Grounding Conceptual Knowledge with Spatio-Temporal Multi-Dimensional Relational Framework Trees

(ARTIFICIAL INTELLIGENCE AND ROBOTICS TECHNICAL REPORT #1138)

Matthew Bodenhamer
School of Computer Science
University of Oklhoma
mbodenhamer@ou.edu

Thomas Palmer
School of Computer Science
University of Oklahoma
tompalmer@ou.edu

Dougal Sutherland
School of Computer Science
Carnegie Mellon University
dsutherl@cs.cmu.edu

Andrew H. Fagg
School of Computer Science
University of Oklahoma
fagg@cs.ou.edu

Abstract

The real world is composed of sets of objects that have multidimensional properties and relations. Whether an agent is planning the next course of action in a task or making predictions about the future state of some object, useful task-oriented concepts are often encoded in terms of the complex interactions between the multi-dimensional attributes of subsets of these objects and of the relationships that exist between them. In this paper, we present extensions to the Spatiotemporal Multi-dimensional Relational Framework (SMRF) Trees, a data mining technique that extends the successful Spatiotemporal Relational Probability Tree models. From a set of labeled, multi-object examples of some target concept, our algorithm infers both the set of objects that participate in the concept, as well as the key object and relational attributes that characterize the concept. In contrast to other relational model approaches, SMRF trees do not require that categorical relations between objects to be defined a priori. Instead, our algorithm infers these categories from the continuous attributes of the objects and relations in the training data. In addition, our approach explicitly acknowledges the multi-dimensional nature of attributes, such as position, orientation and color in the creation of these categories. We present an updated learning algorithm for the SMRF approach, and validate our updated algorithm in both two and three dimensional domains that contain groups of static or moving objects.

1 Introduction and Related Work

The world can be modeled as a collections of objects, each with a set of associated attributes. Whether it is a robot preparing to perform the next step in a cooking sequence or an agent generating warnings of severe weather, only a specific subset of the observable objects is relevant to making decisions about what steps to take next. In particular, the relevance of an object is determined by its attributes and the relations that it has with other objects. These attributes are often continuous and multi-dimensional, such as Cartesian positions or colors in a red-green-blue (RGB) space. Given a set of training examples, our challenge is to discover the objects that play the crucial roles in the examples as well as the description of the key object attributes and relations.

The problem of learning these attributes and relations is a form of *relational learning* [8, 12]. Most relational learning approaches are formulated in terms of graphical models [6, 23, 19], logical formulae [2], or some combination thereof [22, 24]. Our work, however, focuses on a different method of solving the problem, utilizing augmented decision trees. In particular, the SMRF approach is inspired by the successful Relational Probability Tree (RPT) [20] and the Spatiotemporal Relational Probability Tree (SRPT) [18] approaches, which create probability estimation trees [21], a form of a decision tree with probabilities at the leaves. Splits in the decision trees can ask questions about the observed properties of the objects or their relationships. Given a novel graph, these decision trees estimate the probability that the graph contains a set of objects that corresponds to some target concept. Like Kubica et al. [14, 13], these approaches build models using pre-specified categorical relations.

In this paper, we discuss the Spatiotemporal Multidimensional Relational Framework (SMRF) [3], as well as extensions to the original formulation presented in [3], which include a new and improved learning algorithm. SMRF trees extend prior relational probability tree approaches in a number of significant ways. The first extension is the ability to ask questions based on continuous, multi-dimensional attributes. For example, the color of a pixel can be represented as a RGB tuple. Capturing a concept such as "yellow" requires that the blue variable be low but the green and red variables can take on values almost within their full range so long as they vary together. While RPTs and SRPTs can split on individual continuous variables (e.g., [11, 6, 7]), it is desirable to explicitly acknowledge the fact that multiple dimensions can covary in interesting ways.

A second key extension made by the SMRF approach is the ability to define relational categories dynamically. For example, objects may have a position attribute defined within some global coordinate frame. The decision tree splits can be made within a metric space that captures the position of one object *relative* to another. As with the multi-dimensional object attributes, splits on these relational attributes are made using decision surfaces within the metric space. In contrast, RPTs and SRPTs ask relational questions using categorical descriptions of the attributes.

