Efficient exploration for optimizing immediate reward

Dale Schuurmans
Department of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1, Canada
dale@cs.uwaterloo.ca

Lloyd Greenwald
Department of Mathematics and Computer Science
Drexel University
Philadelphia, PA 19104-2875, USA
lgreenwald@mcs.drexel.edu

Abstract

We consider the problem of learning an effective behavior strategy from reward. Although much studied, the issue of how to use prior knowledge to scale optimal behavior learning up to real-world problems remains an important open issue.

We investigate the inherent data-complexity of behavior-learning when the goal is simply to optimize immediate reward. Although easier than reinforcement learning, where one must also cope with state dynamics, immediate reward learning is still a common problem and is fundamentally harder than supervised learning.

For optimizing immediate reward, prior knowledge can be expressed either as a bias on the space of possible reward models, or a bias on the space of possible controllers. We investigate the two paradigmatic learning approaches of *indirect* (reward-model) learning and *direct*-control learning, and show that neither uniformly dominates the other in general. Model-based learning has the advantage of generalizing reward experiences across states and actions, but direct-control learning has the advantage of focusing only on potentially optimal actions and avoiding learning irrelevant world details. Both strategies can be strongly advantageous in different circumstances. We introduce hybrid learning strategies that combine the benefits of both approaches, and uniformly improve their learning efficiency.

Introduction

Reinforcement learning and control learning are significant subareas of machine learning and neural network research. Although a lot of effort has gone into studying these problems, it is fair to say that our current understanding of reinforcement and control learning problems still lags behind our comprehensive understanding of supervised learning. One of the main issues in reinforcement and control learning is *scaling up*. Although several "general purpose" learning algorithms such as Q-learning (Watkins 1989; Watkins & Dayan 1992) have been developed for reinforcement learning problems, and recent variants have been proved to learn "efficiently" in some sense (Kearns & Singh 1998), getting these algorithms to solve significant real world problems ultimately requires the use of prior knowledge and domain constraints, just as in supervised learning. However, the issue of how to effectively exploit domain knowledge in reinforcement learning is still an open research issue—if not the central one in this area (Mahadevan & Kaelbling 1996; Sutton & Barto 1998).

In this paper we consider a simplified version of reinforcement learning and focus on learning to optimize *immediate* rewards. In standard (full) reinforcement learning the agent's actions affect not only its immediate rewards but also its future states, and therefore the agent must *tradeoff* any immediate gains against the long term payoffs it might receive by following certain paths (Sutton & Barto 1998). Here we consider a simpler interaction where the agent's actions determine its immediate reward, but do not affect its future states. That is, the agent interacts with the environment by receiving a current state s, chosing an action a, and then receiving an immediate reward r; but the next world state is then chosen *obliviously* to the agent's action. (For example, we can assume that the environment choses states independently from a stationary random source.) The goal of the agent, then, is to learn a behavior map $c : S \to A$ that optimizes its expected (immediate) reward.

Although simpler than full reinforcement learning, immediate reward learning has often been studied in the literature under the name of "associative reinforcement learning" (Kaelbling, Littman, & Moore 1996; Barto & Anandan 1985; Ackley & Littman 1990) and "associative search" (Sutton & Barto 1998). Examples of this problem include skill acquisition from "episodic" training and learning from repeated "attempts," where the agent must experiment with its actions in order to achieve a desired outcome. Some interesting case studies that have appeared in the literature are: learning the inverse kinematics of a robot manipulator (*i.e.*, learning which joint angle settings will place a robot's hand at a desired location in its workspace) (Jordan & Rumelhart 1992; DeMers & Kreutz-Delgado 1997), learning to volley a ping pong ball to a desired location (Moore

1990), learning to sink billiard balls in designated pockets (Moore 1990), learning to deflect an air hockey puck towards a goal (Brodie & DeJong 1998), learning "hit/no-hit" strategies for blackjack (Sutton & Barto 1998), learning to shoot baskets (illustrative example from (Jordan & Rumelhart 1992)), learning to putt a golf ball into a hole (illustrative example from (Sutton & Barto 1998)), learning to behave optimally with limited resources (Russell, Subramanian, & Parr 1993), neural networks for learning "one shot set-point" control (Miller, Sutton, & Werbos 1990), and learning the inverse of functions with neural networks (Bishop 1995; Kindermann & Linden 1990). In each of these cases, the environment presents a problem to the agent, the agent acts to achieve the desired goal, and the environment immediately responds with an indication of success/failure; then the cycle repeats *independently* of the previous events.

