Non-Linear Stochastic Control in Continuous State Spaces by Exact Integration in Bellman's Equations

Daniel Nikovski and Matthew Brand
Mitsubishi Electric Research Laboratories
201 Broadway, Cambridge, MA 02139, USA
nikovski@merl.com, brand@merl.com

We present an algorithm for sequential control of tasks with non-linear stochastic dynamics in continuous state spaces, characterized by inhomogeneous noise. The algorithm performs approximate value iteration steps on a select set of prototypical states whose cost-to-go is approximated by means of a radial-basis function network. This allows the resulting Bellman's equations to be integrated exactly with respect to the transition densities of a large class of stochastic dynamical systems, resulting in a fast and efficient modified value-iteration procedure.

Introduction

One of the principal problems in the fields of planning and optimal control under uncertainty is how to find optimal policies for controlling dynamical systems with continuous state spaces and non-linear stochastic transition dynamics. While the actions used to control the system are usually continuous and are being applied in continuous time as well, the resulting most-general formulation of the control problem involves sets of non-linear differential equations and is typically very hard to analyze and solve (Stengel 1994). However, a large class of systems in the fields of robotics and automatic control can be solved successfully by applying only discrete actions at regular time intervals (discrete time). Such are robot navigation, where actions such as turning left, turning right, and going straight are usually sufficient (Thrun 2000), as well as various instances of bang-bang control such as pole balancing, swinging up pendulums, etc. (Jervis & Fallside 1992; Spong 1995). At least in theory, it is also possible to discretize the state space, but in practice, the resulting sequential decision-making problem suffers from the well-known "curse of dimensionality" and has an intractably large number of states.

Consequently, we are considering the case when a continuous-state system is controlled at discrete time intervals by actions a_i from a discrete set A, according to stochastic equations of motion

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), a(t)) + w(\mathbf{x}(t), a(t)), \qquad (1)$$

where $f(\mathbf{x}, a)$ is a deterministic nonlinear function of the current multi-dimensional state $\mathbf{x}$ and the applied action a, and $w(\mathbf{x}, a)$ is a noise term with a distribution which depends on $\mathbf{x}$ and a. Such systems models can either be derived analytically, or learned from sampled transitions in a supervised learning setting.

Each state transition incurs a transition cost $C(t) = c(\mathbf{x}(t), a(t))$ which is assumed to be bounded. The objective of the controller is to find an optimal policy $a = \pi^*(\mathbf{x})$, which minimizes the expected cumulative discounted cost V_π with discounting factor $0 < \beta < 1$ for each starting state $\mathbf{x}_0$:

$$V_\pi(\mathbf{x}_0) = E_\pi[\sum_{t=0}^{\infty} \beta^t C(t)], \qquad (2)$$

for any policy π. Bellman's optimality principle states that for discounted costs and fully-observable Markov decision processes (MDPs), it is true that for each state $\mathbf{x}$,

$$V_{\pi*}(\mathbf{x}) = \min_a \{E[c(\mathbf{x}, a)] + \beta \int_{\mathbf{y}\in\Omega} Pr(\mathbf{y}|\mathbf{x}, a) V_{\pi*}(\mathbf{y}) d\mathbf{y}\}, \qquad (3)$$

where $\mathbf{y}$ ranges over the whole state space Ω, and $Pr(\mathbf{y}|\mathbf{x}, a)$ is the conditional density expressing the probability that the system would end up in state $\mathbf{y}$ at time $t+1$ if action a is applied in state $\mathbf{x}$ at time t (Bertsekas 2000).

One of the principal algorithms for exact solution of sets of Bellman's equations is value iteration which produces successive improvements of the estimated optimal value function $V(\mathbf{x})$ for all states $\mathbf{x} \in \Omega$ (Puterman 1994). The value iteration algorithm starts with an initial arbitrary estimate $V(\mathbf{x})$ for all $\mathbf{x} \in \Omega$ and repeats the following recurrent update, known as a Bellman back-up, for each state in the state space Ω, until the estimated optimal value function converges within a predefined tolerance:

$$V(\mathbf{x}) := \min_a \{E[c(\mathbf{x}, a)] + \beta \int_{\mathbf{y}\in\Omega} Pr(\mathbf{y}|\mathbf{x}, a) V(\mathbf{y}) d\mathbf{y}\} \qquad (4)$$

Bellman back-ups can be performed efficiently in small state spaces, but cannot be applied directly to MDPs with continuous state spaces for two reasons: first, the estimates of the optimal value function $V(\mathbf{x})$ cannot be stored in memory when the state space is infinite or very large, and second,

performing Bellman back-ups over the whole state space is not computationally feasible. The latter obstacle has two manifestations. First, the starting states of system transitions are infinitely many, and proper backing up of the cumulative cost would involve a Bellman back-up for each of them. Second, when the system dynamics are stochastic, even a single starting state can have infinitely many successor states, whose cumulative costs have to be taken in consideration.

