ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-973

Learning action effects in partially observable domains

Kira Mourão and **Ronald P. A. Petrick** and **Mark Steedman** [1]

Abstract. We investigate the problem of learning action effects in partially observable STRIPS planning domains. Our approach is based on a voted kernel perceptron learning model, where action and state information is encoded in a compact vector representation as input to the learning mechanism, and resulting state changes are produced as output. Our approach relies on deictic features that assume an attentional mechanism that reduces the size of the representation. We evaluate our approach on a number of partially observable planning domains, and show that it can quickly learn the dynamics of such domains, with low average error rates. We show that our approach handles noisy domains, conditional effects, and that it scales independently of the number of objects in a domain.

1 INTRODUCTION AND MOTIVATION

Acquiring a domain model automatically through learning and experience gives an agent greater flexibility to handle unexpected situations, and avoids the need for a predefined world model. Existing approaches either work within the space of transition rules to find a "good" set, or all consistent sets, of rules [2, 3, 12, 15], or they operate at the sensor level by constructing transition rules from actions and robot sensor data coded as sets of objects or raw sensor readings, and predicates derived from this data [7, 10]. The former, high-level, methods have been applied to partially observable [2, 15] or non-deterministic [12] domains, but are not applicable to domains which are both noisy and partially observable; few are also able to learn conditional effects. The latter, low-level, methods can learn in noisy, partially observable domains, but the domains are much simpler, without relations between objects, and sometimes without objects at all. Here, we extend our previous work on learning action models in noiseless, fully observable domains [11]. Our method learns the effects of STRIPS actions [4], extended to admit conditional effects, in deterministic, noisy and partially observable versions of the more complex domains typical of the high-level approaches.

2 REPRESENTATION

We learn action models from sequences of interleaved actions and state observations. Each observation initially encompasses as much of the world state as the agent is able to detect, with some parts of the state potentially unobserved or corrupted by noise. We reduce the size of each observation by only considering objects which can be identified by a deictic reference [1], and then transform each observation into a vector to use as input to the learning model.

A deictic representation maintains pointers to objects of interest in the world, with objects coded relative to the agent or current action.

[1] University of Edinburgh, UK, email: `kira.mourao@ed.ac.uk`, `{rpetrick,steedman}@inf.ed.ac.uk`

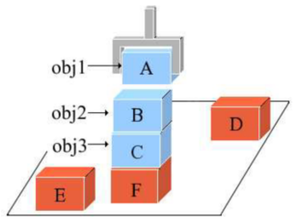

Figure 1. Computing deictic references: an example from the BlocksWorld domain, in which an agent can manipulate a set of blocks on a table. Given the action *stack*(A, B), i.e., stack block A on top of block B, the initial set of objects of interest is $\{A, B\}$. The only object related to A or B is C, since B is on C. Therefore the full set of objects of interest is $\{A, B, C\}$.

We take a similar approach to previous work applying deictic representations to learn domain dynamics [3, 12]. For a given action instance we construct the set of objects of interest, consisting of the set of objects which are parameters of the action, and the objects which, in the current state, are related to any object in the action parameters (see Figure 1). This single step computation is in contrast to previous approaches, where the set of objects under consideration is the full transitive closure under all relations among objects. Also, whereas previous approaches ignored objects if they were not *uniquely* defined by deictic reference, we allow deictic references to any set of objects that are indistinguishable relative to the action parameters.

An input vector representing the reduced state space is then constructed by assigning a bit for each action, 0-ary fluent, and for each possible relation involving only the objects in the reduced state space. The value of a bit is 1 (-1) if the corresponding fluent is true (false), or if the corresponding action is (not) the current action. Bits for unobserved or unused fluents are set to an arbitrary value *N*, which is ignored during learning.

Vectors representing an action's effects on a state are identical in form to the input vectors, except that actions are excluded from the vector, and bits are set to 1 (-1) if the corresponding fluent changes (does not change). Bits corresponding to unobserved or unused fluents are set to *N*.

