Learning to Predict Action Outcomes in Continuous, Relational Environments

Thomas J. Palmer, Matthew Bodenhamer, Andrew H. Fagg
Symbiotic Computing Laboratory, University of Oklahoma, USA
tompalmer@ou.edu, mbodenhamer@ou.edu, fagg@cs.ou.edu

***Abstract*—We present a method for predicting action outcomes in unstructured environments with variable numbers of participants and hidden relationships between them. For example, when pouring flour from a cup into a mixing bowl, important relations must exist between the cup and the bowl. The action $Pour(x, y)$ might depend on the precondition $Above(x, y)$. How well the predicate $Above$ actually predicts action success often depends on complicated world dynamics and perhaps other objects in the scene. While such predicates are commonly hand-crafted, we present in this paper a method for learning these predicates directly from the continuous data. In this manner, an agent's own developmental experience can drive its world representations. Here, we learn such representations as ensembles (or *forests*) of probability trees using the Spatiotemporal Multidimensional Relational Framework (SMRF). By reasoning about individual objects, SMRF trees allow us to focus attention on action parameters while still considering other objects in a scene. We demonstrate our method on three simulated problems. Two are in a blocks world with gravity: one predicting the tipping direction of a balance scale, inspired by Robert Siegler's classic cognitive psychology work, and the second predicting the success of dropping one block on another. The third problem predicts the success of passing a ball in a soccer domain. In these tasks, we show an ability to scale prediction to scenes with more objects than are present in the training data.**

I. Introduction and Related Work

The real world consists of far more variables than an intelligent agent can possibly model. Therefore, some form of abstract representation is vital. One approach is to represent an environment as a set of objects, each with its own attributes. This still leads to a vast space for consideration, especially if relationships between objects are important. Furthermore, on different occasions, certain roles might be played by different individual objects. A developing child or robot cannot expect to optimize decisions in the full space. Selective attention and higher-level concepts provide potential building blocks for performing tasks in a complex world.

For example, a robot working in an unstructured kitchen needs to avoid obstacles, retrieve items, open containers, and combine and mix ingredients, among other activities. In these tasks, each action involves only a subset of the objects in the environment. When pouring flour into a bowl, the relative positions of the bowl and measuring cup are important. The pouring edge should be approximately centered above the bowl, although not *too* high. Other nearby objects might also interfere, but the exact locations of items on the spice shelf are probably irrelevant. Some attributes of key objects also are likely irrelevant, such as the color of the bowl. For the objects and attributes that do matter, rather complicated world dynamics are at play, including gravity and the dispersion of flour in the air. The robot's own developmental experience can help it to form the concepts needed to successfully perform tasks such as these.

Learning world dynamics can mean the ability to predict any aspect of the future world, whether that be full world state, a subset of state variables, or expected reward. In our present work, we are primarily concerned with predicting the success or failure of agent actions, as this ability can form the component of a higher-level planning or learning system. As an example of such a planner, Stulp et al. [1] learn position boundaries that predict success or failure of actions such as robotic grasping of objects on a table. Identification of probable action success can also be viewed as a form of affordance [2]. Stulp et al., however, do not address the learning of relations for arbitrary numbers of objects, except in certain *ad hoc* fashions. Many other recent works also address learning consequences of agent actions in continuous, multidimensional domains, often predicting detailed world state (e.g., [3], [4], [5]). Again, however, such works commonly avoid arbitrary object relationships.

For addressing relational concerns, logical formulations are a common and traditional technique. Such representations are frequently purely discrete or, perhaps, include one-dimensional continuous attributes. For example, a predicate for $Above(x, y)$ might help in predicting the outcome of agent actions such as $Pour(x, y)$. Learning in such domains is also well established [6], [7], [8]. This includes the general field of relational reinforcement learning (RRL) which adapts reinforcement learning (RL) techniques to such relational, logical domains and representations [9]. However, when applied to multidimensional continuous attributes, a majority of these approaches depend on hand-crafted predicates. This requires manual effort and may fail to consider the specific requirements of such concepts in a highly detailed and unstructured world.