The problem of infinitely large storage requirements for the cumulative costs of states is usually addressed by employing various universal function approximators (UFA). UFA are commonly used to represent the value function in a concise form: $V(\mathbf{x}) \approx F(\mathbf{x}, \theta)$, where θ is a vector of parameters of finite (and usually small) size. In such cases, Bellman back-ups are performed only for a small set of sampled states, with the objective of finding a value for θ which satisfies the Bellman equations at the sampled states as accurately as possible. This class of algorithms are collectively known as Approximate Value Iteration (AVI) (Bertsekas & Tsitsiklis 1996).

The choice of function approximator is very critical for the success of an AVI algorithm. While parametric approximators such as neural networks have had spectacular successes in learning the value functions of complex decision problems such as backgammon (Tesauro 1992), job-shop scheduling (Zhang & Dietterich 1996), and elevator dispatching (Crites & Barto 1998), they have also been shown to fail completely for many trivial problems (Boyan & Moore 1995). As a consequence, much research has been concentrated on finding stable value-function approximators which converge to a reasonable approximation of the true optimal value function.

Tsitsiklis and van Roy proposed an algorithm for value-function approximation by means of radial-basis functions and reported good results on decision problems with large state spaces (Tsitsiklis & van Roy 1996). However, their algorithm is guaranteed to converge only under quite restrictive assumptions about the basis functions used. Furthermore, the basis functions have to be provided by a human expert. Gordon demonstrated that averaging approximators such as k-nearest neighbors and kernel regression always lead to stable approximation, unlike "exaggerating" approximators such as neural nets and linear regression (Gordon 1996). In line with Gordon's results, Ormoneit proposed a universally stable algorithm for value-function approximation by means of kernel regression (Ormoneit & Sen 1999). Thus, even though finding the best value-function approximator for MDP problems is still an open question, a large number of suitable choices already exist and have produced good results.

However, most AVI algorithms use value-function approximators to handle only one manifestation of the computational problem created by infinite state spaces. Even when value iteration is performed only at a small set of select states, the question remains how to take into consideration the expected cost-to-go over *all* successor states $\mathbf{y} \in \Omega$ under an action a for a single starting state $\mathbf{x}$. Finding this estimate would involve integration over the whole state space Ω, which is computationally unfeasible. In rare cases, when a state transitions to only few successor states, the integral can be replaced by a sum over those states. This approach has been explored previously (Tsitsiklis & van Roy 1996), but is only viable when transition densities are indeed sparse.

For the general case when the number of successor states is large or infinite, one common approach is to replace the expectation of the cost-to-go over all states with the cost-to-go of the most likely successor state, which is equivalent to ignoring the noise term $w(\mathbf{x}, a)$ in the state transition function of the controlled dynamical system. However, the resulting policies cannot make a difference between two actions with the same deterministic transition function $f(\mathbf{x}, a)$, but different noise terms, and have no reason to prefer one to the other. As a result, sub-optimal actions can be selected. Furthermore, when the transition dynamics of a particular action are multi-modal, replacing the expected cost-to-go of all successor states with the cost-to-go of the expected successor state can be completely wrong, because that state might not even be likely.

A third possible solution is to estimate the expected cost-to-go of successor states by sampling them from the transition density of the current state, and averaging their cost-to-go estimates obtained under the current set of approximation parameters. However, obtaining such Monte Carlo estimates of the expected cost-to-go would typically require a lot of successor states to be sampled (Bertsekas & Tsitsiklis 1996).

