

Prediction in Dynamic Environment: Robocup Rescue Exploration

Andy Song
School of CS and IT, RMIT
University
GPO Box 2476V
Melbourne, Australia
asong@cs.rmit.edu.au

Lin Padgham
School of CS and IT, RMIT
University
GPO Box 2476V
Melbourne, Australia
linpa@cs.rmit.edu.au

Lawrence Cavendon
School of CS and IT, RMIT
University
GPO Box 2476V
Melbourne, Australia
lcavendon@cs.rmit.edu.au

ABSTRACT

In dynamic environments BDI agents typically involve an “Observe, Think, Act” cycle. However, an ability to predict environmental events and dynamics in advance allows an agent to begin to act earlier and potentially acting more efficiently. In a multi-agent setting, this may also help with more efficient distribution of resources. This paper presents a framework within a BDI architecture for predicting future developments in a dynamic environment and investigates the issue of learning from past observation. We evaluate the approach in the Robocup Rescue domain on tasks to predict the increase in intensity of fires. The results indicate that the framework could effectively be used to anticipate and plan ahead for future environment change. Furthermore strategies and issues of learning are explored in the study.

1. INTRODUCTION

Belief Desire Intention (BDI) agents are a popular class of agents which are able to balance proactive goal seeking behaviour and reactive behaviour which is responsive to the environment. These systems, examples of which include JACK [3], PRS [5], 3APL [7], are typically adaptive in the sense that they are able to recognise environmental changes and choose appropriate plans for the current context. However they do not typically include any *learning* component where the agent actually generates new knowledge which subsequently affects its behaviour.

Learning is a fundamental aspect of intelligence, and unsurprisingly there are many applications where it would clearly be advantageous if BDI agents could incorporate learning into their framework. Much of the work on learning in agents deals either with reinforcement learning focusing on how to act or on classification. The former focusses on learning an efficient action policy by trial and error based on observation, while the latter is for recognizing different patterns.

Our work in this paper focuses on *learning to predict*. Clearly if an agent is able to predict the likely evolution

of a given scenario, she is better able to make choices about how to act. This may involve acting so as to influence a preferred evolution of a course of events, or, more straightforwardly, initiating a plan or action which takes time to execute but which addresses a future state that is predicted to come about. We situate our exploration of learning about prediction of scenarios, within the Robocup Rescue domain, a simulated disaster scenario [8]. In particular we focus on learning for the fire brigade agents in the simulator. Being able to predict fire development in advance allows limited resources (i.e., fire-fighting agents) to be deployed efficiently and effectively.

In this paper, we demonstrate that fire brigade agents can successfully learn to predict the growth of fires based on information perceived by these agents the simulated world. We start by presenting the framework of agent learning, and the learning techniques that are used to develop the ability to successfully predict the likely evolution of a fire. Issues of learning are investigated in the study such as agent perspectives, the effect of agents’ position and the impact of number of agents with time factor.

2. THE FRAMEWORK

For the purposes of this paper, we focus only on fire-brigade agents, effectively focussing on a *homogeneous communicating multiagent domain* [18]. We do not at this point consider the issue of collaborative action, but concentrate solely on the learning and prediction task.

2.1 Learning Infrastructure Overview

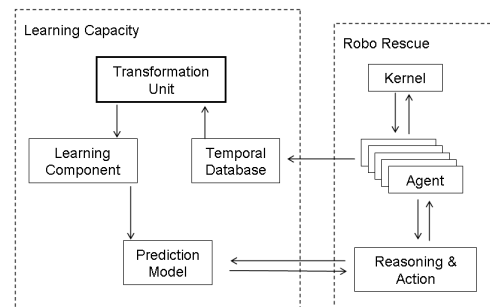


Figure 1: Structure of the Learning Framework

Figure 1 shows the architecture of the learning framework,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'07 May 14–18 2007, Honolulu, Hawaii, USA.
Copyright 2007 ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

which contains four basic components. The first is a temporal database where all relevant observed information is stored. Periodically the filtering and transformation component retrieves information from the data base, processes it, and provides it to the learning component, which in this case uses a neural network approach. The learning component produces a model of the environment as it changes over time. In this case the model is in the form of a trained neural net, which can then be queried by the agent.

The learning capability can either be a standalone facility, or agent, which obtains data and responds to queries from agents in the environment. Alternatively it can be a capability incorporated into each agent. Because of the nature of the application environment each agent will see only a limited number of fires. Consequently it makes sense to share the information from all agents in order to facilitate having a sufficiently broad base for learning. This could be done by having a coordinator agent which receives data, does the learning, and advises or coordinates individual fire-brigade agents. Alternatively agents could broadcast data to colleagues, and each do individual learning. In our current implementation we have chosen a centralised learning facility. In different application areas it is conceivable that an individual agent would itself collect sufficient data for learning, over a period of time.

