

(the slack measures how far you would have to “move” that point to predict its output correctly). The optimization problem works by trying to minimize the combination of the size of the parameters of our model α , b and the sum of the slacks s indicating how far off our solution is. These components are weighted relatively using a parameter C which the user picks (or is selected via a tuning set) that indicates the trade-off between data fit and model complexity.

In the KBKR method, a rule such as the one discussed in the last section is represented in the following form:

$$Bx \leq d \implies f(x) \geq h^T x + \beta. \quad (2)$$

For the rule from the last section, the first row of B would have a 1 in the column for $dist(goalCenter)$, and the second row would have a -1 in the column for $angle(goalCenter, me, goalie)$ (the rule specifies \geq , so we need to multiply both sides by -1). The d would have the value 15 in the first row and -25 in the second row. The β for this rule would be 0.9 (the value Q_{shoot} is greater than).

Implication (2) cannot be used directly in a support-vector machine. However, by using Motzkin’s theory of the alternative, this implication can be converted to a complicated set of linear constraints (Mangasarian, Shavlik, & Wild 2004). Following standard practice, these constraints are “slacked” to allow for the possibility of the advice not being perfect. This results in the optimization problem:

$$\min_{(\alpha, b, s, z, u \geq 0, \zeta \geq 0)} \|\alpha\|_1 + |b| + C\|s\|_1 + \mu_1\|z\|_1 + \mu_2\zeta \quad (3)$$

$$\begin{aligned} \text{s.t.} \quad & -s \leq K(A, A^T)\alpha + b - y \leq s \\ & \text{for each piece of advice } i \in \{1, \dots, k\} : \\ & -z \leq K(A, B_i^T)u + K(A, A^T)\alpha - Ah \leq z \\ & -d^T u + \zeta \geq \beta_i - b. \end{aligned}$$

In essence, KBKR introduces a set of variables u defining a new space where there is a solution to u only if there is no solution in the original space that would allow the rule to be violated. The slacks that are introduced to this solution (the variables z and ζ) allow the system to find a solution u that only partially meets the constraints.

A major advantage of this approach is that the solution produced by the system will match the advice even if there is no training data that matches the preconditions of the advice, which allows for rapid learning from very few input-output pairs that are the typical source of training information. But this approach also has its limitations. One limitation is that this approach introduces not only a large number of new constraints to the problem (depending on the size and number of the advice rules), but also introduces a number of new variables u , z , ζ into the system of equations that must be solved for (thereby increasing solution time). In addition, the overall solution is fairly complex because the u variables are not directly part of the solution, but are used to indirectly constrain the variables of the solution α and b .

A second limitation of this approach is the interpretation of the slacked advice variables (z and ζ) following training. These slacks allow the learned model to partially contradict the advice, which is necessary since one cannot assume the user-provided advice is perfectly correct. However,

the slacks are associated with how the α variables and the u variables combine. This approach encourages certain types of changes to the advice to account for noise; specifically that the slacks are a measure of how much the preconditions of an advice rule need to be rotated (z) and translated (ζ) in feature space in order for the advice to be consistent with the learned model (recall that the slack s on a standard training point is the amount it needs to be translated in feature space). But these types of advice-refinement do not always match how one would like advice to be refined, such as scaling the polygon representing a rule’s preconditions.

Finally, there is the problematic aspect of the kernelization of the B matrix – the value $K(B, A^T)$. The assumption in KBKR and KBSVM is that the kernelized matrix representing the constraints will match the original constraints in the new space (the antecedents to rules will be altered appropriately when kernelized). This limitation led to Mangasarian and Wild (2005) introducing a method that does not have this weakness. However, their proposed method is fairly complex, and requires extensive sampling of feature space, making scalability a significant question.

To address some of these limitations we propose a simpler family of methods, which we call ExtenKBKR and ExtenKBSVM, for Extensional Knowledge-Based Kernel Regression and Support Vector Machines. We chose the term “extensional” because our approaches represent the polyhedral region of a rule’s preconditions via a sample of the data points included within it, whereas prior work has directly manipulated the “intensional” representation of the advice.