A third key extension made by the SMRF approach is the ability to reason explicitly about object instances. In graph-based approaches (such as RPTs and SRPTs), the objects/relations that satisfy a particular question are typically not represented in such a way that they can be referenced by questions deeper in the tree. This limits the types of concepts that can be easily represented and can make it difficult to address the question of *who* the actors are that play the key roles in determining the graph label. We refer to this as a graph-based method because the query (a graph representing objects and relations) descends as a single unit through the decision tree. In contrast, an instance-based method (such as SMRF trees) explicitly represents the acting objects through an instantiation process. A question at one level in the tree can then refer to the set of objects that have been instantiated to that point. This allows absolute questions about individual objects or relative questions about two or more objects to be asked.

In this paper, we provide an overview of SMRF trees, and then discuss an improved learning algorithm, compared to [3]. We then describe experiments performed using the new algorithm and discuss the implications of their results.

2 SMRF Trees

SMRF trees assess the probability that a given collection of objects contains an instance of a particular target concept. We model these collections of objects as graphs with attributed objects and relations. Many relations in our context are defined as a function of the attributes of the objects. 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. A *target concept* encompasses a subset of objects from an example and encodes specific attribute values of those objects and relations between them. Each graph is labeled as either positive or negative on the basis of whether or not some subset of its objects matches the target concept definition.

The learning problem to be solved is: given some number of positive and negative graphs, construct a model that can assess the probability that a given graph contains the target concept [17]. For instance, a target concept may be "a red object near a green object." When, for example, blue objects are also present in some of the positive graphs, the learning approach must weigh the set of

possible target concepts, and choose the one that best explains the positive graphs. This problem of having to identify the subset of objects and their relationships that define a target concept can be viewed as one of *multiple instance learning* (MIL) [4, 16, 17], since the label of each graph is known, but it is unknown which combinations of objects in the graph encode the target concept.

As an illustration of the classification process, Figure 1 shows two hand-constructed SMRF trees. Panel *a* shows a very simple SMRF tree that identifies yellow objects. The first node in the tree in panel a dynamically binds objects from the graph to the variable A. This instantiation action enables SMRF trees to classify graphs based on attributes of particular objects or relations. Although the question in the figure is rendered in English, it is actually asked with respect to a decision volume represented in RGB space. For RGB variables (as well as positions in Euclidean space), it is convenient to use an ellipsoidal decision volume. If the query falls within this volume, the object falls down the *Yes* branch of the question node. If the query falls outside the volume, the object falls down the *No* branch. If the object has no color attribute, it falls down the *Error* branch.

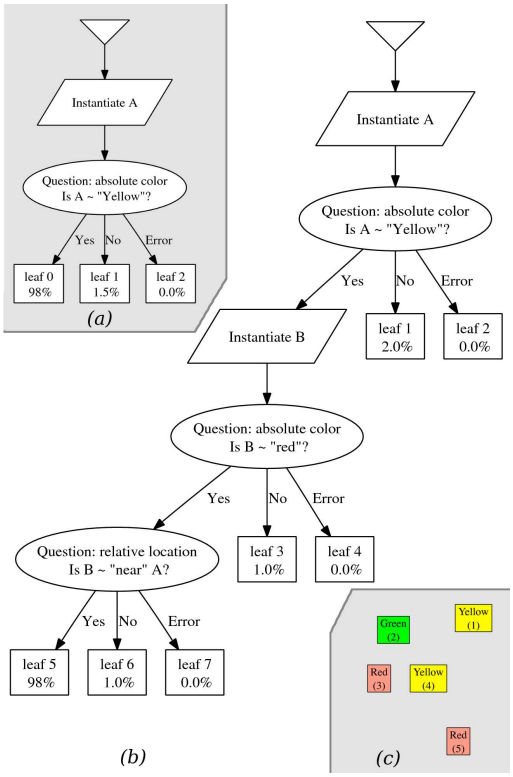


Figure 1: (a) A hand-crafted SMRF tree that identifies yellow objects. Although we describe the decision tree split with a categorical label ("yellow"), the concept is represented as a volume in RGB space. (b) A tree that identifies yellow objects that are near red objects. (c) A set of objects scattered on a plane.