Some approaches combine continuous and relational learning of world dynamics. These include Verbancsics and Stanley [10], who evolve neural networks capable of representing physical relationships using a grid-based world layout. With this representation, among other test domains, they learn to play a 2D simulated soccer keepaway game [11], scaling to more players in test than in training, even before additional learning. Another example of combined continuous and relational dynamics learning is that of Xu and Baird [12], who

learn separate continuous and relational models using a form of case-based reasoning and allow interaction between the two models using human-defined predicates.

In our work, we address relational learning in multidimensional continuous spaces using the Spatiotemporal Multidimensional Relational Framework (SMRF) of Bodenhamer et al. [13], [14]. Given a set of training scenes, each labeled as containing some target concept or not, the SMRF algorithm learns a probabilistic decision tree that identifies the key objects in the scenes and the multidimensional continuous attributes of the objects and relations between them. The goal is to discover what is in common about each scene containing the target concept, but does not occur in the negative scenes. We use this capability here to determine the probability of binary outcomes of world dynamics, including those of parameterized actions.

Unlike many relational decision tree approaches, SMRF also includes the ability to explicitly reason about specific object instances. This provides SMRF additional expressiveness. Here, we further use this capability to focus attention on action parameters. In the $Pour(x, y)$ example, x and y identify certain participant objects in each use (or potential use) of this action. Another key aspect of our work is that the SMRF learning algorithm is stochastic. Repeated training runs produce a variety of trees. Such variety can increase generalization accuracy [15]. Given an ensemble (or *forest*) of SMRF trees, we apply the forest to new scenes to predict binary outcomes. In this way, the trees form a basis for representing the world and its dynamics.

In the remainder of this paper, we describe our method, including the SMRF algorithm and our application of it for world outcome prediction. We then show results for three different problems in 2D simulated physical worlds.

II. Method

A. Overview

We present a method for predicting binary outcomes of actions in continuous relational worlds, where the relations themselves are implicit in the continuous data. The core of our approach is based on Spatiotemporal Multidimensional Relational Framework (SMRF) trees [13], [14], and we begin with an overview of this algorithm. We then address the use of action parameters within the SMRF framework and how a forest of SMRF trees forms a basis to learn outcome prediction in new situations.

B. SMRF Trees

SMRF trees assess the probability that a given collection (or *scene*) of objects contains an instance of a particular *target concept*. As shown in Fig. 1, a SMRF tree is composed of several node types. *Instantiation nodes*, shown as parallelograms, bind an object from a scene to a variable. An ordered sequence of such bindings is called an *instantiation sequence*. *Question nodes*, shown as ellipses, sort instantiation sequences down various branches of the tree, based on the attributes of the objects in that sequence. *Leaf nodes*, shown as rectangles, give the probability of membership in the target concept for instantiation sequences arriving there. The probability that a scene contains an instance of the target concept is the highest probability assigned to any instantiation sequence from the scene. In this way, SMRF trees resemble existentially quantified logical expressions. For example, the SMRF tree in Fig. 1, given some probability decision threshold, is equivalent to the following first-order logic expression:

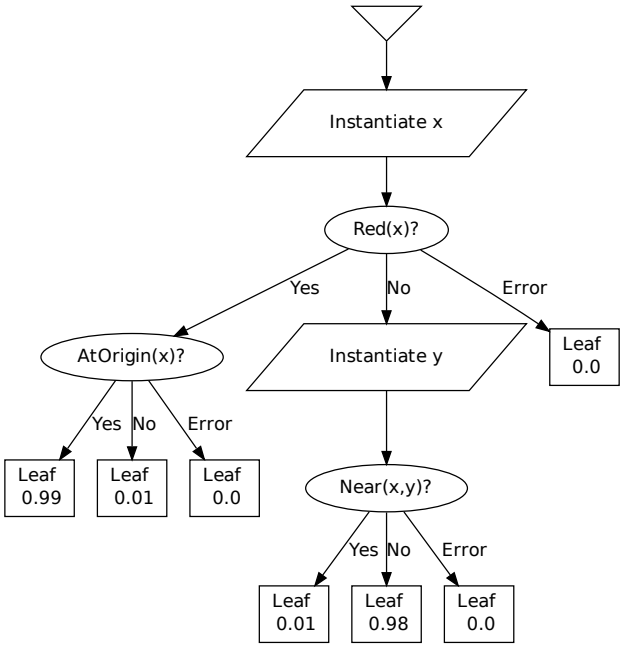

Fig. 1. Hand-crafted SMRF tree encoding a disjunctive concept that identifies either a red object at the origin, or an object that is not red and not near to another object. Predicates such as Red and $Near$ are defined in terms of decision volumes in a metric space.

$$\exists_x \big[(Red(x) \quad \wedge \quad AtOrigin(x)) \vee (\neg Red(x) \quad \wedge \quad \exists_y \left[\neg Near(x, y) \wedge x \neq y \right]) \big].$$

In SMRF, each object has attributes that can be multidimensional and continuous. Relations are implied as functions of attribute values. For example, objects may include a *position* attribute, as defined within some fixed coordinate frame. In addition, the position of one object may be described *relative* to that of another. Each object also has a distinct identity, and subsequent instantiations down a branch of a tree do not repeat objects earlier in the sequence (hence, $x \neq y$ in the example). Sequential instantiations produce permutations of the objects, pruned by question nodes between them. Often, only a small subset of earlier instantiation sequences filter down to branches with additional instantiation nodes.

The SMRF approach allows complex question node models to be defined in terms of the multidimensional attributes of the objects in the training data. While a decision surface in a multidimensional space can be represented in a variety of ways, we choose to define them in terms of three components:

- a *mapping function*, ϕ, which computes some quantity over some subset of the attributes of some subset of the objects in an instantiation sequence,

- a pdf $p(\bullet|\theta)$, defined over the codomain of the mapping function, and
- a likelihood threshold Θ, which determines the sorting at the question node.

A mapping function maps an instantiation sequence to a value in a metric space. It does this by selecting some number of objects out of an instantiation sequence, and performing some numeric operation on their attributes. For instance, a mapping function might compute the relative difference of the color attributes of the first and second objects in the sequence. Another mapping function might simply return the value of the location attribute of the fourth object in the sequence.

For defining a decision boundary in Euclidean spaces, the Gaussian distribution is a convenient choice of pdf. Combined with a likelihood threshold, this defines an ellipsoidal volume in the space; points falling within this volume are considered as satisfying the question. For 2D orientation, we use a von Mises distribution [16]. The representative power of the SMRF approach is found in the different possible mapping functions that are available, and the ability to create an appropriate decision volume based on the training data. If, for example, the distance between two objects is a central aspect of the target concept, a "distance" mapping function allows this relationship to be expressed, and the pdf/threshold pair allows the appropriate distance between the objects to be modeled from the training data.

Formally, in the classification process, a question node computes $p(\phi(I)|\theta)$ for each instantiation sequence I sorted to that node. For each I, if $p(\phi(I)|\theta) \geq \Theta$, I is sorted down the Yes branch. If $p(\phi(I)|\theta) < \Theta$, I is sorted down the No branch. If $\phi(I)$ is undefined for I (e.g., if insufficient objects exist for instantiation or if an attribute used by ϕ is absent), then I is sorted down the Error branch.

C. SMRF Tree Learning Algorithm

The objective of the SMRF tree learning algorithm is to grow a tree that accurately predicts the label of novel object scenes. Because scenes are probabilistically classified, the algorithm seeks to build a tree that maximizes likelihood over the training set. The likelihood (L) of a tree given the training data is defined as follows:

$$L = \left(\prod_{S \in S^+} \Pr\left(\mathcal{W}(S)\right) \right) \times \left(\prod_{S \in S^-} \left(1 - \Pr\left(\mathcal{W}(S)\right)\right) \right), \quad (1)$$

where S^+ and S^- denote the set of positive and negative scenes, respectively, labeled based on whether or not they contain an instance of the target concept. $\mathcal{W}(S)$ denotes the highest-probability leaf in the tree into which some instantiation sequence from scene S is sorted. $\Pr(l)$ denotes the probability value that leaf l assigns to instantiation sequences that reach it.

