Appears in the Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005), Pittsburgh, PA.

Giving Advice about Preferred Actions to Reinforcement Learners Via Knowledge-Based Kernel Regression

Richard Maclin[†], Jude Shavlik[‡], Lisa Torrey[‡], Trevor Walker[‡], Edward Wild[‡]

Computer Science Department[†]
University of Minnesota Duluth
1114 Kirby Drive
Duluth, MN 55812
rmaclin@d.umn.edu

Computer Sciences Department[‡]
University of Wisconsin Madison
1210 West Dayton Street
Madison, WI 53706
{shavlik,ltorrey,twalker,wildt}@cs.wisc.edu

Abstract

We present a novel formulation for providing advice to a reinforcement learner that employs support-vector regression as its function approximator. Our new method extends a recent advice-giving technique, called Knowledge-Based Kernel Regression (KBKR), that accepts advice concerning a *single* action of a reinforcement learner. In KBKR, users can say that in some set of states, an action's value should be greater than some linear expression of the current state. In our new technique, which we call *Preference KBKR (Pref-KBKR)*, the user can provide advice in a more natural manner by recommending that some action is *preferred* over another in the specified set of states. Specifying preferences essentially means that users are giving advice about *policies* rather than *Q* values, which is a more natural way for humans to present advice. We present the motivation for preference advice and a proof of the correctness of our extension to KBKR. In addition, we show empirical results that our method can make effective use of advice on a novel reinforcement-learning task, based on the RoboCup simulator, which we call *Breakaway*. Our work demonstrates the significant potential of advice-giving techniques for addressing complex reinforcement learning problems, while further demonstrating the use of support-vector regression for reinforcement learning.

Introduction

Advice-taking methods have proven effective in a number of domains for scaling reinforcement learning (RL) to complex problems (Clouse & Utgoff 1992; Lin 1992; Gordon & Subramanian 1994; Maclin & Shavlik 1996; Andre & Russell 2001; Kuhlmann *et al.* 2004). One promising recent method, called Knowledge-Based Kernel Regression (KBKR) (Mangasarian, Shavlik, & Wild 2004), has been successfully applied to reinforcement-learning problems (Maclin *et al.* 2005). Advice in KBKR takes the form of an IF-THEN rule such as:

IF $(dist\_goalcenter \leq 15)$ AND
$(angle\_goalcenter\_you\_goalie \geq 25)$
THEN $Q_{shoot} \geq 1$

This rule indicates that in those states where the distance to the center of the goal is less than or equal to 15m and where the angle from the player to the goal center and goalie is greater than 25^o, the *Q* value of shooting should be greater than 1. But selecting an appropriate value for the THEN part of the rule can be difficult for a human advisor who is not knowledgeable about the use of *Q* values to estimate the value of actions in an RL environment.

A much more natural approach is to allow the human to suggest that one action should be preferred over another, so that he or she does not have to understand *Q* values and how they affect learning:

IF $(dist\_goalcenter \leq 15)$ AND
$(angle\_goalcenter\_you\_goalie \geq 25)$
THEN *PREFER Shoot TO Pass*

We present a new formulation based on KBKR, which we call Preference KBKR (Pref-KBKR), that allows a user to specify this type of advice and refine it with data using support-vector regression. In the next section we review support-vector regression and the KBKR technique. Following that we present the Pref-KBKR algorithm. We then present a novel RL task, based on the RoboCup simulator (Noda *et al.* 1998), that we call *BreakAway* and demonstrate the value of Pref-KBKR on this new testbed.

Knowledge-Based Support-Vector Regression

In this section we present the basics of KBKR. For a more complete discussion, see Mangasarian et al. (2004).

Support-Vector Regression (SVR)

Consider an unknown function $f(x)$ from R^n to R, where x is a vector of numeric features describing a particular instance and $f(x)$ is the value that instance is labeled with. For example, x might describe the soccer field as seen from the point of view of a player, and $f(x)$ the expected Q value of taking the action *Pass*. We first consider a simple linear approximation $f(x) \approx w^T x + b$, where w is a vector of weights on the features of x and b is an offset. All vectors are column vectors unless transposed by ^{T}.