Extensional Knowledge-Based Methods

As noted previously, the KBKR and KBSVM methods are effective, but have limitations, notably that the mechanism they use to account for imperfect advice is fairly complex. Our new approach is simpler and involves a more appealing mechanism for modifying imperfect advice. In our extensional methods we determine which training-set points (if any) match each piece of advice and add constraints that state the advice constraint should be met for those points.

To better understand our approach, consider Figure 1(a), which shows examples for a binary classification problem. The advice region is a rectangle in feature space; assume the advice says that points in this region should belong to the solid-circle class. Panel (b) shows the decision boundary a standard SVM might produce. Note there is a penalty, proportional to the distance to the decision boundary, for the misclassified example, illustrated by the dashed line.

Panel (c) illustrates how the advice might be slacked in Fung et al.’s KBSVM, by rotation and translation, so that the advice is consistent with the training data and the decision boundary. Notice the decision boundary has moved slightly and that the penalty for the misclassified example remains.

Panel (d) shows what our ExtenKBSVM might do with this task. It pays a penalty for the white circle inside the advice because the advice says this point should be a black circle, but the decision boundary classifies it as a white one. (As will be seen later – in Equation 8 – the point mispredicted by the advice is involved in two constraints in ExtenKBSVM’s

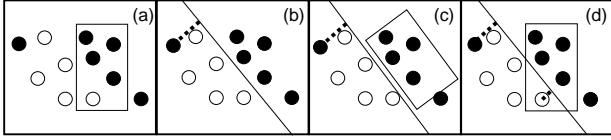


Figure 1: Standard SVMs (panel b) compared to KBSVM (panel c) and ExtenKBSVM (panel d).

linear program; one from the advice and one from treating this example as a standard input-output pair.)

Returning now to regression, our ExtenKBKR version of Equation 3 is:

$$\begin{aligned} \min_{(\alpha, b, s, m \geq 0)} \quad & \|\alpha\|_1 + |b| + C\|s\|_1 + \sum_{i=1}^k \left(\frac{\psi}{|M_i|} \|m\|_1 \right) \\ \text{s.t.} \quad & -s \leq K(A, (A \cup U)^T) \alpha + b - y \leq s \\ & \text{for each piece of advice } i \in \{1, \dots, k\}: \\ & K(M_i, (A \cup U)^T) \alpha + b + m \geq \beta_i. \end{aligned} \quad (4)$$

In this formulation M_i is the set of points that match advice item i . Note that our kernel is taken with respect to the set of points $A \cup U$, where U is a set of unlabeled points. Our method introduces one new parameter ψ which is used to determine the total weighting in the cost formula of the slacks for those points covered by advice (recall that the original KBKR representation had two parameters, μ_1 and μ_2 , that weight the slacks associated with the advice).

Notice that labeled training examples can appear in the linear program multiple times, once in the standard way of fitting the training example's input-output pair, and whenever the example matches some advice rule.

One major advantage of this approach is that advice can be applied not only to points from A , but can be used to derive constraints from unlabeled points U . These unlabeled points may exist for the domain at hand or could be artificially generated. These new advice-induced constraints directly constrain the solution variables α, b of the problem.

Another advantage is that if there are exceptions to advice, they can be captured by training examples. In our approach, if a datapoint does not match the advice, we simply slack the advice-constraints related to that training example and pay a penalty in our cost function. In addition, our approach only introduces new slack variables to the problem and does not introduce an entirely new space (i.e., the u 's of Expression 3) that must be solved to derive a solution.

As mentioned previously, Maclin et al. (2005) introduced preference advice to allow a user to indicate when the value of one function should be higher than another when learning both simultaneously. This is useful in problem domains such as reinforcement learning (RL), to say that in a particular situation one action is preferable to another. The Pref-KBKR formulation for both preference advice and the original form of advice (which we call absolute advice) is captured in the following (from Maclin et al. 2005):

$$\begin{aligned} \min_{(\alpha, b, s, z_i, (\zeta_i, u_i) \geq 0)} \quad & \sum_{a=1}^j (\|\alpha_a\|_1 + |b_a| + C\|s_a\|_1) + \\ & \sum_{i=1}^k (\mu_1 \|z_i\|_1 + \mu_2 \zeta_i) \end{aligned} \quad (5)$$