Leaf nodes represent a probability function over membership in the target concept. In this example, yellow objects are the most likely examples of the target concept. When this tree is used to classify the set of objects shown in Figure 1(c), A is instantiated with each object in turn. Here, objects 1 and 4 fall into the Yes branch leaf node, and are assigned a probability of 98% of being in the target concept, as calculated from the training data. Figure 1(b) shows a more complicated example that identifies a yellow object near a red object. At the root, A is instantiated with each object in the graph. Following the yes branch, B is instantiated with a second object drawn from the graph. The tree then asks whether B is red and, if so, examines the relative position of A and B. In this example, the position of A is measured within a coordinate frame whose origin is defined by the position of object B. We refer to a specific ordered list of instantiations (e.g., A = 1 and B = 2) as an *instantiation sequence*. The variables are implicit in the order of the sequence. For example, the set of instantiation sequences that is expanded and subsequently sorted by the tree in Figure 1(b)

for the objects in Figure 1(c) is: leaf 1: (2), (3), (5), leaf 3: (1, 2), (1, 4), (4, 2), (4, 1), leaf 5: (4, 3), and leaf 6: (1, 3), (1, 5), (4, 5). An instantiation sequence that is expanded into a set of longer instantiation sequences is called a *parent*, and the set of expanded sequences is called the *children*. Because instantiation sequence (4, 3) falls into a leaf of high probability, the graph is labeled as likely to contain an instance of the target concept. Furthermore, the tree identifies objects 3 and 4 as the key actors. Note that not all combinations of objects or relations are considered during the query process because the tree structure enables the search space to be quickly pruned. In Figure 1(b), we only instantiate B objects in the cases where A satisfies the "yellow" model.

The SMRF approach allows complex questions 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 decision threshold Θ, which determines the sorting at the question node.

A *mapping function* maps an instantiation sequence to a value in a metric space. A mapping function selects some number of objects out of an instantiation sequence, and performs some numeric operation on their attributes. For instance, a mapping function might compute the relative distance of the spatial locations of the first and third objects in the instantiation sequence. Another mapping function might compute the relative difference of the color attributes of the first and second objects in the sequence. A third 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. 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 part of the target concept, the mapping function allows the distance 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 $p(\phi(I)|\theta) \leq \Theta$, I is sorted down the No branch. If $p(\phi(I)|\theta) > \Theta$, I is sorted down the Yes branch. If $\phi(I)$ is undefined for I (e.g., if ϕ is defined over an attribute that the objects in I do not possess), then I is sorted down the Error branch.

3 Learning Algorithm

The objective of our SMRF tree learning algorithm is to grow a tree that accurately predicts the label of new graphs. A SMRF tree probabilistically classifies a graph as containing the target concept based upon the probability of the highest-probability leaf node into which an instantiation from the graph in question is sorted. Because graphs are probabilistically classified, the algorithm seeks to build a tree that will maximize the likelihood of correct graph classification over a training set. The likelihood of correct classification (L) is defined as follows:

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

where G^+ and G^- denote the set of positive and negative graphs, respectively, and $\mathcal{W}(G)$ denotes the highest-probability leaf in the tree into which an instantiation sequence from graph G is sorted. In addition, $\Pr(l)$ denotes the probability value that leaf l assigns to instantation sequences that reach it. The likelihood of the data given the tree is thus higher when instantiation sequences from positive graphs are sorted into leaf nodes with a high probability, and lower when no instantiation sequences from a positive graph are sorted into a high-probability leaf. The likelihood of the data is also higher when no instantiation sequences from negative graphs are sorted into high-probability leaves, and lower when at least one such instantiation sequence is sorted into a high-probability leaf.

Learning a tree that maximizes L for a given dataset is an iterative multi-step process. On the initial iteration, the learning algorithm begins with a "stub" – a trivial tree comprised of only a root node and a single leaf node. The tree is then repeatedly grown according to the following greedy algorithm:

1. A set of leaf nodes is chosen for possible expansion.
2. For each leaf node to be expanded:
   (a) A set of possible expansions is sampled, resulting in a set of candidate trees.
   (b) For each candidate tree:
      i. The parameters of each question node model are optimized.
      ii. A decision threshold in likelihood space is chosen.
      iii. The tree leaf node probabilities are re-computed.
3. The candidate tree with the greatest improvement to L is identified. If the improvement is statistically significant, it replaces the current tree and the algorithm begins again at step 1. Otherwise, the algorithm halts.

The details of the above operations are explained in more detail below.