Note that, even though this problem is simpler than the full reinforcement learning problem, associative reward learning is still fundamentally harder than standard supervised learning. This is because in reward learning we only receive *partial* evaluation feedback from the environment. That is, in supervised learning, every training instance $\langle x, y\rangle$ evaluates *every* candidate function $f : X \to Y$ via the prediction error $loss(f(x), y)$. In a reward learning problem, by contrast, each training instance $\langle s, a, r\rangle$ evaluates only a *subset* of the candidate control functions $c : S \to A$; namely, those that would have taken action a in state s (Barto & Anandan 1985; Kaelbling, Littman, & Moore 1996). The reward r says nothing *directly* about what would have been obtained if a controller took a different action a' in situation s.

One key observation is that this type of partial feedback automatically creates a distinction between two types of prior knowledge/constraints that does not exist in complete-feedback learning: prior knowledge could be about the set of possible reward models one might face in the world, or it could specify a restricted class of controllers one might be limited to considering. This distinction is mirrored in two fundamental learning approaches that one might consider for this problem:

Model-based learning: Do we learn a model of the total reward function, and then infer the best candidate controller?

Direct-control learning: Do we restrict attention to a set of possible controllers, and attempt to identify a good one directly?

Note that this distinction is independent of temporal credit assignment and exploration/exploitation tradeoff issues. Thus, we can investigate the issue of model-based versus direct-control learning in a very simple setting. In fact, in this paper we will consider the problem of *batch* learning a good controller from a stationary distribution of training instances. The simplicity of this framework allows us to make some straightforward yet telling observations about the relative benefits of the two learning approaches. Here we can clearly determine when one is advantageous over the other, and devise hybrid learning strategies that dominate the performance of both.

We begin the investigation by formalizing our model and drawing the major distinctions we wish to make. We then briefly investigate each of the two fundamental learning approaches—model-based and direct-control learning—and present learning strategies and data-complexity results for each. The paper then brings these two investigations together and asks which learning strategy is best. We show that in fact neither approach dominates the other in general: for some classes of reward models it is best to learn the model first and then infer the optimal controller, but for other classes it is best to ignore the models entirely and just focus on the corresponding class of potentially optimal controllers. This distinction is made strictly in terms of the *data-complexity* of learning, and is not necessarily tied to the issue of computational efficiency (as is sometimes suggested, for example in (Wyatt 1997)).

Finally, we consider whether there are hybrid learning strategies that are neither purely model-based nor controller-based, but that can strictly dominate the performance of both. We demonstrate that such learning strategies do indeed exist.

Formulation

The learner interacts with the world according to a protocol where the world chooses a state $s \in S$, the learner responds with an action $a \in A$, and the world returns a scalar reward $r \in \mathbb{R}$; and the cycle repeats. The reward process could be stochastic (or even nonstationary) in general, but for simplicity we will find it convenient to think of the rewards as being generated from a deterministic function $m_w : S \times A \to \mathbb{R}$, which we refer to as the (true) reward model.

The direct goal of reward learning is not to identify the underlying reward model *per se*, but rather identify a good behavior strategy $c : S \to A$. Of course, this might be achieved indirectly, by first approximating the reward model and then inferring a good controller. (But whether one would actually want to learn a controller this way is exactly the point of this investigation.) Notice that attempting to identify a good control function directly from training data introduces the problem of incomplete feedback: each training instance $\langle s, a, r\rangle$ evaluates only a *subset* of possible controllers—namely, those controllers c such that $c(s) = a$—but does not directly inform us about the rewards that other controllers might have obtained in this state. Therefore, given an accumulated sequence of training instances $\langle s_1, a_1, r_1\rangle, ..., \langle s_t, a_t, r_t\rangle$ we can estimate the expected rewards of a set of controllers in the usual way, but the effective sample size of these estimates will *vary* from controller to controller; *i.e.*, some will have been "tried" more than others. (This is unlike standard supervised learning where each candidate prediction function f can be evaluated by *every* training example $\langle x, y\rangle$, and thus the effective sample size is always uniform among the candidates.) So there is an inherent exploratory aspect to the learning problem here—the learner must choose actions that evaluate different parts of the controller space (at different rates) to build up enough evidence to reliably distinguish good candidates from bad.

Formally, we assume successive states are drawn independently according to some stationary distribution P_S, and therefore characterize the world by a state distribution P_S and the reward model m_w. For each state $s \in S$ a controller c picks an action $c(s) \in A$ and receives reward $m_w(s, c(s)) \in I\!R$. Since successive states are independent of a controller's actions, we can characterize the expected reward of a controller simply in terms of its expected *immediate* reward

$$\mathrm{E}R(c) = \int m_w(s, c(s))\, d\mathrm{P}_S.$$

In this paper we also focus on a *batch* learning protocol where the learner is free to choose actions to gain information in an initial training phase, but then settles on a fixed controller in the subsequent test phase. The significance of this assumption is that the need for *exploitation* is completely eliminated from the training phase, and the learner can concentrate solely on gaining information. That is, the problem is entirely *exploration.* (We therefore do not address the exploration/exploitation tradeoff directly in this paper. The benefit of focusing on the batch paradigm instead of the customary on-line model is that we can still formulate the distinction between model-based and direct-control learning, but compare them in a much simpler setting that permits provable separations between the two approaches and clear suggestions for new learning strategies.)