Exact Bellman Back-ups in Infinite Spaces

We propose an algorithm for performing exact Bellman back-ups in infinite state spaces by matching the distributions of the noise in the transition dynamics to a suitable value-function approximator — one which allows exact computation of the expected value of all successor states $\mathbf{y} \in \Omega$ for a particular starting state $\mathbf{x}$ and action a. An even more general case of transition dynamics than equation (1) is when the density is a mixture of multivariate Gaussians:

$$Pr(\mathbf{y}|\mathbf{x}, a) = \sum_{i=1}^{n} p_i N(\mathbf{y}; f_i(\mathbf{x}, a), \Sigma_i), \qquad \sum_{i=1}^{n} p_i = 1,$$

where p_i are mixture proportions, and henceforth $N(\mathbf{x}; \mu, \Sigma)$ will denote a multivariate Gaussian density evaluated at point $\mathbf{x}$, with mean μ, covariance matrix Σ, and number of dimensions equal to that of the state space. (If the transition dynamics have a deterministic component, such as in equation (1), it can be represented by a mixture component with zero variance.) The vast majority of dynamical system models use a deterministic component plus either a single Gaussian term with a constant covariance matrix, or one that depends on the action a. For example, in mobile robot navigation, the location and orientation of the robot are commonly assumed to come from independent Gaussian distributions whose variance depends on the action taken (Thrun, Burgard, & Fox 1998). (For example, robot rotation is known to be much more imprecise than robot translation.) We will derive the case for a single action-dependent Gaussian noise component below, the extensions to a mixture of Gaussians being straightforward.

Value-Function Approximator

The cost-to-go approximator we are using is very similar to a radial-basis function (RBF) network, with the addition of a matrix A, which pre-multiplies the linear coefficients:

$$V(\mathbf{x}) = U(\mathbf{x})Av', \tag{5}$$

where $U(\mathbf{x})$ is a row vector containing the values of a set of non-linear basis functions evaluated at $\mathbf{x}$, and v' is the column vector of m coefficients which have to be determined by approximate value iteration. We use as RBFs m Gaussian kernels with means $\boldsymbol{\mu}_i$ and covariance matrices Σ_i, $i = 1, 2, ..., m$, which cover the state space as densely as possible. The choice of Gaussians as basis functions is very common in the theory and practice of RBF networks (Girosi & Poggio 1990). The Gaussians can either be placed manually so as to cover the state space well, or can be obtained by clustering the set of states visited under a suitable exploration policy applied to the dynamical system to be controlled.

The inclusion of the matrix A lets us give a meaningful interpretation to the parameter vector v'. We choose it to be the vector of m approximate value-function estimates evaluated at $\mathbf{x} = \boldsymbol{\mu}_i$, $i = 1, 2, ..., m$. We can then compute the matrix A that is consistent with this definition. Writing down the approximations of the value function for $\mathbf{x} = \boldsymbol{\mu}_i$, $i = 1, 2, ..., m$, we obtain

$$v' = \bar{U}Av', \tag{6}$$

where row i of the square matrix $\bar{U}$ of size $m \times m$ is the row vector $U(\boldsymbol{\mu}_i)$, i.e. the element $\bar{u}_{ij}$ of this matrix is the value of the j-th RBF evaluated at the i-th center $\boldsymbol{\mu}_i$. Hence, $A = \bar{U}^{-1}$, and the final approximation scheme is $V(\mathbf{x}) = U(\mathbf{x})\bar{U}^{-1}v'$ for any $\mathbf{x} \in \Omega$.

Exact Integration in Bellman's Equations

Approximate value iteration is performed at the centers $\boldsymbol{\mu}_i$, $i = 1, 2, ..., m$ of the m RBF kernels, according to the equation

$$V(\boldsymbol{\mu}_i) := \min_a \{E[c(\boldsymbol{\mu}_i, a)] + \beta \int_{\mathbf{y}\in\Omega} Pr(\mathbf{y}|\boldsymbol{\mu}_i, a)V(\mathbf{y})d\mathbf{y}\} \tag{7}$$

for each center $\boldsymbol{\mu}_i$, $i = 1, 2, ..., m$. For each kernel, we substitute the form of the transition density and the chosen representation for the approximation scheme into the equation above:

$$V(\boldsymbol{\mu}_i) := \min_a \{E[c(\boldsymbol{\mu}_i, a)] + \beta \int_{\mathbf{y}\in\Omega} N(\mathbf{y}; \mathbf{y}_{ia}, \Sigma_{ia}) \sum_{j=1}^{m} N(\mathbf{y}; \boldsymbol{\mu}_j, \Sigma_j)(\bar{U}_j^{-1} \cdot v')d\mathbf{y}\}$$