The SMRF learning algorithm expands one leaf at each iteration. Each expansion includes zero or more instantiation nodes followed by one question node. Due to the size of the search space, mapping functions and initial pdf parameters are sampled from the available set. The best expansion is accepted if it passes a likelihood ratio test [17] with Šidák correction [18] for the repeated expansion attempts. Bodenhamer et al. [13], [14] present full details of the SMRF tree learning algorithm.

D. Parameterized Actions

We apply SMRF to the learning of outcomes for parameterized actions. When learning the outcomes of such actions, we include in each training scene, not only the world state, but also the arguments of the action. For the $Pour(x, y)$ example, training samples include the identification of x (the thing being poured from) and y (the thing being poured into).

To use this information when learning SMRF trees for a k-ary action, we constrain the objects selected for the first k instantiation nodes of any branch in the tree. When predicting the outcomes of the same action in future situations, we again bind the action arguments to the first k instantiation nodes. Any additional instantiation nodes produce permutations of other objects in the scene per standard SMRF evaluation. For example, if the SMRF tree in Fig. 1 were for some action $Act(x)$, x would only be bound to the action argument, but all remaining objects could be bound to y.

E. Forest-Based Classification

Because question node expansions are sampled, tree growth is a stochastic process. Therefore, while each SMRF tree yields probabilities from a fixed set of leaves, multiple SMRF trees together, all learned for the same classification task, provide a richer representation. Ensemble techniques (aggregating individual classifiers) have also been shown to improve accuracy when applied to new data [15].

The existential nature of SMRF concepts is another reason to use multiple trees. Sometimes absence, rather than existence, is what determines action success. For example, passing a ball in soccer is likely to be successful if *no* opponent is between the passer and receiver. Such *negative existential* concepts depend on $universal$ quantification. For instance, $\neg\exists_x T(x)$ is logically equivalent to $\forall_x \neg T(x)$. Because SMRF currently has no support for universal quantification, we instead learn trees for both the original and negated labels.

Therefore, to predict world outcomes, we learn a set of n trees, each using a subset of data sampled from an original training set. Half of the trees are learned using the original labels and half using negated labels. Each tree provides a predicted probability of outcome for a scene. We can thereby transform any domain state of arbitrary dimensionality (referring to objects and their attributes) into a new abstract state $s \in [0, 1]^n$. An aggregate classifier can then be learned on new scenes converted into n-dimensional vectors. Specifically, we use support vector machine (SVM) classification because it is effective and has readily available implementations, such as LIBSVM [19]. For the present work, we use only discrete classification from LIBSVM, although it and other techniques are also able to provide probability estimates, a capability that might be valuable for more advanced applications.