We now describe the two approaches to learning that we will be considering. The key distinction is how we express our prior knowledge/constraints on the learning problem.

The first approach, embodied by direct-control learning, is just to directly consider a restricted class of control functions C. In this case, C expresses any prior constraints we have about the solutions to the learning task, but does not express direct knowledge about the reward model m_w. In this situation the learner is forced to learn from partial evaluations, since any action it chooses can only return information about a *subset* of C. Learning strategies in this situation therefore amount to strategies for deciding which subsets of C to examine in order to accumulate evidence about which controllers are best and which are suboptimal.

The second approach we consider is to directly learn the reward model $m_w : S \times A \to I\!R$ and then, once an accurate reward model has been acquired, using it to deduce a good controller. In this situation prior knowledge is not expressed as a restricted class of controllers, but rather as a restricted class of possible reward models M. This type of learning seems advantageous since it is just a *supervised* learning task; that is, each training instance $\langle s, a, r\rangle$ evaluates *every* possible model m in M. (This is a slightly novel supervised learning problem however, in that it has both passive and active elements—the world chooses the state s but the learner chooses the action a.)

On the face of it, the model-based approach seems superior to direct-control learning, since supervised learning is intuitively easier than partial feedback learning. However, to properly compare the two approaches, we need to adequately control for their different forms of prior knowledge. Note that there is a natural relation between reward models and controllers: for any reward model $m : S \times A \to I\!R$ there is a controller $c_m : S \to A$ that is optimal for m (perhaps more than one). Thus (since states are independent of actions) we can characterize the optimal controller c_m as the one which takes the immediately optimal action in each state:

$$c_m(s) = \operatorname*{argmax}_{a \in A} m(s, a).$$

Therefore from a class M of possible reward models, we obtain a corresponding induced class C_M of potentially optimal controllers. From this correspondence we can formulate a direct comparison between the two learning approaches: Given a class of reward models M and implied class of controllers C_M, how do direct-control learning and model-based learning compare in terms of inherent data-efficiency? Below we show that neither approach dominates the other in general. However we then show that there is a hybrid learning strategy which dominates both.

Direct-control learning

We first consider the direct-control approach to learning. Here prior knowledge is expressed as a restricted class of control functions $C = \{c : S \to A\}$ which we assume contains a good candidate. Notice that this says nothing directly about the reward model m_w (except perhaps that one of the candidate controllers is optimal) so in principle we have no way of generalizing the outcome $\langle s, a, r\rangle$ of a particular action a to other actions a'. As noted above, direct-control methods have to accumulate evidence for the quality of each controller in a "differential" fashion: each training instance evaluates only a subset of the controllers, therefore some controllers will have been "tried" more often than others and consequently have a higher quality estimate of their true performance. So direct-control learning methods need to decide which actions (controllers) to focus their attention on at each stage.

Direct-control learning methods were the focus of much early research on associative reward learning (Kaelbling, Littman, & Moore 1996; Kaelbling 1994; Ackley & Littman 1990; Barto & Anandan 1985; Williams 1988; 1992), learning in behavior based robotics (Maes & Brooks 1990), neural networks for learning direct inverse control (Brodie & DeJong 1998; DeMers & Kreutz-Delgado 1997; Mel 1989), and neural nets for learning the inverse of functions (Bishop 1995; Kindermann & Linden 1990). There is a large and interesting literature on this approach (even though most recent research has focused on the model-based approaches considered below).

To gain a concrete understanding of this problem, we will investigate a simple version where we assume rewards are 0-1 valued and that there exists a perfect controller in C (*i.e.*, a controller $c \in C$ that always receives reward 1). This allows us to formulate a very simple "PAC learning" version of the task.

Problem: Batch DC-Learning. Given a class of controllers C, an approximation parameter ϵ, and a reliability parameter δ; with probability at least $1-\delta$ return a controller c whose expected reward is at least $1-\epsilon$, for any distribution P_S and reward model m_w.

The benefit of this formalization is that we can now ask concrete questions and prepare for future comparisons: What are good learning strategies for batch DC-learning, and how can we measure the "complexity" of C so as to quantify the inherent data-complexity of the task? (That is, how can we quantify the "strength" of the prior knowledge encoded by C?)

To measure the complexity of C we note that in the two action case it suffices to use the standard notion of VC-dimension.[1] (For the multi-action case we need to resort to the generalized notions of VC-dimension developed by (Ben-David *et al.* 1995; Natarajan 1991).) We observe that, as in standard PAC learning, there is a simple generic learning strategy that achieves near optimal data-efficiency.