3 LEARNING MODEL

The task of the learning mechanism is to learn the deterministic associations between action-state pairs and their effects. It is assumed that the number and type of parameters of each action, predicate and function are known. Action preconditions and effects are not known, and effects may be conditional. Disjunctive effects are not allowed. Instead, all effects are conjunctions of predicates, meaning it is sufficient to learn the rule for each predicate separately. Using the vector representation defined above, state transitions can be learnt using a bank of classifiers, one for each bit of the output vector.

To address our learning problem we construct a variant of the perceptron algorithm [13], using the voted perceptron [5], which is

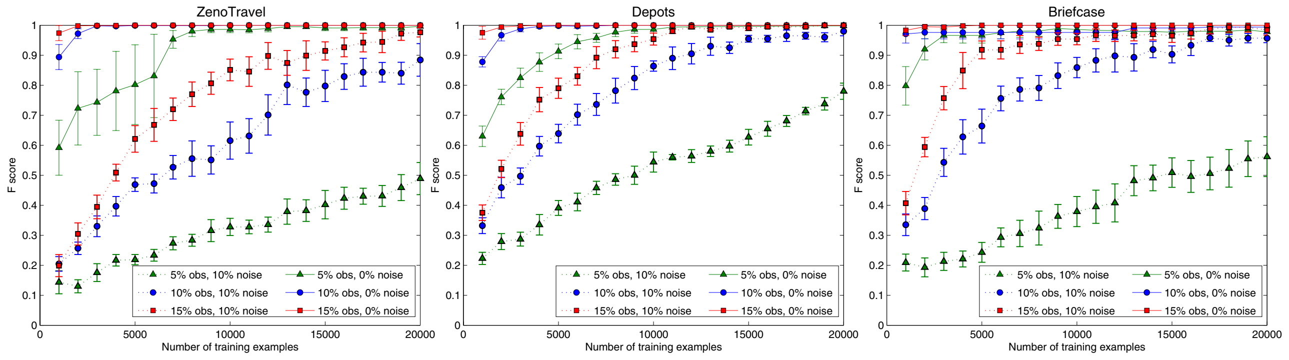

Figure 2. Results of learning action models in standard planning domains. Error bars are 95% confidence intervals. In noiseless, fully observable domains, models fully predict all test cases after less than 200 examples (results not shown). While observing only a small fraction of the state, without noise, the learning model completely predicts the test set after 20000 examples, in many of the test cases: with 15% of the state observable, the F-score is not significantly different from 1 (t-tests, $p > 0.05$) in any of our domains. With noise, the learnt models are clearly poorer, but some aspects of each domain are still learnt: with 15% of the state observable, the F-score is significantly different from 1 for ZenoTravel ($p = 0.004$) and Briefcase ($p = 0.046$), but not for Depots ($p > 0.05$).

noise-tolerant [8] and computationally efficient, producing performance close to the best performing maximal-margin classifiers (e.g. SVMs) on similar problems. We use the DNF kernel [14], which allows the perceptron to run over the feature space of all possible conjunctions of bits in the input space, i.e., the space of possible rules.

4 EXPERIMENTS

We tested the learning model on standard planning domains from the 3rd International Planning Competition (IPC): Depots, ZenoTravel and DriverLog; a standard BlocksWorld domain; and Briefcase, a domain with conditional effects. Sequences of random actions and resulting states were generated from PDDL domain descriptions [9] and used as training and testing data.[2] Specific problems from the IPC were used to set the sizes of the initial states for each sequence. The actual initial states were generated at random using the IPC3 problem generator and a Briefcase state generator [6].