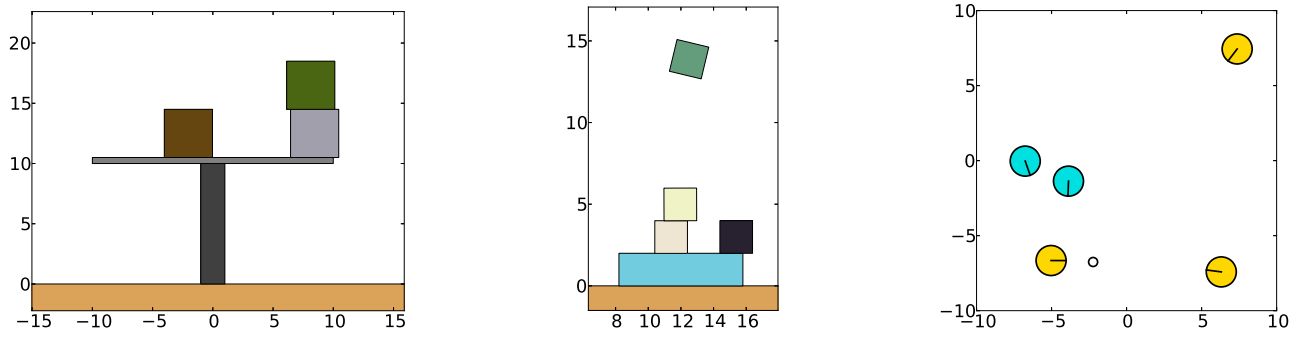

Fig. 2. Example starting states for the three tasks evaluated. The balance scale (a) tips either left or right, where we label left as true. The $DropOn(x, y)$ action (b) is successful if dropped block x comes to rest anywhere above y, here the elongated horizontal (cyan) block. The keepaway $Pass(x, y)$ action (c) is successful if the "keepers" (yellow) retain possession of the ball (white) after the pass from x (here, lower left) to y (lower right).

III. Experimental Results

Here, we present results for three problems: two in a 2D simulated physical blocks world and one in 2D simulated soccer. SMRF has been applied to concept learning in 3D domains as well [13], [14]. In all experiments, we provide the following mapping functions to SMRF: absolute and relative position, distance, reframed position (translated and rotated according to the location of two other objects), scaled reframed position (scaling the coordinate frame such that one unit is between the reference objects), absolute orientation, size, elongation, and individual and relative color. Some of these fit more naturally to some tasks than others.

All data sets consist of between 560 and 1000 total scenes, depending on the task, and we subsample 500 scenes without replacement for all SMRF and SVM training sets. For each task, we learn 50 SMRF trees with original labeling and 50 with negative labeling. This gives us a core set of learned representations. To observe variance resulting from the set of trees learned, we then subsample 25 positive and 25 negative trees, and perform 10-fold cross validation with LIBSVM using a second data set. We perform SMRF tree subsampling and SVM cross-validation 100 times. For the SVM, we use a linear kernel. Within each training set, we choose the SVM cost parameter C by means of internal 10-fold internal cross validation, evaluating options for C by powers of 10. Our performance measure, which also drives the choice of C, is the Peirce Skill Score (PSS) [20], which yields 1 for perfect classification, 0 for random guessing, and -1 for perfectly incorrect classification. This metric penalizes a model for merely guessing the most frequent class. We base each PSS value on aggregate results across the 10 test folds.

A. Blocks World

We first demonstrate our method on two tasks in a simulated blocks world built using the JBox2D physics engine [21]. This world is governed by a hidden gravity vector. For both tasks, no agent actions occur after each initial state. Only gravity and collisions affect the outcome.

The first of these two experiments is inspired by Siegler's classic developmental cognitive psychology work on human learning of rules for predicting the outcome of placing a set of weights on a balance scale [22]. The dynamics question is whether the spatially distributed weights will cause the scale to tip left, tip right, or remain balanced. This task focuses on an unstable world state rather than a parameterized agent action, although similar active tasks could be imagined. Also, while SMRF's capabilities are different than Siegler's proposed human rules, this task still provides an interesting case for comparison between human developmental learning of world dynamics and that of our method.

Here, we simplify the problem to whether the scale tips left or right, and do not consider balanced cases. An example initial state is shown in Fig. 2(a). We construct each initial scene with one to four randomly placed weights of equal size and mass along the horizontal beam. Weights that would otherwise overlap are stacked upward, approximately centered above the base of the stack. The constant-sized balance scale is always at the same location, and, while it clearly affects world dynamics, we leave it out of the scene description because it is constant. The absolute positions of weights directly indicate whether they are to the left or the right side.

To predict tipping left or right, learned SMRF trees commonly focus on the positions of the weights. For example, weights to the left suggest tipping left. Absolute position is indeed the most common mapping function in trees for this task. However, such weights cannot always be considered in isolation. With a small number of weights present, SMRF uses error branches, where failures to instantiate imply that there must be no counter weight on the opposite side. Relative mapping functions, especially distance, also appear in the trees. With more weights, high vertical positions imply large stacks, a heuristic perhaps usable by humans in approximate counting. Overall, the learned trees are effective, and our method performs nearly perfectly in this task.