If we are given a training set of states consisting of m input vectors in R^n as rows of the matrix $A \in R^{m \times n}$ with $y \in R^m$ the corresponding vector of desired $f(x)$ values, we can learn a model by finding a solution to the problem

$$Aw + be \approx y \quad (1)$$

where e denotes a vector of m ones. From now on, we shall omit e for clarity, with the understanding that b is always a scalar. Solutions to this problem are ranked by how well they meet some performance criterion, such as minimum error with respect to the y values. In a kernel approach, the weight vector w is replaced with its dual form $A^T\alpha$, which converts Eq. 1 to

$$AA^T\alpha + b \approx y. \quad (2)$$

We can generalize this formulation by replacing the AA^T term with a kernel, $K(A, A^T)$, to produce

$$K(A, A^T)\alpha + b \approx y. \quad (3)$$

To treat this as a linear or quadratic programming problem we simply have to develop a formulation that allows us to indicate the error measure to be minimized in producing a solution. One such formulation, from Mangasarian et al., (2004) is

$$\begin{array}{ll} \min\limits_{(\alpha,b,s)} & ||\alpha||_1 + \nu|b| + C||s||_1 \\ s.t. & -s \leq K(A, A^T)\alpha + b - y \leq s. \end{array} \quad (4)$$

Where $\nu \geq 0$ and $C > 0$ are fixed parameters. In this formulation we use the vector s to measure the error of the solution on each training example, and we penalize inaccuracies in the objective function that is to be minimized. We minimize a weighted sum of the s, α, and b terms (the one-norm, $||\cdot||_1$, computes the sum of absolute values while $|\cdot|$ denotes the absolute value). The penalties on α and b penalize the solution for being more complex. C is a parameter for trading off how inaccurate the solution is (the s term) with how complex the solution is (the α and b terms). The resulting minimization problem is then presented to a linear program solver, which produces an optimal set of α and b values. In this work we also examine the simpler linear regression formulation which can be effective and produce solutions that can be easily understood by the user. This formulation can be written as

$$\begin{array}{ll} \min\limits_{(w,b,s)} & ||w||_1 + \nu|b| + C||s||_1 \\ s.t. & -s \leq Aw + b - y \leq s. \end{array} \quad (5)$$

In our empirical tests we employ the linear formulation in Eq. 5, because linear models are more understandable and scale better to large numbers of training examples. We use tile coding (Sutton & Barto 1998) to produce the non-linearity in our models. The reader should note that using Eq. 5 is not identical to using a linear kernel $K(A, A^T) = AA^T$ in Eq. 4. We use the CPLEX commercial software package to solve our linear program.

Knowledge-Based Kernel Regression

In KBKR, a piece of advice or domain knowledge is represented using the notation

$$Bx \leq d \Longrightarrow f(x) \geq h^T x + \beta. \quad (6)$$

This can be read as

> If certain conditions hold ($Bx \leq d$), the output ($f(x)$) should equal or exceed some linear combination of the inputs ($h^T x$) plus a threshold term (β).

The term $Bx \leq d$ allows the user to specify the region of input space where the advice applies. Each row of matrix B and its corresponding d values represents a constraint in the advice. For example, the user might give as advice the first rule from the Introduction. For this IF-THEN rule, the matrix B and vector d would have two rows. In the first row there would be a 1 in the column for feature $dist\_goalcenter$. The corresponding d row would hold the value 15. In the second row of B there would be a -1 for the column for feature $angle\_goalcenter\_you\_goalie$ and a -25 in the corresponding d row (the values are negated to capture the idea that this condition represents a $\geq$ condition). The vector h would simply be 0 for this rule and the scalar β would be the value 1. This advice would then be used when attempting to learn the Q function corresponding to the action *Shoot*. Using this advice format, a user in a reinforcement-learning task can define a set of states in which the Q value for a specific action should be high (or low).