$$\begin{aligned} \text{s.t.} \quad & \text{for each action } a \in \{1, \dots, j\}: \\ & -s_a \leq K(A_a, A_a^T) \alpha_a + b_a - y_a \leq s_a \\ & \text{for each preference advice } i \in \{1, \dots, k\}: \\ & -z_i \leq K(A, A^T) \alpha_p - K(A, A^T) \alpha_n + K(A, B_i^T) u_i \leq z_i \\ & -d^T u_i + \zeta_i \geq \beta_i - b_p + b_n \\ & \text{for each absolute advice } i \in \{1, \dots, k\}: \\ & -z_i \leq K(A, A^T) \alpha_a + K(A, B_i^T) u_i \leq z_i \\ & -d^T u_i + \zeta_i \geq \beta_i - b_a. \end{aligned}$$

In this formulation, models are learned for actions $1 \dots j$ simultaneously. The subscripts p and n refer to the preferred and non-preferred action in preference advice, and a refers to the action for absolute advice.

We can easily formulate an ExtenKBKR version using the same pieces of advice:

$$\begin{aligned} \min_{(\alpha, b, s, m \geq 0)} \quad & \sum_{a=1}^j (\|\alpha_a\|_1 + |b_a| + C\|s_a\|_1) + \\ & \sum_{i=1}^k \left(\frac{\psi}{|M_i|} \|m\|_1 \right) \end{aligned} \quad (6)$$

$$\begin{aligned} \text{s.t.} \quad & \text{for each action } a \in \{1, \dots, j\}: \\ & -s_a \leq K(A_a, (A \cup U)^T) \alpha_a + b_a - y_a \leq s_a \\ & \text{for each piece of prefer advice } i \in \{1, \dots, k\}: \\ & K(M_i, (A \cup U)^T) \alpha_p + b_p - K(M_i, (A \cup U)^T) \alpha_n \\ & \quad - b_n + m \geq \beta_i \\ & \text{for each piece of absolute advice } i \in \{1, \dots, k\}: \\ & K(M_i, (A \cup U)^T) \alpha_a + b_a + m \geq \beta_i. \end{aligned}$$

And these ideas can be used for a linear formulation such as Maclin et al. (2005) found effective for RL:

$$\begin{aligned} \min_{(\alpha, b, s, m \geq 0)} \quad & \sum_{a=1}^j (\|\alpha_a\|_1 + |b_a| + C\|s_a\|_1) + \\ & \sum_{i=1}^k \left(\frac{\psi}{|M_i|} \|m\|_1 \right) \end{aligned} \quad (7)$$

$$\begin{aligned} \text{s.t.} \quad & \text{for each action } a \in \{1, \dots, j\}: \\ & -s_a \leq A w_a + b_a - y_a \leq s_a \\ & \text{for each piece of prefer advice } i \in \{1, \dots, k\}: \\ & M_i w_p + b_p - M_i w_n - b_n + m \geq \beta_i \\ & \text{for each piece of absolute advice } i \in \{1, \dots, k\}: \\ & M_i w_a + b_a + m \geq \beta_i. \end{aligned}$$

We can also adapt ExtenKBKR quickly and easily to other forms of advice.

The extensional notion can also be applied to binary classification problems. The ExtenKBSVM formulation is:

$$\begin{aligned} \min_{(\alpha, b, (s, m \geq 0))} \quad & \|\alpha\|_1 + |b| + C\|s\|_1 + \sum_{i=1}^k \left(\frac{\psi}{|M_i|} \|m\|_1 \right) \\ \text{s.t.} \quad & y(K(A, (A \cup U)^T) \alpha + b) + s \geq 0 \\ & \text{for each piece of advice } i \in \{1, \dots, k\}: \\ & \beta_i (K(M_i, (A \cup U)^T) \alpha + b) + m \geq 0. \end{aligned} \quad (8)$$

In this formulation a piece of advice indicates that when $Bx \leq d$, then the predicted class should be β (1 or -1).

To compare the efficiencies of the old method approach and our new approach, we can look at the size of the resulting optimization problem. For KBKR, each piece of advice introduces one new variable for each data point (the z terms) plus one extra variable (the ζ term), for a total of $E + 1$, where E is the number of training examples.