where $\bar{U}_j^{-1}$ is the j-th row of the matrix $\bar{U}^{-1}$ and we have used the expansion of the approximation scheme $V(\mathbf{y}) = \sum_{j=1}^{m} U_j(\mathbf{y})(\bar{U}_j^{-1} \cdot v')$, taking into consideration that $U_j(\mathbf{y}) = N(\mathbf{y}; \boldsymbol{\mu}_j, \Sigma_j)$. The vector $\mathbf{y}_{ia} = f(\boldsymbol{\mu}_i, a)$ is the expected value of the successor state $\mathbf{y}$ under action a starting from state $\boldsymbol{\mu}_i$, and Σ_{ia} is its covariance. By recognizing that $\mathbf{y}$ appears only in the two normal densities, this can be rearranged as:

$$V(\boldsymbol{\mu}_i) := \min_a \{E[c(\boldsymbol{\mu}_i, a)] + \beta \sum_{j=1}^{m} z_{iaj}(\bar{U}_j^{-1} \cdot v')\} \tag{8}$$

$$z_{iaj} = \int_{\mathbf{y}\in\Omega} N(\mathbf{y}; \mathbf{y}_{ia}, \Sigma_{ia})N(\mathbf{y}; \boldsymbol{\mu}_j, \Sigma_j)d\mathbf{y} \tag{9}$$

The integrals z_{iaj} involve only the product of two Gaussian densities and it is known that they can be computed analytically, if the space Ω of the successor state y coincides with $\mathbb{R}^D$:

$$z_{iaj} = \frac{(2\pi)^{-D/2}|\boldsymbol{\Sigma}_{iaj}|^{+1/2}}{|\boldsymbol{\Sigma}_{ia}|^{-1/2}|\boldsymbol{\Sigma}_j|^{-1/2}} \cdot \exp[-\frac{1}{2}(\mathbf{y}_{ia}^T\boldsymbol{\Sigma}_{ia}^{-1}\mathbf{y}_{ia} + \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j - \boldsymbol{\mu}_{iaj}^T\boldsymbol{\Sigma}_{iaj}^{-1}\boldsymbol{\mu}_{iaj})],$$

where the new vector $\boldsymbol{\mu}_{iaj}$ and matrix $\boldsymbol{\Sigma}_{iaj}$ are defined as follows:

$$\boldsymbol{\Sigma}_{iaj} = (\boldsymbol{\Sigma}_{ia}^{-1} - \boldsymbol{\Sigma}_j^{-1})^{-1} \tag{10}$$

$$\boldsymbol{\mu}_{iaj} = \boldsymbol{\Sigma}_{iaj}\boldsymbol{\Sigma}_{ia}^{-1}\mathbf{y}_{ia} + \boldsymbol{\Sigma}_{iaj}\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j \tag{11}$$

It can be recognized that the values z_{iaj} can also be computed as

$$z_{iaj} = \frac{N(\mathbf{y}_{ia}; 0, \boldsymbol{\Sigma}_{ia})N(\boldsymbol{\mu}_j; 0, \boldsymbol{\Sigma}_j)}{N(\boldsymbol{\mu}_{iaj}; 0, \boldsymbol{\Sigma}_{iaj})} \tag{12}$$

We assume that the expected costs $E[c(\boldsymbol{\mu}_i, a)]$ for taking action a at state $\mathbf{x} = \boldsymbol{\mu}_i$ can either be computed from a known analytical model, or a separate cost model has been learned from sampled transition data, and this model has been evaluated with inputs $(\boldsymbol{\mu}_i, a)$ (Girosi & Poggio 1990). The left-hand side of equation (13) can be expressed as an approximation in the chosen basis functions:

$$V(\boldsymbol{\mu}_i) = \sum_{j=1}^{m} U_j(\boldsymbol{\mu}_i)(\bar{U}_j^{-1} \cdot v'), \tag{13}$$

or, in a matrix form, the column vector of dimension m, whose i-th component is $V(\boldsymbol{\mu}_i)$, becomes $\bar{U}\bar{U}^{-1}v'$, which simplifies to v'. By denoting with Z_a the square matrix with elements $Z_a(i, j) = z_{iaj}$, we obtain approximate back-up equations in matrix form:

$$v' := \min_a \{C_a + \beta Z_a\bar{U}^{-1}v'\}, \tag{14}$$

where C_a denotes the column-vector of expected costs under action a, starting from each of the kernel centers: $C_a(i) = c(\mu_i, a)$. The above equation is analogous to Bellman's equations for a discrete MDP with m states, and can be used for approximate value iteration. Its convergence depends on the properties of the matrix products $Z_a\bar{U}^{-1}$ for

each action a. These matrices possess many of the properties of stochastic matrices: Their row-sums are typically very close to one and their spectral radii are strictly < 1. We are currently engaged in a theoretical analysis of convergence but a large set of experiments leads us to believe that this method reliably converges to a fix-point.