Mangasarian et al. demonstrate that the advice implication in Eq. 6 can be approximated by the following set of equations having a solution u:

$$\begin{array}{r} K(A, B^T)u + K(A, A^T)\alpha - Ah = 0 \\ -d^T u + b - \beta \geq 0, u \geq 0. \end{array} \quad (7)$$

If we employ *slack variables*[1] in these equations to allow them to be only partially met (if the data indicates the advice may be imperfect), we have the following formulation:

$$\min_{(\alpha,b,s,z,u\geq 0,\zeta\geq 0)} ||\alpha||_1 + \nu|b| + C||s||_1 + \mu_1||z||_1 + \mu_2\zeta \quad (8)$$

$$\begin{array}{ll} \text{s.t.} & -s \leq K(A, A^T)\alpha + b - y \leq s \\ & -z \leq K(A, B^T)u + K(A, A^T)\alpha - Ah \leq z \\ & -d^T u + \zeta \geq \beta - b. \end{array}$$

Here z and ζ are the slacks introduced to allow the formulas in Eq. 7 to be only partially met for this piece of advice. The fixed, positive parameters μ_1 and μ_2 specify how much to penalize these slacks. In other words, these slacks allow the advice to be only partially followed by the learner. Mangasarian et al. (2004) tested their method on some simple regression problems and demonstrated that the resulting solution would incorporate the knowledge. However, the testing was done on small feature spaces and the tested advice placed constraints on all of the input features.

SVR, RL and Preference KBKR

In order to use regression methods for RL we must first formulate the problem as a regression problem. Some previous research (Dietterich & Wang 2001; Lagoudakis & Parr 2003) has employed support-vector approaches for reinforcement learning, but these methods focused on learning a single state-evaluation function and relied on having a world simulator to be able to simulate the effects of actions, a strong assumption. In previous work (Maclin *et al.* 2005)

[1] Which we will call "slacks" for short from now on. Variable s in Eqs. 4 and 5 is another example of a slack.

we investigated using support-vector approaches for reinforcement learning and in particular the difficulties of applying KBKR to RL. In order to learn a Q function without needing a world model, we formulated the learning problem as that of learning a *set* of regression models, one for each action in the environment. We could then employ support-vector regression methods to learn each of these functions individually. This technique is effective and with a number of other extensions and refinements to the KBKR method led to significant gains in performance. But one significant drawback of employing the advice is that the advisor must determine the value the Q function should meet or exceed for a set of states and this can be difficult.

Note that one does not have to learn the models for each action separately; the problem could be formulated as one where the solver had one set of α or w values and a b value for each action and the solve for all of these parameters simultaneously. If we index the set of actions by the set $\{1, \ldots, j\}$ we have the problem

$$\begin{aligned} \min_{(\alpha_a, b_a)} & \sum_{a=1}^{j} (||\alpha_a||_1 + \nu|b_a| + C||s_a||_1) \\ \text{s.t.} \quad & \text{for each action } a \in \{1, \ldots, j\} : \\ & -s_a \leq K(A_a, A_a^T)\alpha_a + b_a - y_a \leq s_a. \end{aligned} \tag{9}$$

Here the α_a and b_a are the parameters for the model for action a. The A_a values are the states where action a was taken and the y_a values are our estimates for the Q values for the action a for each of those states. There are then a set of slacks s_a for each action.

Although this formulation may be appealing in the sense that all of the Q action functions are being solved simultaneously, in practice the individual action optimization problems are independent and can thus be solved separately (which is often a bit more efficient computationally). Thus for standard KBKR we do not employ this approach. However, this formulation does open up the possibility of considering the relative Q values for a pair of actions simultaneously. In particular, this formulation allows us to now represent a novel form of advice where we indicate that under a set of conditions, one action is *preferable* to another. In precise terms, we can represent advice of the form

$$Bx \leq d \Longrightarrow Q_p(x) - Q_n(x) \geq \beta, \tag{10}$$

which can be read as:

> If certain conditions hold ($Bx \leq d$), the value of preferred action p ($Q_p(x)$) should exceed that of non-preferred action n ($Q_n(x)$) by at least β.

This type of advice, where one action is marked as being *preferred* over another is the essence of our new approach, which we call Preference KBKR (Pref-KBKR for short).