For our ExtenKBKR, each piece of advice introduces one new variable for each example that matches the advice (which we refer to as M_k , the number of examples that match advice item k). In general one would expect M_k to be much less than $E + 1$; in addition, one can limit the size of M_k and only use a random subset of the examples that match a rule’s preconditions (e.g., in our RoboCup experiments described later, we restricted M_k to be at most 100).

In terms of non-zero terms in the constraint matrix, if there are no unlabeled data points used, then in KBKR each piece of preference advice introduces on the order of $2E^2$ nonzero terms, and each piece of absolute advice introduces on the order of E^2 new terms. The ExtenKBKR introduces on the order of $2E * M_k$ nonzero terms for each piece of preference advice and $E * M_k$ for each piece of absolute advice. Again one would expect that since $M_k \ll E$, ExtenKBKR would have many fewer such terms. Note that in the case where unlabeled data points are used in ExtenKBKR the size of the basis for the kernel would grow, though this would likely still leave ExtenKBKR with many fewer terms.

In the next few sections we report some experiments on synthetic data where we know the underlying function, to demonstrate that the ExtenKBKR method works as expected, and present some results on a difficult problem, a subtask of soccer within the RoboCup simulator.

Synthetic Experiments

In order to initially validate our method we performed a number of simple experiments on synthetically generated data, one of which we present here. Standard Support Vector Regression (SVR) involves one parameter, C , the relative weight of data fit compared to model complexity. ExtenKBKR has a second parameter, ψ (Expression 4), which is the relative weight of the fit to the prior knowledge. KBKR has two parameters in addition to C . These are μ_1 and μ_2 (Expression 3), which measure fit to the prior knowledge.

We selected good parameter values by running some “tuning-set” experiments. We considered $C \in \{10, 100, 250, 500\}$, $\mu_1 \in \{1, 5, 10, 20, 50, 100\}$ and $\mu_2 \in \{1, 5, 10, 20, 50, 100\}$, and for ExtenKBKR we considered $\psi \in \{5, 10, 20, 50\}$. We found that for a wide range of experiments, C of 100, μ_1 and μ_2 of 1 and ψ of 50 worked at or near optimal. Following parameter tuning, we

Table 1: Advice used in the synthetic dataset experiment.

```

IF  $x_1 \geq .7 \wedge x_2 \geq .7 \wedge x_3 \geq .7 \wedge x_4 \geq .7$  THEN  $f_1(x) \geq 4$ 
IF  $x_5 \geq .7 \wedge x_2 \leq .3 \wedge x_6 \geq .7 \wedge x_4 \leq .3$  THEN  $f_2(x) \geq 5$ 
IF  $x_5 \geq .6 \wedge x_6 \leq .6$  THEN PREFER  $f_2(x)$  TO  $f_1(x)$  BY .1
IF  $x_5 \leq .3 \wedge x_6 \leq .3$  THEN PREFER  $f_1(x)$  TO  $f_2(x)$  BY .1
IF  $x_2 \geq .7 \wedge x_4 \geq .7$  THEN PREFER  $f_1(x)$  TO  $f_2(x)$  BY .1
IF  $x_2 \leq .3 \wedge x_4 \leq .3$  THEN PREFER  $f_2(x)$  TO  $f_1(x)$  BY .1

```

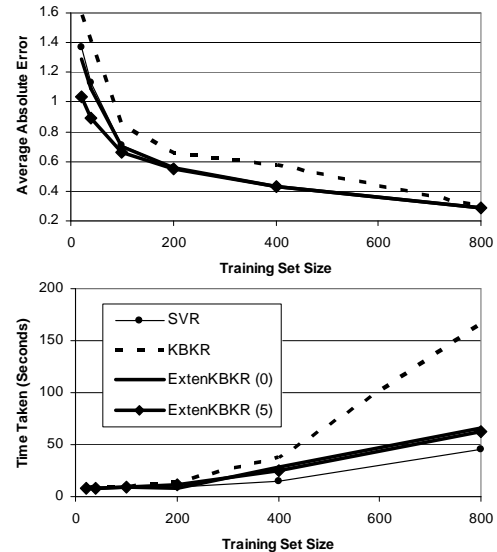


Figure 2: The top graph shows average absolute error for SVR, KBKR, ExtenKBKR and ExtenKBKR with 5 pseudo examples available per advice item. The bottom graph show the average time to solve the linear program.

ran a fresh set of experiments and report those results.