Procedure: DC-Learn. Proceed in a series of rounds. For each round, observe random states $s \in S$ and take uniform random actions $a \in A$ until every controller $c \in C$ has been "tried" at least once; that is, for every $c \in C$ there exists an example $\langle s_i, a_i, r_i \rangle$ such that $c(s_i) = a_i$. (Below we note that this occurs within a reasonable number of examples for most C.) Repeat for a number of rounds that is sufficient to obtain uniform ϵ-accurate estimates of the expected reward of every controller in C with probability at least $1-\delta$ (using the standard PAC bounds from VC-theory, and its generalizations (Blumer *et al.* 1989; Ben-David *et al.* 1995)). Return any controller that has always been successful every time it has been "tried".

The idea here is simply to accumulate evidence across the entire space of possible controllers to reliably distinguish the good candidates from bad. As simple-minded as this procedure seems, it turns out that it is impossible to dramatically improve its data-efficiency in terms of scaling in ϵ, δ and the VC-dimension of C.

Proposition 1. For any $\epsilon > 0$, $\delta > 0$ and class C with VC-dimension d, procedure DC-learn correctly solves the batch DC-learning problem and halts with an expected sample size of $O\left(\frac{1}{\epsilon}\left((d^2 \ln d)(\ln \frac{1}{\epsilon}) + \ln \frac{1}{\delta}\right)\right)$.

(Proof idea) The key step is showing that each round of DC-learn halts within a reasonable number of training examples. This is proved by observing that the only way a controller can be completely avoided is by always matching the actions of its "negation". But the class of negated control functions also has small VC-dimension, and the probability of staying in this class by choosing random actions becomes vanishingly small as the sample size increases. This leads to an $O(d \ln d)$ expected stopping time for each round.

The next proposition shows that it is impossible to improve on the performance of DC-learn beyond a $(d \ln d)(\ln \frac{1}{\epsilon})$ factor.

Proposition 2. It is impossible to solve the batch DC-learning problem with an expected sample size that is less than $\Omega\left(\frac{1}{\epsilon}(d + \ln \frac{1}{\delta})\right)$.

(Proof idea) The idea is simply to fix a set of shattered states $s_1, ..., s_d$ and choose a difficult distribution such that every state must be seen to guarantee a good controller, and yet one state is left out with significant probability if the expected sample size is too small (Schuurmans & Greiner 1995).

Although the generic direct-control learning procedure DC-learn does not seem very refined, it has some interesting advantages over well-known learning algorithms in the literature: For example, consider the IEKDNF algorithm from (Kaelbling 1994), which learns controllers $c : \{0,1\}^n \to \{0,1\}$ that can be expressed as k-DNF formulae. The class of k-DNF controllers clearly has finite VC-dimension, so the procedure DC-learn is guaranteed to reliably learn a near-optimal controller using a reasonable sample size. However the IEKDNF procedure presented in (Kaelbling 1994) is not! In fact, there are simple cases where IEKDNF is guaranteed to converge to a bad controller, even when a perfect controller exists in the assumed class C_{k-DNF}.

(To see this, consider a state space described by two bits $s = \langle x_1, x_2 \rangle$ and concentrate on the class of pure disjunctive controllers C_{1-DNF}. Assume the distribution P_S puts probability 1/3 on state $\langle 0,1 \rangle$ and 2/3 on $\langle 1,0 \rangle$, and the reward model is such that $m_w(s,1) = 1$ and $m_w(s,0) = 0$ for all states s. Then, using the notation of (Kaelbling 1994), we let $er(x_i, a)$ denote the expected reward of a specific controller c which takes actions $c(\langle x_1, x_2 \rangle) = a$ if $x_i = 1$ and $\neg a$ if $x_i = 0$. In this case we have $er(x_1, 0) = 1/3$, $er(x_1, 1) = 2/3$, $er(x_2, 0) = 2/3$, and $er(x_2, 1) = 1/3$. So here IEKDNF will converge to the controller $c(\langle x_1, x_2 \rangle) \equiv x_1$ with probability 1, and this controller has expected reward

[1] It is often claimed that the two action case reduces to standard supervised learning—by making the hypothesis that $\langle s, a, r \rangle$ implies $\langle s, \neg a, \neg r \rangle$. But this cannot work in general (Sutton & Barto 1998). If the actions are not randomly sampled with a *uniform* distribution, then an inferior action can accidentally demonstrate a higher expected reward. Barto and Anandan (Barto & Anandan 1985) reduce the two-action case to supervised learning by making the *assumption* that $m(s, \neg a) = 1 - m(s, a)$. But this just amounts to making explicit assumptions about the reward model. We address such a model-based approach to learning in the next section.