Writing Eq. 10 using our models of Eq. 2 we get

$$Bx \leq d \Longrightarrow \alpha_p^T A_p x + b_p - \alpha_n^T A_n x - b_n \geq \beta. \tag{11}$$

In order to incorporate a nonlinear kernel into our prior knowledge formulation, following Mangasarian et al., we assume x can be accurately approximated as a linear combination of the rows of A to get

$$BA^T t \leq d \Longrightarrow \alpha_p^T A_p A^T t + b_p - \alpha_n^T A_n A^T t - b_n \geq \beta. \tag{12}$$

We then kernelize this equation, obtaining

$$\begin{aligned} & K(B, A^T)t \leq d \Longrightarrow \\ & \alpha_p^T K(A_p, A^T)t + b_p - \alpha_n^T K(A_n, A^T)t - b_n \geq \beta. \end{aligned} \tag{13}$$

Finally, we require the kernels in Eq. 13 be symmetric, that is $K(B, A^T)^T = K(A, B^T)$. In order to incorporate Eq. 13 into our optimization problem, we employ the following proposition from the literature.

Proposition: *[(Mangasarian, Shavlik, & Wild 2004), Proposition 3.1] Let the set $\{t|K(B, A^T)t \leq d\}$ be nonempty. Then, for a fixed (α, b, h, β), the implication*

$$K(B, A^T)t \leq d \Longrightarrow \alpha^T K(A, A^T)t + b \geq h^T A^T t + \beta \tag{14}$$

is equivalent to the system of linear equalities in Eq. 7 having a solution u.

Substituting our new formulation for advice from Eq. 13 into this proposition, we are interested in the following corollary.

Corollary: *Let the set $\{t|K(B, A^T)t \leq d\}$ be nonempty. Then, for a fixed $(\alpha_p, b_p, \alpha_n, b_n, \beta)$, the implication in Eq. 13 is equivalent to the following system of linear inequalities having a solution* u

$$\begin{aligned} K(A, B^T)u + K(A, A_p^T)\alpha_p - K(A, A_n^T)\alpha_n = 0 \\ -d^T u + b_p - b_n - \beta \geq 0, u \geq 0. \end{aligned} \tag{15}$$

Proof: The corollary follows directly using appropriate substitutions. For concreteness, we prove the corollary here using the proof of Proposition 2.1 of Mangasarian et al. (2004) as a template.

The implication in Eq. 13 is equivalent to the following system of equations having no solution (t, ζ)

$$\begin{aligned} -\zeta < 0, K(B, A^T)t - d\zeta \leq 0 \\ (\alpha_p^T K(A_p, A^T) - \alpha_n^T K(A_n, A^T))t + (b_p - b_n - \beta)\zeta < 0. \end{aligned} \tag{16}$$

This captures the notion of the implication in Eq. 13 that we cannot come up with a solution that makes the left-hand side of the equation true and the right-hand side false. Using Motzkin's theorem of the alternative (Mangasarian 1994) we know that Eq. 16 being insoluble is equivalent to the following system having a solution (u, η, τ)

$$\begin{aligned} K(A, B^T)u + (K(A, A_p^T)\alpha_p - K(A, A_n^T)\alpha_n)\eta = 0 \\ -d^T u + (b_p - b_n - \beta)\eta - \tau = 0, u \geq 0, 0 \neq (\eta, \tau) \geq 0. \end{aligned} \tag{17}$$

The expression $0 \neq (\eta, \tau) \geq 0$ states that both η and τ are non-negative and at least one is non-zero. If $\eta = 0$ in Eq. 17 then we contradict the nonemptiness of the set $\{t|K(B, A^T)t \leq d\}$ (assumed in the proposition) because for $t \in \{t|K(B, A^T)t \leq d\}$ and (u, τ) that solve this equation with $\eta = 0$, we obtain the contradiction

$$\begin{aligned} 0 \geq u^T(K(B, A^T)t - d) = \\ t^T K(A, B^T)u - d^T u = -d^T u = \tau > 0. \end{aligned} \tag{18}$$