Our synthetic data involves 10 input features, with each feature having a random value between 0 and 1. In this test we create two functions, so that we could test preference advice. The two functions were $f_1(x) = 20x_1x_2x_3x_4 - 1.25$ and $f_2(x) = 5x_5 - 5x_2 + 3x_6 - 2x_4 - 0.5$. We selected these functions since one is highly nonlinear, the other linear, and there are some overlap and significant differences to the functions, allowing for advice. We use a Gaussian kernel with a variance of one. Table 1 shows our advice.

Our tests involve ten repeated experiments, each for dataset sizes of 20, 40, 100, 200, 400, and 800.

In addition, we test ExtenKBKR not only in the regular situation, but where we provided up to five randomly generated pseudo examples per piece of advice (pseudo examples are used only when less than five training data match a given piece of advice). Recall that ExtenKBKR is able to use unlabeled, or in this case, arbitrary random examples, as its sample of the extension of the advice’s preconditions. A graph of the average absolute error on a separate set of 500 test points is shown in Figure 2.

The top graph in Figure 2 shows that ExtenKBKR can outperform KBKR and the no-advice approach, especially when a small sample of unlabeled (pseudo) examples is used when there are only a few labeled training examples. In the top graph KBKR initially performs poorly because the advice is useful but not perfect and the training set sizes are very small. The lower graph demonstrates that ExtenKBKR is more efficient computationally than KBKR.

Experiments in RoboCup Soccer

To further demonstrate the effectiveness of our new formulation, we experimented with a task in the RoboCup soccer simulator called *BreakAway* by Torrey et al. (2005).

In M -on- N BreakAway (see Figure 3), the objective of

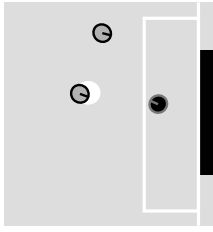


Figure 3: A sample *BreakAway* game where two attackers, represented as light circles with dark edges, are attempting to score on a goalie. The goal is shown in black and the ball is the large white circle.

the M reinforcement learners called *attackers* is to score a goal against $N - 1$ hand-coded *defenders* and a *goalie*. The game ends when they succeed, when an opponent takes the ball, when the ball goes out of bounds, or after a time limit of 10 seconds. Our experiments use 2-on-1 BreakAway.

The attacker who has the ball may choose to move with the ball, pass to a teammate, or try to score a goal by shooting. We limit movement directions to *ahead*, *away*, *left*, and *right*, with the goal as the point of reference, and we limit shots to the left side, right side, or center of the goal. Attackers not in possession of the ball follow a hand-coded strategy to position themselves to receive a pass. The goalie and defenders also follow hand-coded strategies to prevent attackers from scoring and gain possession of the ball.

We adopted the state representation of Torrey et al., which consists of distances and angles between players, distances and angles involving the goal, and the time left in the game. The learners receive a +1 reward at the end of the game if they score a goal, and a 0 reward otherwise.

All features are discretized into intervals called *tiles*, each of which is associated with a Boolean feature. A tile feature is true when the numeric value falls into its interval and false otherwise. This enhancement of the state space was used in RoboCup by (Stone & Sutton 2001), and we adopted it to give our linear Q -function model the ability to represent more complex functions. We create 64 tiles, of varying width, for each feature.

We give advice after playing 100 games using randomly chosen moves. After every 100 games, we retrain the models, and all three variants use the same TD(λ) Q -learning algorithm with SARSA estimates (Sutton & Barto 1998).

We first ran short parameter-tuning experiments where five times we played 1000 games for each of the settings described below. For SVR, KBKR, and ExtenKBKR we considered $C \in \{150, 500, 1500, 5000\}$. For ExtenKBKR, we considered $\psi \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$. For KBKR, we considered $\mu_1 \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$, while always setting $\mu_2 = 100\mu_1$. The parameter settings that worked best in the parameter-tuning experiments are: for SVR, $C = 1500$; for KBKR, $C = 500, \mu_1 = 10^5$; for ExtenKBKR, $C = 500$ and $\psi = 10^5$.