To determine error bounds on our results, we used 10 different randomly generated training and testing sets. Each training set consisted of 1000-20000 actions and matching state observations. Partial observability was simulated by randomly selecting a fraction (5-20%) of bits to retain in each state vector, and setting the remaining bits to *N*. Sensor noise at 10% was simulated by flipping each bit in the state vector with probability 0.1. Each test set was a fully observable, noiseless sequence of 2000 actions and observations. We measured the performance on our test sets by considering the fluents which our model predicted would change versus the fluents which did change, and calculating the balanced F-measure, the harmonic mean of precision and recall (*true positives/predicted changes* and *true positives/actual changes*, respectively). Selected results of the experiments are shown in Figure 2.

5 CONCLUSIONS AND FUTURE WORK

We have presented a method for learning deterministic action models which is fast, scalable and handles noise and partial observability of the world state. Furthermore, the error rate of the predictions made by the model is low. The speed, scalability and accuracy make the approach highly suitable for use in planning applications.

Additionally, our approach can learn conditional effects. Note that the success or failure of an action, which depends on its preconditions, is a form of conditional effect (the action has a null effect unless the preconditions are satisfied). Therefore action preconditions can also be learnt, if examples of action failures as well as action successes are provided.

A key step in future work will be to extract STRIPS-style rules from the sets of ordered pairings of entire states presently learnt by the model, so that the learning model can be integrated with standard planning software. We also plan to apply our method to intrinsically noisy and partially observable real-world robot environments.

ACKNOWLEDGEMENTS

This work was funded by the EU PACO-PLUS project (FP6-2004-IST-4-27657) and Edinburgh University Neuroinformatics DTC.

REFERENCES

[1] P. E. Agre and D. Chapman, 'Pengi: an implementation of a theory of activity', in *Proc. of AAAI*, pp. 268–272, (1987).
[2] E. Amir and A. Chang, 'Learning partially observable deterministic action models', *JAIR*, **33**, 349–402, (2008).
[3] S. S. Benson, *Learning Action Models for Reactive Autonomous Agents*, Ph.D. dissertation, Stanford University, 1996.
[4] R. E. Fikes and N. J. Nilsson, 'STRIPS: A new approach to the application of theorem proving to problem solving', *AIJ*, **2**, 189–208, (1971).
[5] Y. Freund and R. Schapire, 'Large margin classification using the perceptron algorithm', *Machine Learning*, **37**, 277–296, (1999).
[6] J. Hoffmann and B. Nebel. FF domain collection. `http://www.loria.fr/~hoffmanj/ff-domains.html`.
[7] M. Holmes and C. Isbell, 'Schema learning: Experience-based construction of predictive action models', in *NIPS 17*, pp. 585–562, (2005).
[8] R. Khardon and G. M. Wachman, 'Noise tolerant variants of the perceptron algorithm', *JMLR*, **8**, 227–248, (2007).
[9] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, 'PDDL - the planning domain definition language', Technical report, CVC TR-98-003, Yale, (1998).
[10] J. Modayil and B. Kuipers, 'The initial development of object knowledge by a learning robot', *Robot. Auton. Syst.*, **56**(11), 879–890, (2008).
[11] K. Mourão, R. Petrick, and M. Steedman, 'Using kernel perceptrons to learn action effects for planning', in *Proc. of CogSys*, pp. 45–50, (2008).
[12] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, 'Learning symbolic models of stochastic domains', *JAIR*, **29**, 309–352, (2007).
[13] F. Rosenblatt, 'The perceptron: a probabilistic model for information storage and organization in the brain', *Psych. Rev.*, **65**(6), 386–408, (1958).
[14] K. Sadohara, 'Learning of boolean functions using support vector machines', in *Proc. of ALT*, pp. 106–118, (2001).
[15] Q. Yang, K. Wu, and Y. Jiang, 'Learning action models from plan examples using weighted MAX-SAT', *AIJ*, **171**(2-3), 107–143, (2007).

[2] All data was generated using the Random Action Generator 0.5 available at `http://magma.cs.uiuc.edu/filter/`.