2/3. However, the optimal controller in C_{1-DNF}, $c(\langle x_1, x_2\rangle) \equiv x_1 \vee x_2$, actually has expected reward 1; and *this* is the controller that will be discovered by DC-learn with probability at least $1-\delta$ for reasonable ϵ.)

Given that the IEKDNF algorithm was actually designed to cope with the exploration/exploitation tradeoff this might not seem like an appropriate comparison. However, notice that convergence to a suboptimal controller means that IEKDNF could not compete with DC-learn after the batch training phase is complete, even in an on-line evaluation. That is, convergence to a sub-optimal controller is an undesirable property whether one is using a batch or an on-line assessment. This example demonstrates the benefit of our analysis: carefully evaluating the data-complexity of learning led us to uncover a weakness in an existing learning procedure that had not been previously observed.

Model-based learning

We now consider the alternative model-based approach to learning. Here we consider prior knowledge that is expressed in a rather different form: we assume the world's reward model m_w belongs to some restricted class of models M. Learning a good reward model $m : S \times A \to I\!R$ from training examples $\langle s, a, r\rangle$ is a standard supervised learning problem, so it seems like it should be easier than direct-control learning. Once an adequate model $m \in M$ has been identified, we can simply infer a corresponding optimal controller c_m and return this as the final hypothesis for the test phase.

This model-based approach to learning control has been commonly pursued in the literature on associative reinforcement learning (Munroe 1987; Kaelbling, Littman, & Moore 1996), particularly under the guise of learning "forward models" for control (Moore 1990; 1992; Jordan & Rumelhart 1992).[2]

As in the previous section, to develop a concrete understanding of this problem and to facilitate comparisons with the direct-control approach, we consider a simple version of the learning task where we assume that the rewards are 0-1 valued, and that the class of reward models M always contains the true model m_w.

Problem: Batch MB-Learning. Given a class of reward models M, an approximation parameter ϵ, and a reliability parameter δ; with probability at least $1-\delta$ return a reward model m that agrees with the true reward model m_w on every action for at least $1-\epsilon$ of the states, for any state distribution P_S and any true reward model $m_w \in M$.

[2]Note that Moore (Moore 1990; 1992) does not consider a restricted class of models per se, but his work makes a strong model-based assumption that similar actions in similar states will yield similar outcomes. We view this as relying on the true reward model to be sufficiently well-behaved so that one can generalize across states and actions. Generalizing in this way is what we are characterizing as a model-based approach to learning.

Clearly, identifying an accurate model of the payoff function to this extent is sufficient to infer a near optimal controller: If successful, we can correctly deduce the optimal action for at least $1-\epsilon$ of the states and therefore return a controller that achieves expected reward at least $1-\epsilon$. Note that this is a slightly unorthodox supervised learning problem in that the states are observed passively but the actions chosen actively, and both are inputs to the target function m_w. Moreover, we have to *identify* the marginal reward map $A \to I\!R$ exactly for most states $s \in S$, so our success criterion is harder than standard PAC learning. This forces us to define a measure of complexity for M that combines the notion of the VC-dimension of the class M on its domain $S \times A$, with a measure of how hard it is to exactly identify the marginal reward map $A \to I\!R$ for each state. We do this by appealing to the notion of "universal identification sequence" developed in (Goldman & Kearns 1991; Goldman, Kearns, & Schapire 1990) which measures the difficulty of identifying an arbitrary boolean function from a given class of functions. (A universal identification sequence is a set of points that is labeled uniquely by each function in the class.)

As is typical for batch learning tasks, we find that a very simple learning procedure can achieve near optimal performance in terms of scaling in ϵ, δ and the complexity of M.

Procedure: MB-learn. For each state $s \in S$ identify off-line a shortest universal identification sequence $\langle s, a_{i_1}\rangle, ..., \langle s, a_{i_k}\rangle$ for the class of maps $A \to I\!R$ that M induces from s. Proceed in a series of rounds. Given a random state s, choose an action a_{i_j} uniformly at random from the universal identification sequence for s. Repeat until a sufficient number of training examples have been observed. Take any model m that is consistent with every training example and return a corresponding optimal controller c_m. (Note that this procedure is not limited to a finite state or action space, so long as there is a finite bound on the lengths of the universal identification sequences.)

Proposition 3. For any $\epsilon > 0$, $\delta > 0$ and class M with VC-dimension d over the joint space $S \times A$, procedure MB-learn correctly solves the batch MB-learning problem and halts with an expected sample size of $O\left(\frac{t}{\epsilon}(d \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta})\right)$; where t is the length of the longest universal identification sequence over all $s \in S$.

(Proof idea) The only minor trick is to observe that learning a global approximation with error less than ϵ/t on the induced joint distribution over $S \times A$ means that the model could make a mistake on one action for at most ϵ of the states (since each action is observed with conditional probability at least $1/t$ in any state).