Thus we know that $\eta > 0$ in Eq. 17 and by dividing this equation through by η and redefining (u, α, τ) as $(\frac{u}{\eta}, \frac{\alpha}{\eta}, \frac{\tau}{\eta})$ we obtain the system in Eq. 15. □

Note that we can specify multiple pieces of advice using the above corollary, each with its own B, d, p, n, and β. Indexing the pieces of advice with the set $\{1, \ldots, k\}$ and adding the constraints from Eq. 15 we get a new, Pref-KBKR, formulation of our problem

$$\begin{aligned}
&\min_{(\alpha_a, b_a, s_a, z_i, (\zeta_i, u_i) \geq 0)} \\
&\sum_{a=1}^{j} (||\alpha_a||_1 + \nu|b_a| + C||s_a||_1) + \sum_{i=1}^{k} (\mu_1 ||z_i||_1 + \mu_2 \zeta_i) \\
&\text{s.t.} \quad \text{for each action } a \in \{1, \ldots, j\} : \\
&\qquad -s_a \leq K(A_a, A_a^T)\alpha_a + b_a - y_a \leq s_a \\
&\qquad \text{for each piece of advice } i \in \{1, \ldots, k\} : \\
&-z_i \leq K(A, A_p^T)\alpha_p - K(A, A_n^T)\alpha_n + K(A, B_i^T)u_i \leq z_i \\
&\qquad -d^T u_i + \zeta_i \geq \beta_i - b_p + b_n.
\end{aligned} \tag{19}$$

For readability, we omit the subscripts on p and n. Note that we have added slack variables z_i and ζ_i so that the advice can be satisfied inexactly.

To close this section, we note that in our experiments we introduced nonlinearity through the use of tiling rather than through the use of a nonlinear kernel. Thus, we used the simple form of advice

$$Bx \leq d \Longrightarrow w_p^T x + b_p - w_n^T x - b_n \geq \beta. \tag{20}$$

The above analysis can be repeated to obtain an optimization problem that incorporates the advice of Eq. 20 as

$$\begin{aligned}
&\min_{(w_a, b_a, s_a, z_i, (\zeta_i, u_i) \geq 0)} \\
&\sum_{a=1}^{j} (||w_a||_1 + \nu|b_a| + C||s_a||_1) + \sum_{i=1}^{k} (\mu_1 ||z_i||_1 + \mu_2 \zeta_i) \\
&\text{s.t.} \quad \text{for each action } a \in \{1, \ldots, j\} : \\
&\qquad -s_a \leq A_a w_a + b_a - y_a \leq s_a \\
&\qquad \text{for each piece of advice } i \in \{1, \ldots, k\} : \\
&\qquad -z_i \leq w_p - w_n + B_i^T u_i \leq z_i \\
&\qquad -d^T u_i + \zeta_i \geq \beta_i - b_p + b_n.
\end{aligned} \tag{21}$$

We note that here w_a need not be a linear combination of the rows of A_a. This formulation is different from the formulation of Eq. 19 with the linear kernel $K(A, A^T) = AA^T$.

RoboCup Soccer: The BreakAway Problem

To demonstrate the effectiveness of our new formulation we experimented with a new subtask we have developed based on the RoboCup Soccer simulator (Noda *et al.* 1998). We call this new task *BreakAway*; it is similar in spirit to the *KeepAway* challenge problem of Stone and Sutton (2001).

BreakAway is played at one end of the soccer field, and the objective of the *N attackers* is to score goals against *M defenders*. A BreakAway game ends when a defender takes the ball, the goalie catches the ball, the ball gets kicked out of bounds, a goal gets scored, or the game length reaches 10 seconds. For simplicity, several rules about player behavior near the goal (such as the off-sides rules) are ignored. Figure 1 contains a sample snapshot of the BreakAway game. In our experiments we use this configuration, which contains two attackers, zero defenders, and a goalie.