Our second blocks world task is for an agent action, $DropOn(x, y)$, where block x is dropped on support block y. This action is considered successful if the dropped block indeed comes to rest over the support, even if other blocks are stacked between them. An example initial state is shown in Fig. 2(b). We choose a uniformly randomly distributed length

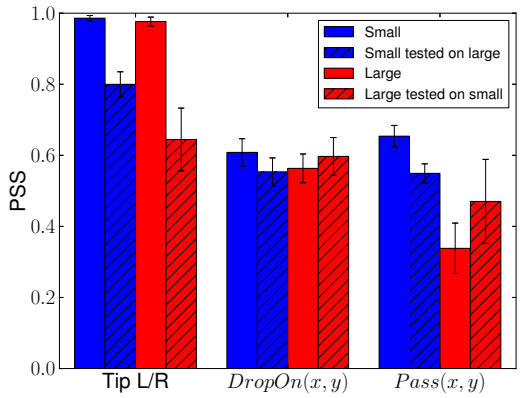

Fig. 3. Mean and standard deviation of classification performance, including cross testing for scenes with different numbers of objects than occurred during training. Small scenes contain fewer objects, and large scenes contain more.

and horizontal position for the support block, within certain bounds, as well as its orientation. We place the block to be dropped with normally distributed horizontal and uniformly distributed vertical position relative to the top of the support. The block to be dropped is a square of constant size. Zero to three potentially interfering blocks are dropped according to the same distribution in advance of the block drop action to be classified.

In this experiment, relative position is crucial. Learned trees often ask whether the dropped block is above the support block. Questions often follow about the elongation and orientation of the support. For branches identifying a narrow support block, questions constraining the dropped block's relative horizontal variance sometimes appear. Other structural forms of trees also occur, but often with similar kinds of questions. Rarely do trees instantiate additional interfering blocks. Such blocks seem to help as much as hinder, and seem more just to provide noise to the system.

Both block problems generalize to higher numbers of objects in test scenes than in training data, as shown in Fig. 3. In the balance scale case, we place five to six blocks for "large" scenes. Counting via error branches does not scale to larger scenes, but classification scores remain strong, implying that positional questions are still helpful. For the drop action, large scenes contain four to five interfering blocks, and even for trees learned in the larger scenes, such interfering blocks are rarely instantiated.

B. Soccer Domain

Our second experimental domain is that of the RoboCup 2D soccer simulator, which at this time we use only for the keepaway benchmark task [11]. While we do not attempt to scale to full soccer here, subtasks such as this have an eye toward learning some of the representations needed for the full game or other similar activities.

The keepaway game consists of two teams: keepers and takers. The keepers begin in possession of the ball, and an episode ends when the takers take possession or the ball goes out of bounds. The common configuration is 3 vs. 2, meaning three keepers and two takers, on a 20m by 20m playing area, as shown in the example state in Fig. 2(c). At each decision step, the keeper in control of the ball chooses either to hold the ball or to pass to a chosen teammate. All other players, including the takers, follow manually scripted behavior. We represent the pass action as $Pass(x, y)$ where x is the player with the ball and y is the intended recipient of the pass. For the purposes of producing training experience, we select the random policy for the keeper with the ball, meaning a uniformly random choice among all available pass and hold actions. The objects in the world are the players themselves. In this work, we do not include the ball in the scene description.

In keepaway, there is no inherent up or down, unlike the situation with gravity in the blocks world. On the other hand, opponents between passer and receiver are important, and the scaled reframed position mapping function, placing the passer at the origin and the receiver at (1, 0), commonly appears in learned trees. In this coordinate frame, potential interceptors appear near the x axis between coordinates 0 and 1. Exact boundaries depend on world dynamics, including agent behavior. Because interceptors cause failure rather than success, identifying them depends on negative labeling. Some learned trees consider multiple interceptors. Interestingly, with large SMRF training sets (such as the 500 scenes used here), we also see many learned trees for positive labeling, focusing initially on the position of the passer or the relative position of the receiver.