Choosing Leaves for Expansion In order to make the algorithm more efficient, only a subset of the available leaves are considered for expansion. To select the leaves to expand, the algorithm scores each of the leaves, and selects the n highest-scoring leaves above a minimum threshold. In this work, n was empirically set to three. Each leaf is scored according to how much L would improve if the leaf were to be replaced by an optimal expansion.

Generating Candidate Trees The expansion process replaces a leaf node in the tree with a partial tree, called an *expansion*. All legal expansions contain one question node, which has three leaf nodes as children (on its Yes, No, and Error branches, respectively). Some legal expansions also include one or more instantiation nodes above the question node. The expansion process thus enables the tree to ask a new question about previously instantiated objects, about new objects, or incorporating both. Each candidate question node is determined by its mapping function. Each mapping function is associated with a particular pdf, such as a Gaussian or von Mises distribution. The pdf parameters and decision threshold are learned later in the algorithm.

A number of mapping functions are defined, allowing the learning algorithm to ask a variety of types of questions. "Absolute" mapping functions simply return the value of an attribute at a particular position in the instantiation sequence. For example, a mapping function might return the location of the third object in the instantiation sequence. "Relative" mapping functions return the (vector) difference between the values of a particular attribute for two different objects. For example, a mapping function might compute the vector difference in RGB space between the second and third object of the instantiation sequence. Mapping functions that compute the Euclidean distance between two attribute values are also defined.

For each leaf node selected for expansion, a number of candidate expansions are individually applied to the tree, resulting in a set of candidate trees. Each candidate tree is then optimized according to the following procedures.

Optimizing Question Node Models For a given candidate tree, the parameters of the pdf at the expansion question node are optimized so as to capture the output of the mapping function for the "best" instantiation sequences. Each instantiation sequence is given a weighting, and the maximally-weighted instantiation sequences from each positive graph are used to estimate the model parameters.

Models are learned using an iterative process. An initial model with parameters θ_1 is selected based on a point of maximal diverse density [15]. Based on the current model, each instantiation sequence I at iteration t is assigned a weight $w_t(I)$, which is equivalent to $p(\phi(I)|\theta_t)$. New model parameters θ_{t+1} are estimated to maximize the following likelihood:

$$\hat{L}(\theta_{t+1}) = \prod_{G \in G^+} p(\phi(I_x^G)|\theta_{t+1}))^{w_t(I_x^G)}, \tag{2}$$

where I_x^G denotes the highest-weighted instantiation sequence from G at the question node being optimized. This process is repeated until $\hat{L}$ converges.

Computing Decision Thresholds Once the final parameters θ_q of the pdf model have been determined, a decision threshold is computed to determine how the model will sort instantiation sequences. Following Friedman (1977) [5], we use the point of maximum Kolmogorov-Smirnov (K-S) distance between $p(\phi(I_x^G)|\theta_q)$'s of the positive graphs and the $p(\phi(I_x^G)|\theta_q)$'s of the negative graphs as the decision boundary. After determining the parameters of the question node model, the instantiation sequences are sorted into the appropriate child leaf nodes.

Computing Leaf Node Probabilities The last step of the optimization procedure is to compute tree leaf node probabilities that maximize L for the current sorting. This task is made difficult given that the value of L is determined by a series of related $\max$ operators. However, we can see that if a leaf l is the maximum-probability leaf for p positive graphs and n negative graphs, the probability assignment that maximizes l's contribution to the overall likelihood L is $\frac{p}{n+p}$ (the intuitive frequentist value). We can therefore maximize the overall likelihood through the following process: The ratio $\frac{p}{n+p}$ is computed for each leaf l. The maximum such ratio is assigned as the probability for its corresponding leaf. The process is then repeated, with new p and n counts ignoring the graphs sorted to leaves that have already been assigned probabilities. Once all graphs have been removed from consideration, any remaining leaves are assigned probabilities of 0.

Evaluating Statistical Significance Over the set of candidate expansions, the best performing candidate tree T_c is considered as a replacement for the current tree. If this new tree performs significantly better than the current tree according to a likelihood ratio test, then this replacement is kept. Because the sample size is reasonably large, $-2 \log \frac{L}{L_{T_c}}$ is approximately χ^2 distributed with degrees of freedom equal to the difference in the number of parameters in the candidate tree and the current tree [9]. The number of parameters in the tree is the number of leaf and instantiation nodes, plus the sum of the number of distribution parameters in each question node. To compensate for the multiple comparisons problem, we employ a Šidák correction to obtain a collective cutoff of $\alpha = 0.1$ [10].