In addition, in all runs, we exponentially increased C and exponentially decreased μ_1, μ_2 , and ψ as the number of games increased, since the value of prior knowledge becomes less important as the amount of actual experience increases. The C values above are the asymptotic values; we

Table 2: Advice used in our RoboCup experiments.

IF	$distanceTo(goalCenter) \geq 15$
THEN	PREFER move(ahead) TO <i>other actions</i> BY .01
IF	$distanceTo(goalLeft) \leq 14.9 \wedge$ $distanceTo(goalCenter) \leq 14.9 \wedge$ $(distanceBetween(teammate, goalie) -$ $distanceTo(goalie)) \geq 3 \wedge$ $distanceTo(teammate) \geq 10 \wedge$ $angle(teammate, me, dgoalie) \geq 25 \wedge$ $angle(topRight, goalCenter, me) \geq 126$
THEN	PREFER passTo(teammate) TO <i>other actions</i> BY .01
IF	$distanceTo(goalCenter) \leq 9.9 \wedge$ $angle(goalLeft, me, goalie) \geq 40 \wedge$ $angle(topRight, goalCenter, me) \leq 80$
THEN	PREFER shoot(left) TO <i>other actions</i> BY .01

started at $C = 0.1C_{asympt}$ and reached half the asymptotic value at 5000 games. The μ_1, μ_2 , and ψ are the *initial* values and their decay rate was $e^{(1-\#games/100)}$. We did not tune the slopes of these exponential curves.

Once we selected parameter settings via the process described above, we performed 10 repeated trials, each starting fresh at game #1. We ensure that each advice rule covers at least 10 and at most 100 examples. If a rule covers too few real examples, we sample the feature space uniformly to create random unlabeled “pseudo-examples.” If a rule covers too many real examples, we randomly discard the excess. In terms of the number of pseudo examples, ExtenKBKR adds 19 examples to the 300 or so examples for the leftmost point in Figure 4. In terms of efficiency, we found that ExtenKBKR executes in approximately half the CPU time.

Table 2 contains the advice we used. Basically it says that the player with the ball should move in when far from the goal, that it should pass to its teammate if that player seems to have a good shot, and it should shoot if close in and has a good shooting angle. In addition, the second advice rule was designed to have the player with the ball pass when in the lower-right corner of the field, thereby luring the goalie to the right edge of the goal and opening a shooting opportunity for the teammate. We did not manually alter this advice, which is imperfect, during the course of our experiments.

Figure 4 shows the probability of scoring a goal averaged over sets of 100 games, using the three approaches. ExtenKBKR is statistically-significantly better (at the 95% confidence level) than no-advice Support Vector Regression at 1000, 2000, 3000, 4000, and 5000 games, based on an unpaired, two-tailed t -test. Also based on that test, KBKR is statistically-significantly better than SVR at 1000, 2000, and 3000 games. Hence our second experiment demonstrates that advice-taking can significantly improve reinforcement learning in a challenging task, and that KBKR and the faster ExtenKBKR perform about equally accurately.

Related Work

As discussed earlier, our work closely relates to the techniques KBKR, KBSVM, and Pref-KBKR. In addition there have been a number of papers that have examined the use of prior knowledge with kernel methods. Schoelkopf et al.

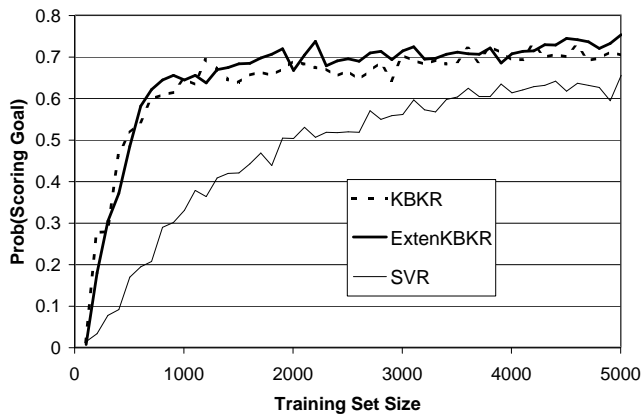


Figure 4: Probability of scoring a goal over the past 100 BreakAway games, for standard support vector regression (SVR), KBKR, and ExtenKBKR.