The extension to the case when the transition densities are mixtures of Gaussians is straightforward; in this case, the values z_{iaj} can be computed for each Gaussian component of the mixture, and the resulting matrices added up, weighted proportionately to the mixing coefficients.

Experimental Verification

Task Description

The proposed algorithm was verified experimentally on a test problem with a two-dimensional continuous state space, which resembles navigation of a mobile robot. The state of the system changes under five actions according to the following law of motion:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \Delta_x d(a(t)) + w(a(t)), \qquad (15)$$

where $d(a_1) = (0,1)$, $d(a_2) = (1,0)$, $d(a_3) = (0,-1)$, $d(a_4) = (-1,0)$, $d(a_5) = (0,0)$, and Δ_x=1. The noise term $w(a)$ had Gaussian distribution with zero mean and a covariance matrix dependent only on the action taken. For actions a_2 through a_5, the covariance matrix was diagonal with diagonal entries $\mathbf{\Sigma}_i^a(1,1) = \mathbf{\Sigma}_i^a(2,2) = 0.25$, $i = 2,3,4,5$. For action a_1, $\mathbf{\Sigma}_1^a(1,1) = \mathbf{\Sigma}_1^a(2,2) = 2.25$. Thus, action a_1 was much noisier than the remaining four actions. The approximation was performed on a set of radial basis functions whose centers were placed on a regular grid covering the rectangle determined by points $(0,0)$ and $(10,10)$. The standard deviation of all Gaussian kernels was set to half the distance between two neighboring kernels along a state dimension. The objective of control was to reach as fast as possible the goal region defined as a square of size $2\Delta_x$ centered at various points within the same rectangle.

Effect of Cost Structure

Initially, we defined the cost to be 0, if the system entered the goal region as a result of a transition, and 1 elsewhere. While the algorithm converged for all positions of the goal region that we tried, we determined that it sometimes converged to a wrong approximation of the value function. After some analysis, we found the reason to lie in the characteristics of radial basis function networks as function approximators, and in particular, their inability to extrapolate correctly outside of the region populated with radial basis kernels. The approximation is of the form $V(\mathbf{x}) = U(\mathbf{x}) \cdot \theta$, where the values of θ have been determined from the estimated costs-to-go at the centers of the radial basis functions. Since the values of all radial basis functions decrease rapidly as $\mathbf{x}$ goes outside of the region populated by radial basis kernels, while the parameter vector θ remains constant, it is clear that $V(\mathbf{x})$ will also become close to zero in such cases. As a consequence, the RBF approximation optimistically underestimates the cost-to-go when extrapolating, which results in a tendency to choose actions that lead the system outside of the region where the kernels reside.

This situation can be remedied by changing the cost structure of the decision problem. We changed the objective to be maximization of reward rather than minimization of cost, and assigned a reward of 1 when the system entered the goal region, and 0 elsewhere. In this manner, the approximation scheme was underestimating (pessimistically) the rewards in unknown regions of state space, and did not choose actions leading into such regions. Under the new cost structure, the algorithm always converged to a reasonable approximation of the expected value function, for many different grid sizes, motion step sizes, and degree of noise for actions.

Comparison with Deterministic Back-ups

We compared the proposed algorithm with another one that used the exact same approximation scheme for the value function, but ignored the stochasticity of system actions and always assumed that the system transitioned deterministically to the most likely successor state. Under such assumptions, the back-up equations for the centers of the m approximation kernels reduce to

$$v' := \min_a \{C_a + \beta U_a' \bar{U}^{-1} v'\}, \qquad (16)$$

where U_a' is a square matrix of size m whose element $U_a'(i,j) = U_j(f(\mu_i, a)) = N(\mathbf{y}_{ia}; \mu_j, \mathbf{\Sigma}_j)$. In order to compare the performance of the two algorithms, we recorded the cumulative (non-discounted) reward accumulated over 20 time steps, starting from each kernel center, for each of the two algorithms. The optimal action for a particular state $\mathbf{x}$ was assumed to be the optimal action for the closest kernel in state space. (This is only one available option for using an approximate value function in control; many others are possible too (Bertsekas & Tsitsiklis 1996)).