4 Experimental Results

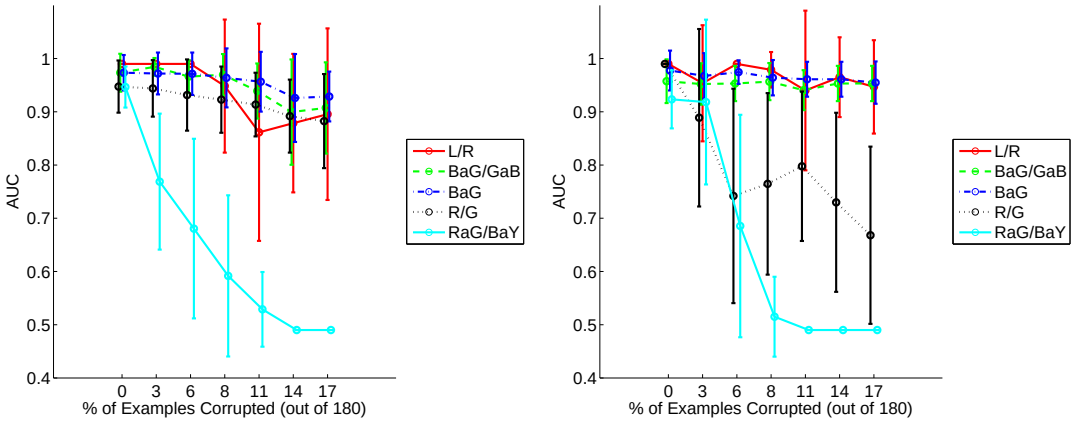
Through the following experiments, we demonstrate five key properties of the proposed learning approach. First, that the learning algorithm is able to construct an appropriate model for grounding a particular target concept. Second, that the algorithm can construct complex target concepts consisting of multiple questions. Third, that the algorithm can find an appropriate target concept even when "distractor" objects are present. Fourth, that the algorithm can be robust to noise in the labels of the training examples. And fifth, that the algorithm can learn disjunctive concepts. The algorithm was tested with six problems: five involving the classification of multi-object structures and one involving the prediction of the outcome of actions in a dynamic environment.

4.1 Multi-Object Structures

The following datasets were utilized to test the ability of the algorithm to classify conjunctive and disjunctive spatial concepts:

1. a blue object above a green object (BaG),
2. a blue object above a green object, or a green object above a blue object (BaG/GaB),
3. a red object above a green object, or a blue object above a yellow object (RaG/BaY),
4. a red object or a green object (R/G),
5. an object oriented to the left, or an object oriented to the right (L/R).

With the exception of the last dataset, all were comprised of objects with two attributes: *3D position* and *RGB color*. The last dataset was comprised of objects with three attributes: *2D position*, *RGB color*, and *2D orientation*. All of the datasets were computer-generated.



(a) 0 Distractors (b) 2 Distractors

Figure 2: Mean and std AUCs for the five datasets.

A number of mapping functions were provided to the algorithm. For the 3D (2D) position attribute, the algorithm could choose to model the absolute location of an object in $\mathbb{R}^3$ ($\mathbb{R}^2$), the distance between the object locations, or the difference vector between two objects in $\mathbb{R}^3$ ($\mathbb{R}^2$). The pdfs used for these models were Gaussian distributions of three (two) dimensions. For the color attribute, the algorithm could choose to model either the RGB color of a single object, or the difference vector in RGB space between the colors of two objects. These pdfs were also Gaussians of three dimensions. For 2D orientation, the algorithm could choose to model the absolute orientation of an object in angular coordinates. The pdf for this model was a von Mises distribution.

For these data sets, 100 positive example graphs and 100 negative example graphs were generated, according to the specific target concept involved. For training and testing the algorithm, 10-fold cross-validation was employed. SMRF trees do not definitely classify a graph as containing or not containing a target concept, but rather return the probability that the graph contains the target concept. For this reason, classification performance was measured in terms of the receiver operating characteristic (ROC) curve, and the area under this curve (AUC).