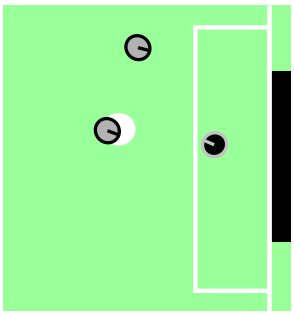

Figure 1: A sample *BreakAway* game where two attackers, represented as light circles with dark edges, are attempting to score on a goalie. The goal is shown in black and the ball is the large white circle.

To simplify the learning task, attackers only learn when they are in control of the ball. Those attackers that do not have the ball follow a hard-wired strategy: they move to intercept the ball if they estimate that they can reach it faster than any other attacker; otherwise, they move to a good shooting position near the goal and wait to receive a pass.

The attacker in possession of the ball has a learnable choice of actions. It may choose to move with the ball, pass the ball to a teammate, or shoot the ball at the goal. We limit the movement actions to four choices: forward toward the center of the goal, directly away from the goal, and right or left along the circle centered at the goal. The shoot action directs the ball at the center, right side, or left side of the goal, whichever path is least blocked by the goalie. Note that all of these high-level actions actually consist of multiple low-level commands in the simulation.

Defenders, if there are any besides the goalie, follow a simple hard-wired strategy. The one that can get to the ball fastest tries to intercept it. If there are others, they move to block the path between the ball and the nearest attacker, to prevent it from receiving a pass.

The goalie tries to pick up the ball if close enough, and otherwise it moves to block the path from the ball to the center of the goal. This is an effective strategy; two attackers who choose among their actions randomly will score against this goalie in less than 3% of their games, even without any non-goalie defenders.

For the case with two attackers and one goalie, a set of 13 features describes the world state from the perspective of the attacker with the ball. These features provide pair-wise distances between players (3 features); distances between the attacker with the ball and the left, center, and right portions of the goal (3 features), distance from the other attacker to the center of the goal (1 feature); the angle formed by the three players (1); the angles between the goalie, the attacker with the ball, and each of the left, center, and right portions of the goal (3); the angle from the top-left corner of the field,

the center of the goal, and the attacker with the ball (1); and the time left in the game (1).

The task is made more complex because the simulator incorporates noise into the players' sensors. In addition, player actions contain a small amount of error. For example, there is a small probability that a player attempting to pass the ball will misdirect it and send it out of bounds.

The attackers receive a reward at the end of the game based on how the game ends. If they score a goal, the reward is +2. If the ball goes out of bounds, the goalie steals it, or time expires, the reward is -1. Missed shots, including those the goalie blocked, lead to rewards of 0 (we chose this neutral value in order to not discourage shooting).

In their work on KeepAway, Stone and Sutton showed that reinforcement learning can be used successfully. They used a tile encoding of the state space, where each feature is discretized several times into a set of overlapping bins. This representation proved very effective in their experiments on this task, and we use tiling in our experiments.

Experimentation

We performed experiments on BreakAway using three learning algorithms: (a) Pref-KBKR, (b) a KBKR learner that only allows advice about individual actions (Eq. 6), and (c) a support-vector reinforcement learner without advice (Eq. 5).

The two attackers pooled their experiences to learn a single shared model. The attackers use that model to choose the predicted best action 97.5% of the time (exploitation) and otherwise randomly choose an action 2.5% (exploration). We set the discount rate to 1 since games are time-limited.

We gave one piece of advice for each of the six possible actions. Advice for when to shoot appears in the Introduction. We advise passing when more than 15m from the goal and one's teammate is closer to the goal but at least 3m from the goalie. We advise moving ahead when more than 15m from the goal but closer than one's teammate. We advise *against* moving away when more than 15m from the goal, we advise *against* moving left when the angle with the top-left corner is less than 20^o and advise *against* moving right when this angle exceeds 160^o. We advise against an action by saying its Q value is lower than a constant, which we determine by computing the mean Q value in the training set.

In our experiments with Pref-KBKR, we use advice that is similar to the advice we used for KBKR, except three preferences are provided. We used the advice about preferring Shoot to Pass shown in the Introduction. We also said that Pass is preferred to Move Ahead when more than 15m from the goal, and one's teammate is closer to the goal but at least 3m from the goalie. Our third piece of relative advice is to prefer Move Ahead over Shoot when more than 15m from the goal and closer than one's teammate to the goal. We did not provide preference advice for the other three move actions and simply used the same advice for these actions as used in the standard KBKR approach.