For keepaway, "large" scenes are based on 4 vs. 3 play on a 25m by 25m field. As for the blocks world, we again transfer somewhat to the larger game without retraining. While we do not use our system here for actual agent control, our transfer from 3 vs. 2 to 4 vs. 3 suggests similar capabilities to the scalability of the technique of Verbancsics and Stanley [10]. We show performance in Fig. 3.

Perhaps surprisingly, prediction is actually *better* on 4 vs. 3 play when transfering from 3 vs. 2 than when learning directly in the larger game. This seems to be due to a frequent inability of the algorithm to learn any tree expansions in 4 vs. 3 beyond the initial tree with one leaf. Such a tree provides only an unconditioned probability and is therefore uninformative about world state. Because so many trees learned in 4 vs. 3 are uninformative, a larger sampling of trees is necessary to ensure that at least some informative trees are included. Fig. 4 shows this effect, including the wide variance depending on the quantity of informative trees sampled. Note, however, that in 3 vs. 2 keepaway, PSS saturates quickly with relatively small forests. Other learning contexts evaluated in this paper have similarly quick saturation.

IV. Discussion and Conclusion

To function in an unstructured world, agents must understand the consequences of their actions. Here, we have demonstrated a robust method for action prediction on continuous tasks with arbitrary numbers of objects and implicit relations between them. This method learns trees using a technique similar to existentially quantified logic but which supports stochastic outcomes and also avoids the need for hand-crafted

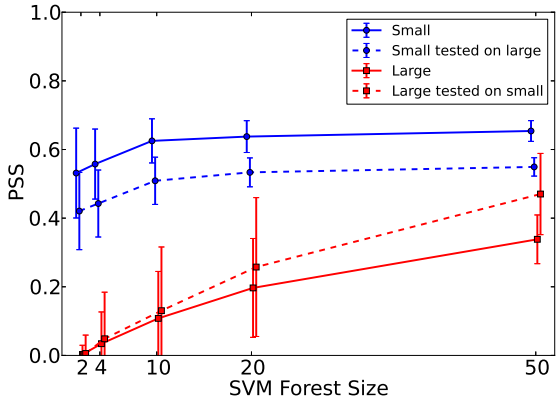

Fig. 4. Mean and standard deviation of performance as a function of forest size for the $Pass(x, y)$ action. Half of the trees in each forest are for positive labeling and half for negative (i.e., varying from 1×2 through 25×2).

predicates. We employ a forest of trees to transform an object-attribute state representation of arbitrary dimensionality into a robust, fixed-dimension space usable in standard vector-based machine learning algorithms. We have successfully applied our method, using a fixed set of mapping functions, to simulated domains with substantially different dynamics. We have also shown scalability, without retraining, to scenes with both more and fewer objects than are present in the training data.

Returning to Siegler's work on human prediction of balance scale outcomes [22], SMRF's tree expansion process is somewhat reminiscent of Siegler's progression to more advanced rules. However, Siegler suggests that humans use state features outside the kinds of mapping functions that we can currently apply in SMRF. For example, his rules include direct counting of blocks on each side of the scale. Both for group attributes, such as quantity, and for the many other varieties of potentially useful mapping functions, the number of compared tree expansions could increase substantially. We use a wide variety of mapping functions here, to good effect, but each carries a cost in the form of the multiple comparisons correction needed for tests of the significance of learned tree expansions. Learning time also increases with more mapping functions. Therefore, to increase flexibility, additional sampling and pruning techniques for mapping functions might be valuable.

In the future, we also intend to use this framework for actual agent control. One straightforward technique could be to sample potential choices for action parameters in the context of a larger planning system. Alternatively, SMRF trees could provide predicates for symbolic dynamics learning such as that developed by Pasula et al. [7] and also implemented by Lang and Toussaint [23]. The fixed-length feature vector could also perhaps be adapted for use as a basis in approximate reinforcement learning techniques [8]. Furthermore, extending SMRF to perform regression on real-valued variables could be fruitful toward the direct learning of more detailed dynamics.

Acknowledgments

During this work, Tom Palmer has been supported in part by a University of Oklahoma Foundation Fellowship. We would also like to thank Dougal Sutherland, Sam Bleckley, Dan Fennelly, and Amy McGovern for their past work on SMRF.