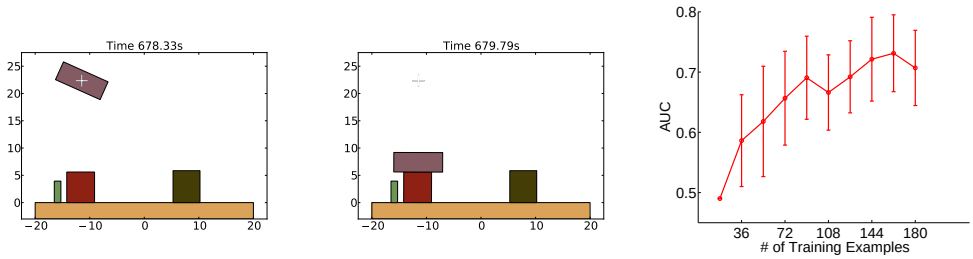
The results of these experiments are shown in Figure 2. While these concepts are complex in their own right, distractor objects were also added to test the ability of the learning algorithm to infer the target concept under more difficult conditions. In addition, to test the robustness of the learning algorithm under other noise conditions, the labels of some percentage of randomly-chosen examples in the training set were inverted (from positive to negative, or vice-versa).

These results indicate that the algorithm is able to learn a range of complex and disjunctive spatiotemporal concepts, involving such varied attributes as location, color, and orientation. In particular, such concepts can be learned to a high degree of accuracy in the no corruption case, even with a moderate number of distractors present, as seen in Figure 2(b). As the amount of corruption increases, classification performance for the harder concepts decreases. However, for most of the simpler concepts, performance does not decrease significantly, even for corruption levels as high as 17%. Together, these results demonstrate an ability to learn varied complex, disjunctive, spatiotemporal concepts, and a robustness to various noise conditions.

4.2 Predicting the Outcome of Dynamic Actions

With an eye toward learning concepts that predict world dynamics, we also employ a 2D simulated physical "blocks world" environment.[1] We construct a scenario where blocks are dropped randomly over the table, and occasionally all blocks are cleared to reset the world state. We then label each state where a block is dropped based on whether the block eventually settles in a state in which it is supported in some fashion by another block. Figure 3a shows an example world state before the drop occurs, which is also the state about which the outcome is predicted. Panel b shows the actual outcome state, which determines the label of this example. Here, the dropped block lands on

[1]Our simulation, available online at http://code.google.com/p/stackiter/, uses the JBox2D physics engine.



(a) Before Drop (b) After Drop (c) Performance

Figure 3: A positive example (a) from the dynamics data set, labeled according to the resting state (b), with mean and std AUC for different numbers of training examples (c).

a second block, resulting in a *positive* label. This continuous domain has inherent "noise" in that simple questions of relative position are insufficient to precisely predict the resting state. As another complication, the number of distractor objects varies depending on the number of dropped blocks since the most recent clearing.

For learning, presumably useful mapping functions include location, distance, relative location, orientation, block size, and the ratio of block width to height. Functions not expected to be useful include color and relative color. Figure 3c shows how the number of training examples influences the quality of learned trees. Given sufficient data, trees predicting this question of world dynamics can be learned, despite its inherent complexity. Learned trees often include either one or two questions, using only location, distance, and/or relative location in the vast majority of cases.

5 Discussion and Conclusion

The SMRF tree learning algorithm must simultaneously identify the objects contained within the training set graphs that participate in the target concept, along with the set of object and relational attributes that explain the target concept. The difficulty of this task is increased by the use of disjunctive datasets, as the algorithm must identify multiple sub-concepts within each positive graph. The learning task is made even more difficult by the addition of noise, both in the form of distractor objects, as well as corruption in the graph labels.

The results on the multi-object structures experiments demonstrate an improvement over the learning algorithm used in [3], which was only able to learn relatively limited conjunctive concepts. The learning algorithm presented in this paper is an improvement over the previous version, which maintained a hidden variable representation for each instantiation sequence. The use of such a representation is computationally costly, and leads to sub-optimal trees in many cases. With no corruption, the algorithm is able to learn all of the concepts, even with a moderate number of distractor objects present, with mean AUCs greater than 0.9. As corruption increases, the algorithm demonstrates a robustness to noise, as classification performance on the easier datasets does not decrease significantly, even out to corruption levels of 17%. On the harder datasets, such as RaG/BaY, which are more sensitive to deviant examples, an increase in the level of corruption drives down classification performance rather quickly. Such a trend is not unexpected, and is representative of the degree of difficulty inherent in the problem itself. The BaG/GaB dataset, on the other hand, does not suffer significantly from corruption, and this is due to the fact that it is an easier concept than RaG/BaY because of its inherent symmetry.