We set the values of C, ν, μ_1, and μ_2 in our optimization problems (Eq. 21) to 100/#examples, 10, 10, and 100 respectively. (By scaling C by the number of examples, we are penalizing the average error on the training examples, rather than the total error over a varying number of examples.) We tried a small number of settings for C for our non-advice approach (i.e., our main experimental control) and found that 100 worked best. We then used this same value for our KBKR-based algorithms. We simply chose these μ_1 and μ_2 values and have not yet experimented with different settings; 10 and 100 are actually the *initial* values for μ_1 and μ_2 – we scale these by $e^{-game\#/2500}$ so that the penalty for not matching the advice is reduced as a function of the number of games played.

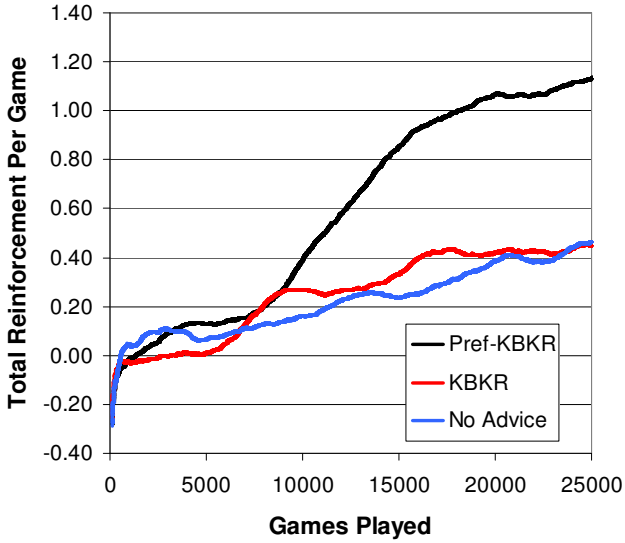

Figure 2: The average total reinforcement as a function of the number of games played for the three algorithms in our experiments.

Since incremental algorithms for support-vector machines are not well developed, we simply retrain models "from scratch" every 100 games, using at most 2000 training examples per action. We choose half of these examples uniformly from all earlier games and half via an exponentially decaying probability according to the example's age, in order to not overfit the most recent experiences while still focusing on recent data. For all three of our approaches, each training cycle we recompute the Q's for the chosen training examples using one-step SARSA learning (Sutton & Barto 1998) obtained via the most recently learned models.

Figure 2 shows the results of our experiments. As a function of the number of games played, we report the average reward over the previous 2500 games. Each curve is the average of ten runs. Our results show that giving preferential advice is advantageous in BreakAway, leading to statistically significant improvements in performance over both KBKR and a no advice approach ($p < 0.001$ in both cases using unpaired *t*-tests on the results at 25,000 games played).

Related Work

Others have proposed mechanisms for providing advice to reinforcement learners. Clouse and Utgoff (1992) developed a technique to allow a human observer to step in and provide advice in the form of single actions to take in a specific state. Lin (1992) "replays" teacher sequences to bias

a learner towards a teacher's performance. Laud and DeJong (2002) developed a method that uses reinforcements to shape the learner. Each of these methods, while providing a mechanism for advice-taking differs significantly from the Pref-KBKR form and usage of the advice.

Several researchers have examined advice-taking methods that use some form of programming language to specify procedural advice. Gordon and Subramanian (1994), developed a method that accepts advice in the form IF *condition* THEN *achieve goals* that operationalizes the advice and then uses genetic algorithms to adjust it with respect to the data. Our work is similar in the form of advice, but we use a significantly different approach (optimization via linear programming) as our means of determining how to incorporate that advice. Maclin and Shavlik (1996) present a language for providing advice to a reinforcement learner that includes simple IF-THEN rules and more complex rules involving recommended action sequences. These rules are inserted into a neural network, which then learns from future experience, whereas KBKR uses support-vector machines. Unlike Pref-KBKR, Maclin and Shavlik's approach did not allow the user to indicate preferences for one action over another. Andre and Russell (2001) describe a language for creating learning agents, but the commands in their language are assumed by their learners to be fully correct, while in KBKR advice is not assumed to be constant and can be refined via slack variables.