References

[1] F. Stulp, A. Fedrizzi, L. Mösenlechner, and M. Beetz, "Learning and reasoning with action-related places for robust mobile manipulation," *Journal of Artificial Intelligence Research (JAIR)*, vol. 43, pp. 1–42, 2012.

[2] J. J. Gibson, "The theory of affordances," in *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, R. Shaw and J. Bransford, Eds. Lawrence Erlbaum Associates, 1977.

[3] J. Mugan and B. Kuipers, "Autonomously learning an action hierarchy using a learned qualitative state representation," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009, pp. 1011–1016.

[4] J. Modayil and B. Kuipers, "The initial development of object knowledge by a learning robot," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 879–890, 2008.

[5] E. Brunskill, B. R. Leffler, L. Li, M. L. Littman, and N. Roy, "Provably efficient learning with typed parametric models," *Journal of Machine Learning Research (JMLR)*, vol. 10, pp. 1955–1988, 2009.

[6] W.-M. Shen, "Discovery as autonomous learning from the environment," *Machine Learning*, vol. 12, pp. 143–165, 1993.

[7] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research (JAIR)*, vol. 29, pp. 309–352, 2007.

[8] J.-H. Wu and R. Givan, "Automatic induction of Bellman-error features for probabilistic planning," *Journal of Artificial Intelligence Research (JAIR)*, vol. 38, pp. 687–755, 2010.

[9] M. van Otterlo, "A survey of reinforcement learning in relational domains," Centre for Telematics and Information Technology (CTIT) University of Twente, Tech. Rep., 2005.

[10] P. Verbancsics and K. O. Stanley, "Evolving static representations for task transfer," *Journal of Machine Learning Research (JMLR)*, vol. 11, pp. 1737–1769, 2010.

[11] P. Stone, G. Kuhlmann, M. E. Taylor, and Y. Liu, "Keepaway soccer: From machine learning testbed to benchmark," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Computer Science (LNCS), A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, Eds. Springer Berlin / Heidelberg, 2006, vol. 4020, pp. 93–105.

[12] J. Z. Xu and J. E. Laird, "Combining learned discrete and continuous action models," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2011, pp. 1449–1454.

[13] M. Bodenhamer, T. Palmer, D. Sutherland, and A. H. Fagg, "Grounding conceptual knowledge with spatio-temporal multi-dimensional relational framework trees," School of Computer Science, University of Oklahoma, Tech. Rep. TR-AIR-1138, 2012.

[14] M. Bodenhamer, S. Bleckley, D. Fennelly, A. H. Fagg, and A. McGovern, "Spatio-temporal multi-dimensional relational framework trees," in *IEEE International Conference on Data Mining (ICDM) Workshop on Spatial and Spatiotemporal Data Mining (SSTDM)*, 2009, pp. 564–570.

[15] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] K. V. Mardia, "Statistics of directional data," *Journal of the Royal Statistical Society, Series B*, vol. 37, no. 3, pp. 249–393, 1975.

[17] J. P. Huelsenbeck and K. A. Crandall, "Phylogeny estimation and hypothesis testing using maximum likelihood," *Annual Review of Ecology and Systematics*, vol. 28, pp. 437–466, 1997.

[18] D. Jensen and P. Cohen, "Multiple comparisons in induction algorithms," *Machine Learning*, vol. 38, no. 3, pp. 309–338, 2000.

[19] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[20] D. B. Stephenson, "Use of the "odds ratio" for diagnosing forecast skill," *Weather and Forecasting*, vol. 15, no. 2, pp. 221–232, 2000.

[21] D. Murphy, "JBox2D: A Java physics engine," accessed August 2, 2010. [Online]. Available: http://jbox2d.org/

[22] R. S. Siegler, "Three aspects of cognitive development," *Cognitive Psychology*, vol. 8, no. 4, pp. 481–520, 1976.

[23] T. Lang and M. Toussaint, "Planning with noisy probabilistic relational rules," *Journal of Artificial Intelligence Research (JAIR)*, vol. 39, pp. 1–49, 2010.