranging from 0.6 to 0.75 depending on the magnitude of the noise. This range of controller performance is consistent with that reported for HDP trained controllers [7].

For both benchmark problems, increasing the amount of noise in the model can lead to instability in the training process. One way of countering this problem is to slow down training by reducing the learning rates used in the weight update process. The combined effect of the noisy model and the reduced learning rates is much slower controller learning or adaptation. A positive side effect of this is that the model noise in the slowed down learning process appears to have an annealing affect, reducing the likelihood of getting stuck in a local optimum far from a desired optimum.

Qualitative Models

Based on the above observations, it is tempting to investigate the use of greatly simplified qualitative models. Such models only determine the sign of the derivatives at each operating point, positive, negative or zero. Since our current experiments are based on plants defined by analytic equations we simply replace all positive derivatives with the value 1 and all negative derivatives with the value -1.

Other authors have hypothesized that qualitative models should be adequate for the pole-cart problem, as there is only a single control variable and straightforward dynamics. Our experience has been that training with such models is generally successful, though usually slower than training with exact models, and is capable of producing controllers of the same quality as when training with exact models.

Narendra's problem is more interesting in this context as it is Multiple Input Multiple Output (MIMO) with significant non-linearity and time delays and requires following an a priori unknown reference trajectory. Significantly, the same sort of qualitative model works rather well in this case. Again training is somewhat slower than when using an exact model. Controller performance on the test trajectory averaged 0.4 RMS error, with a best of 0.32.

The size of the positive and negative values used will vary the "gain" of the training process - thus linking this choice to the selection of learning rates for the training process. Larger values in a qualitative model might require smaller learning rates and vice versa.

Another view of our simplified, qualitative model is that we have constructed a classifier of the plant's qualitative behavior. Our classifier tells us whether a particular variable will increase, decrease or remain unchanged based on the value of some other variable. Estimating such a classifier as a practical matter should be simpler than estimating a quantitative model for the plant. All one needs to determine are the class boundaries in the state space. An important question to explore is with what precision must the class boundaries be known for the qualitative model to be useful?

Experiments with Narendra's system suggest that only very rough models are needed. To demonstrate this, we took the above mentioned qualitative model in which all positive

values were replaced with positive one and all negative values replaced with negative one, and additionally zeroed out all derivatives with absolute value less than 0.1, thus establishing a substantial "don't know" zone between classes. Results with these models were comparable to those using the exact class boundaries reported above, with average RMS error on the test trajectory of 0.38, and a best performance of 0.32. This shows that successful training is possible using only qualitative information for those regions of state space in which the plant's dynamics are unambiguous.

Conclusions

Our experiences outlined above show that the models used in DHP training can be far from perfect and still lead to successful controller training. A variety of options exist for simplifying system identification and modeling in the DHP context. Partial models based on subsets of the available state variables appear to be viable as long as the included variables form an observable system.

Estimation of exact parameter values in system models may not be as important as producing unbiased estimates. In off-line training contexts, the "noise" in these estimated models may even be beneficial in the controller training process, due to its tendency to anneal the critic and controller networks out of unsatisfactory local optima.

The final case of qualitative models is promising because estimating a classifier is in general an easier problem than full regression of a system model. Qualitative models that simply determine whether a derivative takes on a positive, negative or zero value at any operating point are equivalent to pattern classifiers. Their estimation in an on-line context should involve less computational overhead, and should be easier to adapt to non-stationary plants undergoing significant structural change. This could make qualitative models valuable in real time on-line adaptation contexts.

These results suggest that simple or quick and dirty models used in the DHP methodology may well be preferable to using one of the other adaptive critic methods that requires little or no model information for implementation. In particular, a qualitative approach where the plant's dynamics are clearly understood for some regions of state space may in many cases be both efficient and effective.

Acknowledgements

I would like to thank PSU for continuing research support, George Lendaris for all his time, guidance and mentoring, and Annabel & Hal Sacks, Gretchen & Don Liuzzi for the generous use of their kitchen tables.

References

[1] Barto, A.G., R.S. Sutton, & C.W. Anderson, "Neuronlike Elements That Can Solve Difficult Learning Control Problems", *IEEE Transactions on Systems Man & Cybernetics*, vol. 13, pp. 834 - 846.

[2] Lendaris, G. & C. Paintz, "Training Strategies for Critic and Action Neural Nets in Dual Heuristic Programming Method", in *Proceedings of ICNN'97*, Houston, IEEE, 1997, pp. 712 - 717.

[3] Lendaris, G., C. Paintz, & T. Shannon, "More on Training Strategies for Critic and Action Neural Nets in Dual Heuristic Programming Methodology", in *Proceedings of IEEE-SMC'97*, Orlando, IEEE, 1997.

[4] Lendaris, G., & T. Shannon, "Application Considerations for the DHP Methodology", in *Proceedings IJCNN'98*, Anchorage, IEEE, 1998, pp. 1013-1018.

[5] Lendaris, G., T.T. Shannon & A. Rustan, " A Comparison of Model Based Adaptive Critic Training Algorithms for Neuro-control", in *Proceedings of IJCNN'99*, Washington D.C., IEEE, 1999.

[6] Narendra, K. & S. Mukhopadhyay, "Adaptive Control of Nonlinear Multivariable Systems Using Neural Networks", *Neural Networks*, vol. 7, #5, 1994, pp. 737-752.

[7] Prokhorov, D., *Adaptive Critic Designs and their Application*, Ph.D. Dissertation, Department of Electrical Engineering, Texas Tech University, 1997.

[8] Prokhorov, D. & D. Wunsch, "Adaptive Critic Designs", *IEEE Transactions On Neural Networks*, vol.8(5), 1997, pp. 997-1007.

[9] Prokhorov, D., R. Santiago & D. Wunsch, "Adaptive Critic Designs: A Case Study for Neurocontrol", *Neural Networks*, vol. 8 (9), pp. 1367 - 1372.

[10] Santiago, R. & P.J. Werbos, "New Progress Towards Truly Brain-Like Control", *Proceedings of WCNN'94*, San Diego, CA, 1994, pp. 27-33.

[11] Visnevski, N. & D. Prokhorov, "Control of a Nonlinear Multivariable System with Adaptive Critic Designs", C. Dagli, et al., eds., *Intelligent Engineering Systems Through Artificial Neural Networks: Proceedings of ANNIE'96*, vol. 6, ASME Press, 1996, pp. .

[12] Werbos, P.J., "A Menu of Designs for Reinforcement Learning Over Time", in Miller, W.T., R.S. Sutton, & P.J. Werbos eds., *Neural Networks for Control*, MIT Press, Cambridge, MA, 1990, pp. 67 - 95.

[13] Werbos, P.J., "Approximate Dynamic Programming for Real-Time Control and Neural Modeling", in D.A. White & D.A. Sofge eds., *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches,* Van Nostrand Reinhold, New York, 1992, pp. 493 - 525.