Proposition 4. *Any* learning procedure requires at least $\Omega\left(\frac{1}{\epsilon}(d + \ln \frac{1}{\delta})\right)$ training examples to correctly solve the batch MB-learning problem. (Follows from (Ehrenfeucht *et al.* 1989).)

These definitions formalize our intuition about what it means to follow a strictly model-based approach (that is, where we attempt to identify the true payoff structure of the world). Having completed this formalization, we are now in a position to rigorously compare the model-based approach in its simple form against the simple form of direct-control learning described in the previous section.

Comparison

We can now compare the two basic learning approaches considered so far, and examine the presumption that model-based learning is always more efficient than direct-control learning, given that model-based learning apparently receives more feedback from each training example. However, we will see immediately that this fact does not always help. (Note that we draw our distinctions based strictly on the exploration costs of learning and not on issues of computational complexity.)

To conduct the comparison, recall that for any class of reward models $M = \{m : S \times A \to \mathbb{R}\}$ there exists a corresponding class of potentially optimal controllers $C_M = \{c : S \to A\}$. The basis of the comparison is: given a class M and corresponding class C_M, how does the data-efficiency of model-based learning on M compare to that of direct-control learning on C_M?

To best illustrate the fundamental differences, we first consider a simple setting where the world has just one state but there are several possible actions. In this case we can represent a class of reward models M as a matrix whose rows correspond to actions and whose columns represent the individual 0-1-valued reward models $m \in M$.

(A)	m_1	m_2
a_1	0	1
a_2	0	1
a_3	0	1
$\vdots$	0	1
a_n	1	0

First, consider the question of when model-based learning is more data-efficient than direct-control learning. The intuition is clearly that model-based learning should obtain its greatest advantage when the class of reward models allows us to *generalize* the outcome of a single training instance $\langle a, r \rangle$ across a *set* of actions a'. Matrix A demonstrates this in its most extreme form. Here, observing any single action a will allow us to exactly identify the true reward model over the entire action space. Model-based learning in this case will be able to return a successful action after just one training example. On the other hand, direct-control learning needs to examine all n actions (in the worst case) before it could be guaranteed to find a successful one. (This is because a direct-control learner is oblivious to any restrictions on the payoff model, and therefore must follow a fixed exploration strategy through the action space. For any deterministic strategy the rows of the matrix can be permuted and a column selected so that the successful action is the *last* one visited. Similarly, a random (uniform) search strategy takes $|A|$ trials in expectation to find a single successful action.)

(B)	m_1	m_2	m_3	$\cdots$	m_n	m_{n+1}
a_1	0	1	1	1	1	1
a_2	1	0	1	1	1	1
a_3	1	1	0	1	1	1
$\vdots$	1	1	1	0	1	1
a_n	1	1	1	1	0	1

However, this situation could easily go the other way. The problem with naively attempting to identify the reward model is that one could spend significant effort learning details about the model that are simply not relevant for achieving optimal control. For instance, consider Matrix B. Here, any direct-control learning technique needs to examine at most two distinct actions before it is guaranteed to find a successful one. Model-based learning, on the other hand, would need to examine at least $n-1$ actions in this situation before it is guaranteed to identify the true reward model in the worst case. (This is because the data-complexity of identifying the reward model is lower bounded by the "teaching dimension" of the class of models M (Goldman & Kearns 1991). So, for example, the teaching dimension of the last column in Matrix B is $n-1$ (easily determined using the technique of (Goldman & Kearns 1991)). This is a lower bound on the smallest number of actions that can guarantee that this column is distinguished from all others in the matrix.)

Thus, we see that there is no strict domination either way. Of course, these examples might seem trivial in that, once represented this way, it is *obvious* that naive model-based or naive direct-control learning are not sensible approaches to this problem. But that is exactly our point! By conducting a careful evaluation of the data-complexity of learning optimal controllers we have been led to uncover a specific tradeoff that has not always been carefully described. Neither of these two extreme forms of learning can be the proper way to approach associative reward learning problems in general.

These observations easily generalize to the case of stochastic rewards and multi-state problems. For stochastic rewards the learner must repeatedly sample actions to get reliable estimates of their expected reward, but must still identify near-optimal actions with high probability in order to succeed. In this case, constructions very similar to those above can be used to demonstrate the same advantages and disadvantages of the two approaches.

Thus, in general, investigating these two extreme forms of learning highlights the dangers of singlemindedly pursuing either approach, and clarifies the tradeoffs between the two. With this new understanding, we can turn to the question of deriving more refined learning strategies that combine the benefits of both approaches.

Hybrid learning

Given a class of reward models M, it seems sensible to consider a learning approach that exploits this prior knowledge but pays attention to the fact that we are only trying to derive a good controller, and not necessarily learn all the details about the reward model.