In this experiment, a total of 100 kernels were placed on a 10×10 grid. Thus, the maximal possible cumulative reward was 2000, in the extremely unlikely event that mere chance moved the system to the goal region after the first step and kept it there for 20 time steps until the end of the trajectory. In practice, most policies in this system, including the one which chooses actions randomly, would result in a very low cumulative reward. After each of the algorithms converged to a stable policy, we evaluated each of the two policies ten times and performed a statistical comparison of the results. Note that both algorithms always produce the same result for a given dynamical system, so it is sufficient to compute the two approximate value functions only once.

While both policies resulted in high cumulative rewards, the differences between them were highly significant. The first algorithm, which took into consideration the noise in actions, achieved average cumulative reward of 1029 ± 41.83, while the second algorithm, which ignored the stochasticity in system transitions, achieved average cumulative reward of 960 ± 44.69. The associated t-statistic from a t distribution with 18 degrees of freedom was $t = 3.3770$, which results in a significance level of $p = 0.0017$ for the difference between the two algorithms.

One probable reason why the first algorithm has better performance is that it avoids to use action a_1 which is noisy and unreliable. The second algorithm does not distinguish between actions with different degrees of noise, and as a result has less success in bringing the system to the goal region.

Conclusions and Future Work

We presented an algorithm for approximate value iteration in continuous state spaces with nonlinear stochastic dynamics, which computes exact values for the expected cost-to-go over the whole state space of the system by matching its parametric approximation of the cost-to-go to the noise characteristics of the transition dynamics. The resulting back-up equations exhibit fast and consistent convergence.

We also identified a characteristic problem of RBF network approximations of the cost-to-go — their tendency to underestimate it, when extrapolating outside of the region populated by radial basis kernels. By changing the cost structure, however, we were able to achieve accurate and useful approximations. The advantages of the proposed algorithm were demonstrated on a problem, for which taking into consideration the reliability of actions was important for optimal decision making.

A major direction for expanding this algorithm is to apply it to learned models of transition dynamics which are not limited to a single Gaussian distribution. Various graphical models exist that can represent multi-modal posterior distributions, and could possibly serve as models for the transition dynamics of complex systems.

References

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific.

Bertsekas, D. P. 2000. *Dynamic Programming and Optimal Control*. Belmont, Massachusetts: Athena Scientific. Volumes 1 and 2.

Boyan, J. A., and Moore, A. W. 1995. Generalization in reinforcement learning: Safely approximating the value function. In Tesauro, G.; Touretzky, D. S.; and Leen, T. K., eds., *Advances in Neural Information Processing Systems 7*, 369–376. Cambridge, MA: The MIT Press.

Crites, R. H., and Barto, A. G. 1998. Elevator group control using multiple reinforcement learning agents. *Machine Learning* 33:235.

Girosi, F., and Poggio, T. 1990. Networks and the best approximation property. *Biological Cybernetics* 63:169–176.

Gordon, G. J. 1996. Stable fitted reinforcement learning. In Touretzky, D. S.; Mozer, M. C.; and Hasselmo, M. E., eds., *Advances in Neural Information Processing Systems*, volume 8, 1052–1058. The MIT Press.

Jervis, T. T., and Fallside, F. 1992. Pole balancing on a real rig using a reinforcement learning controller. Technical Report CUED/F-INFENG/TR. 115, Cambridge University, UK.

Ormoneit, D., and Sen, S. 1999. Kernel-based reinforcement learning. Technical Report 1999-8, Department of Statistics, Stanford University.

Puterman, M. L. 1994. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc.

Spong, M. 1995. The swingup control problem for the acrobot. *IEEE Control Systems Magazine, Feb*.

Stengel, R. 1994. *Optimal Control and Estimation*. New York: Dover.

Tesauro, G. 1992. Practical issues in temporal difference learning. *Machine Learning* 8:257–278.

Thrun, S.; Burgard, W.; and Fox, D. 1998. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning* 31:29.

Thrun, S. 2000. Probabilistic algorithms in robotics. *AI Magazine* 21(4):93–109.

Tsitsiklis, J. N., and van Roy, B. 1996. Feature-based methods for large scale dynamic programming. *Machine Learning* 22(1/2/3):59–94.

Zhang, W., and Dietterich, T. G. 1996. High-performance job-shop scheduling with a time-delay TD(λ) network. In Touretzky, D. S.; Mozer, M. C.; and Hasselmo, M. E., eds., *Advances in Neural Information Processing Systems*, volume 8, 1024–1030. The MIT Press.