(1998) looked at incorporating information about images and class invariants (e.g., local translations in the image) and information about local structure in images. Epshteyn and DeJong (2005) proposed a method for incorporating prior knowledge that allows a user to specify excitatory and inhibitory conditions between examples (e.g., the presence of the word “medical” should have a positive connection on the word “doctor”). Our work focuses on advice expressed as rules, providing a more general platform for incorporating advice than Schoelkopf et al. and Epshteyn & DeJong. Sun and DeJong (2005) use domain knowledge to select important features of examples and create generalized examples that are used in addition to the standard examples in an SVM. In our approach we use advice to guide the selection of important examples and to associate desired outputs with these examples. Muggleton et al. (2005) developed a method for incorporating prior knowledge using inductive logic programming and support vector methods. Our method differs in that our approach applies to regression as well as classification, and our method explicitly addresses imperfect prior knowledge.

In addition, a number of semi-supervised support vector methods have been developed that relate to our work. Wu and Srihari (2004) developed an SVM method that can use a rule for labeling unlabeled data and then provide a confidence value on the predictions for both labeled and unlabeled data. Franz et al. (2004) perform regularization of the solution to a SVR problem using unlabeled data. We believe that our advice rules provide a more informative source of training than these other approaches.

Conclusions

We have presented a family of methods for incorporating advice represented as rules into support-vector learners for both regression and classification. Our methods, which we call ExtenKBKR and ExtenSVM, depend on an extensional definition of advice, where the information about the advice is given to the learner as a set of sample points meeting the preconditions of the advice. Our algorithm then simply adds, to the linear program being solved, the constraints the advice places on these points. This contrasts with the KBKR and

KBSVM methods that include advice as constraints on the solution to the problem in an intensional manner.

In our experiments we demonstrate that our ExtenKBKR performs similarly empirically to KBKR, both for a synthetic dataset and on a challenging subtask in simulated robotic soccer. We also argue that our formulation is both simpler to define and typically leads to smaller optimization problems that can be solved more efficiently.

One of the significant advantages of our approach is it can be adapted to almost any rule-based advice; all that is needed is a test to see if a datapoint matches the advice. In future work we plan to look at a broader range of advice ideas, especially with respect to RL methods, where advice seems to be an especially appropriate learning mechanism.

Acknowledgements

This research was partially supported by US Naval Research Laboratory grant N00173-06-1-G002 (to RM) and DARPA grant HR0011-04-1-0007 (to JS).

References

- Epshteyn, A., and DeJong, G. 2005. Rotational prior knowledge for SVMs. In *ECML*.
- Franz, M.; Kwon, Y.; Rasmussen, C.; and Schoelkopf, B. 2004. Semi-supervised kernel regression using whitened function classes. In *DAGM*.
- Fung, G.; Mangasarian, O.; and Shavlik, J. 2002. Knowledge-based SVM classifiers. In *NIPS*.
- Fung, G.; Mangasarian, O.; and Shavlik, J. 2003. Knowledge-based nonlinear kernel classifiers. In *COLT*.
- Maclin, R.; Shavlik, J.; Torrey, L.; Walker, T.; and Wild, E. 2005. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *AAAI*.
- Mangasarian, O., and Wild, E. 2005. Nonlinear knowledge in kernel approximation. Technical Report 05-05, UW CS.
- Mangasarian, O.; Shavlik, J.; and Wild, E. 2004. Knowledge-based kernel approximation. *JMLR* 5.
- Muggleton, S.; Lodhi, H.; Amini, A.; and Sternberg, M. 2005. Support vector inductive logic programming. In *Proc. of 8th Int. Conf. on Discovery Science, LNAI 3735*.
- Schoelkopf, B.; Simard, P.; Smola, A.; and Vapnik, V. 1998. Prior knowledge in support vector kernels. In *NIPS*.
- Stone, P., and Sutton, R. 2001. Scaling reinforcement learning toward RoboCup soccer. In *ICML*.
- Sun, Q., and DeJong, G. 2005. Explanation-augmented SVMs. In *ICML*.
- Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.
- Torrey, L.; Walker, T.; Shavlik, J.; and Maclin, R. 2005. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *ECML*.
- Wu, X., and Srihari, R. 2004. Incorporating prior knowledge with weighted margin SVMs. In *KDD*.