(C)	m_1	m_2	m_3	$\cdots$	m_n
a_1	1	0	0	0	0
a_2	0	1	0	0	0
a_3	0	0	1	0	0
$\vdots$	0	0	0	1	0
a_n	1	1	1	1	1

Continuing in the simple setting discussed above, it is actually easy to see what a good hybrid learning strategy might look like. Consider matrix C. In this case the worst case exploration costs of model-based and direct-control learning are $n-1$ and n respectively. However, no exploration is required at all in this case! In this matrix there is a single action, a_n, that is successful in every reward model, and therefore a_n could be immediately returned as the optimal controller without examining a single point. From this simple observation we can derive an optimal learning procedure for the single state task.

Optimal procedure. Given a matrix of possible reward models M, choose a subset of actions $\{a_{i_1}, ..., a_{i_k}\}$ that is a *minimal cover* of the columns of M (where each action "covers" those columns in which it receives reward 1). Explore these actions in any fixed order until either a successful action is found or only one action remains. Return this as the final hypothesis.

Note that this procedure is neither model-based nor direct-control based in the sense discussed above, but it necessarily dominates the data-efficiency of both:

Proposition 5 (Optimality). The size of the minimal cover (minus one) determines the worst case data-efficiency of the optimal procedure. Moreover, the size of this minimal cover (minus one) is also a *lower bound* on the worst case exploration cost of *any* procedure.

(Proof idea) The upper bound is immediate. The lower bound follows because any smaller set of actions must leave some world "uncovered," which permits an adversary to choose a reward model under which every chosen action fails.

It is not hard to scale this basic exploration procedure up to handle multiple states and stochastic rewards. The multi-state case can be handled by considering a minimal cover of the potentially successful actions for each state s, and then follow a strategy that only considers actions that fall within these covers. In fact, we can use this method to construct examples of reward learning problems with boolean controllers $c : \{0,1\}^n \to \{0,1\}^k$ and reward models $m : \{0,1\}^{n+k} \to \{0,1\}$ where our hybrid approach requires $O(1)$ observations to identify a successful controller, but naive model-based and direct-control learning both require $\Omega(k2^n)$ observations in the worst case. (The construction just multiplies a representation of Matrix C across the set of states, $\{0,1\}^n$.)

To handle stochastic rewards, one converts each stochastic reward model $m : s \times a \mapsto$ *"expected reward"* to a 0-1 reward model m', where $m'(s,a) = 1$ if the conditional expected reward of action a given state s is within $\epsilon/2$ of the best conditional expected reward for state s, and $m'(s,a) = 0$ otherwise. Then one can apply the minimal set cover approach on each conditional reward map $A \to \{0,1\}$, as outlined above, and combine this with repeatedly sampling chosen actions for each state to obtain $\epsilon/2$-accurate estimates of their true conditional expected rewards. This leads to a provably correct procedure that has near optimal data-efficiency (as in propositions 1 and 3).

We are currently extending our hybrid learning approach to apply to robot learning problems, and in particular to learning the inverse kinematics for robot manipulators. Here we hope to gain advantages over current approaches in two ways. First, robotic manipulators often have excess degrees of freedom which allow them to place their end-effectors at targets using several different arm configurations. This means that the set of possible control maps c : *"goal position"* $\to$ *"arm configuration"* might contain several optimal solutions. By ignoring the reward models, direct-control learning cannot automatically infer that these distinct controllers are equally effective, and therefore can waste significant training data in detecting their equivalence (or worse, we might accidentally average distinct solutions together to obtain an inferior controller (Bishop 1995, Chapter 6)). On the other hand, a hybrid learning method can exploit the possible error models m : *"target position"* $\times$ *"arm configuration"* $\to$ *"sensed position error"* to determine when two controllers are equally effective and thereby avoid this inefficiency.

Second, sensors must be used to measure the positioning error, and models of these sensors must often be *learned* before one can predict the configurations to use on novel targets. (That is, the robot must learn the error map m.) Although the characteristics of these sensors can vary greatly, which makes the problem of learning the error map quite difficult, most sensors operate in a way that the reported errors are monotonic functions of the true error, and small errors are reported near optimal solutions. Therefore, rather than learn every detail of the error map (which is characteristic of naive model-based learning), we can use the hybrid learning approach to focus on learning an accurate model of the map only in regions that are close to zero-error configurations, and ignore the remaining details of the map that are irrelevant for achieving optimal control.

Conclusions

We considered the two fundamental approaches to associative reward learning, compared their advantages, and proposed a notion for hybrid learning which dominates the performance of both. The key insight is that naively following a strictly model-based or direct-control approach is generally not the most effective way to learn a good controller. Although it is important to exploit prior knowledge that encodes information about the reward models—this can yield tremendous benefits and is essential to scaling up to real-world tasks—careful attention must be paid to the fact that this knowledge will ultimately be used to derive a good controller, and that not every detail about the reward model is needed to accomplish this. The notions for hybrid learning strategies proposed here, we feel, are a first step towards understanding how prior domain knowledge might be concretely exploited to achieve systematic benefits in reward learning tasks.