In addition, the results on the dynamic action outcome prediction experiments demonstrate an ability to work towards a solution of a problem that is rooted in real-world physics. This particular domain features a large search space, and it is encouraging that the algorithm is able to classify significantly better than random.

These results indicate that SMRF has a good deal of promise as a method for learning complex spatiotemporal concepts in real-world applications. Further work is under way to enable the application of SMRF to the domain of Learning from Demonstrations [1], where it would be used to enable a robotic agent to learn to perform tasks through interactions with human teachers. Such interactions would be enriched by the use of social cues, such as gestures and linguistic utterances, which would

be used by the SMRF algorithm to prune the hypothesis space and learn the target concept much more efficiently and accurately.

Acknowledgments

We thank Samuel Bleckley and Daniel Fennelly for their contributions on prior versions of this work. This material is based upon work supported by the National Science Foundation under Grant Nos. IIS/REU/0755462 and IIS/CAREER/0746816.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, 2009.

[2] Hendrik Blockeel and Luc De Raedt. Top-down induction of logical decision trees. *Artificial Intelligence*, pages 285–297, 1998.

[3] M. Bodenhamer, S. Bleckley, D. Fennelly, A. H. Fagg, and A. McGovern. Spatio-temporal multi-dimensional relational framework trees. In *Proceedings of the International Workshop on Spatial and Spatiotemporal Data Mining, IEEE Conference on Data Mining*, 2009. electronically published.

[4] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[5] Jerome H Friedman. A recursive partitioning decision rule for nonparametric classification. *IEEE Transactions on Computers*, C-26(4):404–408, 1977.

[6] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1300–1309, 1999.

[7] Lise Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

[8] Lise Getoor and Ben Taskar, editors. *Introduction to statistical relational learning*. MIT Press, Cambridge, Massachusetts, 2007.

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

[10] D. Jensen and P. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.

[11] David Jensen and Lise Getoor. IJCAI 2003 workshop on learning statistical models from relational data. http://kdl.cs.umass.edu/srl2003/, 2003.

[12] Charles Kemp and Alan Jern. Abstraction and relational learning. *Advances in Neural Information Processing Systems*, 22, 2009.

[13] Jeremy Kubica, Andrew Moore, David Cohn, and Jeff Schneider. Finding underlying connections: A fast graph-based method for link analysis and collaboration queries. In *Proceedings of the International Conference on Machine Learning*, pages 392–399, 2003.

[14] Jeremy Kubica, Andrew Moore, and Jeff Schneider. Tractable group detection on large link data sets. In Xindong Wu, Alex Tuzhilin, and Jude Shavlik, editors, *The Third IEEE International Conference on Data Mining*, pages 573–576. IEEE Computer Society, 2003.

[15] Oded Maron. *Learning from Ambiguity*. Phd thesis, Massachusetts Institute of Technology, 1998.

[16] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 570–576, Cambridge, Massachusetts, 1998. MIT Press.

[17] A. McGovern and D. Jensen. Identifying predictive structures in relational data using multiple instance learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 528–535, 2003.

[18] Amy McGovern, Nathan Hiers, Matthew Collier, David John Gagne II, and Rodger A. Brown. Spatiotemporal relational probability trees. In *Proceedings of the 2008 IEEE International Conference on Data Mining*, pages 935–940, Pisa, Italy, December 2008.

[19] Jennifer Neville and David Jensen. Relational dependency networks. *J. Mach. Learn. Res.*, 8:653–692, 2007.

[20] Jennifer Neville, David Jensen, Lisa Friedland, and Michael Hay. Learning relational probability trees. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 625–630, 2003.

[21] F. Provost and P. Domingos. Well-trained pets: Improving probability estimation trees. Technical report, Stern School of Business, NYU, 2000.

[22] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1–2):107–136, 2006.

[23] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2005.

[24] Jue Wang and Pedro Domingos. Hybrid markov logic networks. In *Proceedings of the Twenty-third AAAI Conference on Artificial Intelligence*, 2008.