Conclusions and Future Work

We have presented a new algorithm, called Preference KBKR (Pref-KBKR), that allows a human user to present advice in a natural form to a reinforcement learner. Advice in Pref-KBKR takes the form of IF-THEN rules, where the IF indicates situations in which the advice applies and the THEN indicates a preference for one action over another. Thus, users present their advice in terms of *policies* – the choice of the action to perform in the current state – rather than giving their advice in terms of Q values, which are an internal detail of the learning algorithm we employ.

Our algorithm is based on Knowledge-Based Kernel Regression (KBKR). We represent the preference advice as additional constraints to a support-vector regression task, which is solved using a linear-program solver. We tested our approach on a new testbed that we only recently developed. It is based on the RoboCup soccer simulator and addresses a subtask of soccer, which we call *BreakAway*. We empirically compare three approaches: a support-vector solver that receives no advice, a KBKR solver that receives advice about desired Q values, and our new Pref-KBKR technique. Our results show that giving advice is advantageous, and that advice about preferred actions can be more effective than advice about Q values.

In our future work, we plan to investigate Pref-KBKR on a wider set of problems and with additional pieces of advice. We also intend to test these systems on a wider variety of parameter settings to see how important the various parameters are to the effectiveness of these methods. Another of our planned experiments is to use kernels (Eq. 19), with and without tiling our features. We also need to address scaling to larger numbers of advice and larger numbers of training examples. We believe that advice-taking methods are critical for scaling reinforcement-learning methods to large problems, and that approaches making use of support-vector techniques will be crucial in this effort.

Acknowledgements

This research was partially supported by DARPA grant HR0011-04-1-0007 and US Naval Research Laboratory grant N00173-04-1-G026. The code for BreakAway can be obtained from ftp://ftp.cs.wisc.edu/machine-learning/shavlik-group/RoboCup/breakaway.

References

Andre, D., and Russell, S. 2001. Programmable reinforcement learning agents. In *NIPS 13*.

Clouse, J., and Utgoff, P. 1992. A teaching method for reinforcement learning. In *ICML'92*.

Dietterich, T., and Wang, X. 2001. Support vectors for reinforcement learning. In *ECML'01*.

Gordon, D., and Subramanian, D. 1994. A multistrategy learning scheme for agent knowledge acquisition. *Informatica* 17:331–346.

Kuhlmann, G.; Stone, P.; Mooney, R.; and Shavlik, J. 2004. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *AAAI'04 Wshp on Supervisory Control of Learning and Adaptive Systems*.

Lagoudakis, M., and Parr, R. 2003. Reinforcement learning as classification: Leveraging modern classifiers. In *ICML'03*.

Laud, A., and DeJong, G. 2002. Reinforcement learning and shaping: Encouraging intended behaviors. In *ICML'02*.

Lin, L. 1992. Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning* 8:293–321.

Maclin, R., and Shavlik, J. 1996. Creating advice-taking reinforcement learners. *Machine Learning* 22:251–281.

Maclin, R.; Shavlik, J.; Torrey, L.; and Walker, T. 2005. Knowledge-based support-vector regression for reinforcement learning. In *IJCAI'05 Wshp on Reasoning, Representation, and Learning in Computer Games*.

Mangasarian, O.; Shavlik, J.; and Wild, E. 2004. Knowledge-based kernel approximation. *Journal of Machine Learning Research* 5:1127–1141.

Mangasarian, O. 1994. *Nonlinear Programming*. Philadelphia, PA: SIAM.

Noda, I.; Matsubara, H.; Hiraki, K.; and Frank, I. 1998. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence* 12:233–250.

Stone, P., and Sutton, R. 2001. Scaling reinforcement learning toward RoboCup soccer. In *ICML'01*.

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