Future work. Obviously these are preliminary theoretical steps, and there is a long way to go before significant contributions to reward and reinforcement learning practice can be made. Some immediate directions are to (1) pursue application studies of these techniques and compare their performance on real tasks, for example, on the problem of learning the inverse kinematics of a robot manipulator, (2) extend the theoretical model to stochastic rewards, and most importantly (3) to introduce dependence between actions and states and extend our learning strategies and theoretical analysis to deal with state-transition dynamics and temporal credit assignment issues (Kearns & Singh 1998).

References

Ackley, D., and Littman, M. 1990. Generalization and scaling in reinforcement learning. In *NIPS-2*, 550–557.

Barto, A., and Anandan, P. 1985. Pattern-recognizing stochastic learning automata. *IEEE Trans. Sys. Man Cyb.* 15(3):360–375.

Ben-David, S.; Cesa-Bianchi, N.; Haussler, D.; and Long, P. 1995. Characterizations of learnability for classes of $\{0,..,n\}$-valued functions. *J. Comput. Sys. Sci.* 50:74–86.

Bishop, C. 1995. *Neural Networks for Pattern Recognition.* Oxford: Clarendon Press.

Blumer, A.; Ehrenfeucht, A.; Haussler, D.; and Warmuth, M. K. 1989. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM* 36(4):929–965.

Brodie, M., and DeJong, G. 1998. Iterated phantom induction: A little knowledge can go a long way. In *AAAI-98*, 665–670.

DeMers, D., and Kreutz-Delgado, K. 1997. Inverse kinematics of dextrous manipulators. In Omidvar, O., and van der Smagt, P., eds., *Neural Systems for Robotics.* Academic Press.

Ehrenfeucht, A.; Haussler, D.; Kearns, M. J.; and Valiant, L. 1989. A general lower bound on the number of examples needed for learning. *Information and Computation* 82:247–261.

Goldman, S., and Kearns, M. 1991. On the complexity of teaching. In *COLT-91*, 303–314.

Goldman, S.; Kearns, M.; and Schapire, R. 1990. Exact identification of circuits using fixed points of amplification functions. In *FOCS-90*, 193–202.

Jordan, M., and Rumelhart, D. 1992. Forward models: Supervised learning with a distal teacher. *Cognitive Science* 16(3):307–354.

Kaelbling, L.; Littman, M.; and Moore, A. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Kaelbling, L. 1994. Associative reinforcement learning: Functions in k-DNF. *Machine Learning* 15:279–299.

Kearns, M., and Singh, S. 1998. Near-optimal reinforcement learning in polynomial time. In *ICML-98*, 260–268.

Kindermann, J., and Linden, A. 1990. Inversion of neural networks by gradient descent. *Parallel Comput.* 14:277–286.

Maes, P., and Brooks, R. 1990. Learning to coordinate behaviors. In *AAAI-90*, 796–802.

Mahadevan, S., and Kaelbling, L. 1996. The NSF workshop on reinforcement learning: Summary and observations. *AI Magazine* 17.

Mel, B. 1989. Further explorations in visually guided reaching: Making MURPHY smarter. In *NIPS-1*, 348–355.

Miller, W.; Sutton, R.; and Werbos, P., eds. 1990. *Neural Networks for Control.* MIT Press.

Moore, A. 1990. Acquisition of dynamic control knowledge for a robotic manipulator. In *ICML-90*, 244–252.

Moore, A. 1992. Fast, robust adaptive control by learning only forward models. In *NIPS-4*, 571–578.

Munroe, P. 1987. A dual back-propagation scheme for scalar reward learning. In *COGSCI-87*, 165–176.

Natarajan, B. 1991. Probably approximate learning of sets and functions. *SIAM J. Comput.* 20(2):328–351.

Russell, S.; Subramanian, D.; and Parr, R. 1993. Provably bounded optimal agents. In *IJCAI-93*, 338–344.

Schuurmans, D., and Greiner, R. 1995. Sequential PAC learning. In *COLT-95*, 377–384.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction.* MIT Press.

Watkins, C., and Dayan, P. 1992. Q-learning. *Machine Learning* 8(3/4).

Watkins, C. 1989. *Models of Delayed Reinforcement Learning.* Ph.D. Dissertation, Psychology Department, Cambridge University.

Williams, R. 1988. On the use of backpropagation in associative reinforcement learning. In *ICNN-88*, 263–270.

Williams, R. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3/4):229–256.

Wyatt, J. 1997. *Exploration and inference in learning from reinforcement.* Ph.D. Dissertation, Artificial Intelligence Department, University of Edinburgh.