We briefly describe each of the components in the learning framework, and outline how the agent would use the learned predictive environmental model.

2.2 The Temporal Database

The fire events are perceived by agents and stored in a temporal database. We use *TimeDB* [17] for this purpose, recording time-stamped information regarding the progression of fires. The temporal information stored at each cycle is the agent’s observations of fires, which are stored as tuples of $\langle \text{buildingid}, \text{time}, \text{fireintensity} \rangle$. In addition static information is stored regarding each building, its area, its construction type, and its neighbours. All data observed is stored, but this is then filtered prior to use.

2.3 Transformation and Filtering

The transformation component is responsible for pre-processing the data that is required before sending it to the learning component. The operations include:

- Choosing the closest (and therefore most accurate) agent’s data regarding a particular fire (identified by building id). The Robocup Rescue kernel allows perception from different distances; however, percepts from a closer distance are more accurate than those from further away. Hence, only the most accurate version of an observation is used to train the model;
- Filtering out data on fires in similar buildings (same area and building type). Oversampling data for a single point could result in bias in the learned model; this tactic is used to mitigate that potential problem;
- Calculating the rate of growth for each fire in the filtered dataset, based on the time-stamped data: rate of growth (across building area and type) is what the model actually learns to compute.

Fire intensity is defined as an integer in the range [1 . . . 7]: a value of 1 to 3 means the building is on fire with increasing

intensity, and a value of 7 means the building is completely burnt. The learning task is how to predict rate of intensity increase across different buildings given their characteristics, in particular, area.

2.4 The Learning Component

In our current implementation, a neural network is used as the technique for learning the prediction model, although this could be replaced by some other regression learner, such as a Support Vector Machine [4] or Decision Tree [15]. We use a neural network because they have been widely used in prediction tasks to forecast future state values, such as stock market prediction, sales forecasting and real estate prediction.

The architecture of the neural network we used, which is a typical three-layer fully connected feed-forward network. The input for each event is the building size for each fire chosen to be in the dataset, while the output is the rate of fire growth. The learned output in our neural network is actually a function of fire growth rate given building size and building materials, rather than a prediction of intensity at a time point.

$$\frac{\text{Intensity}(t_2) - \text{Intensity}(t_1)}{t_2 - t_1} = f(\text{Building Information})$$

A typical training process involves around 1000 epochs. Error is usually significantly reduced after 250 epochs. When there is no significant drop of *mean squared error* (MSE), the training is stopped. The trained neural network can then be used by the agents for prediction.

2.5 Reasoning With the Learned Model

Once the prediction model has been learned, in the form of a neural network, it can be used for providing information about expected rate of change of intensity, which can then be converted to a particular intensity for a future time point. The neural network can either be part of a learning coordinator and can be queried by messaging, or alternatively it can be provided to agents as a learned model, and queried internally. An agent can provide information about a specific building as input to the neural network and the rate of increase in intensity is returned; the agent can then use this information to calculate the fire intensity for a given future time.

Evaluation of the accuracy of the learned model is performed by comparing the discrepancy between intensity *predicted* for some future time, and intensity *actually observed* at that future point. This indicates the accuracy of the model, and whether it needs further training.

3. EXPERIMENTATION

In this section, experiments of generating and evaluating learned predictive model are reported.

3.1 Training and Test Datasets

The predictive model used in each experiment was trained from data points obtained as observations from agents randomly distributed throughout the simulated environment. All agents are fire-brigade agents whose firefighting function has been disabled. When an agent observes a building on fire, it records the building information, its own ID and location, and then observes and periodically sends the fire

intensity. Datasets were collected by running the simulator for approximately 300 minutes each time, resulting in about 2000 observations each time run. The transformation unit removed observations from all but the closest agent to a given fire, as described earlier, reducing the number of training data points to approximately 50 each time.

We collected three independent test datasets, from multiple runs of the simulator, so as to allow cross-validation of the evaluation. Each dataset contains between 35 and 46 fire-observations from different buildings.

3.2 Single Agent Perspective

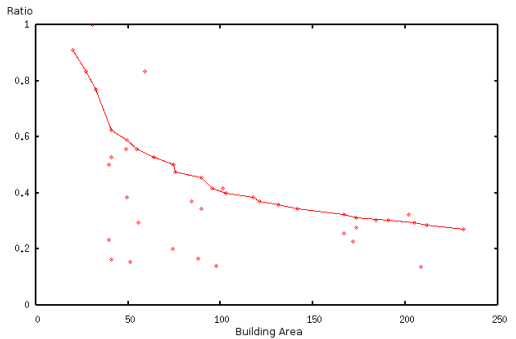


Figure 2: Learning from a Single Agent’s Perspective

Figure 2 shows the test accuracy of a model training based on data collected by only one agent. The graph shows the correlation between the building size (the X axis) and the rate of fire growth (the Y axis). In the graph the points on the solid line are the actual fire growth rates on buildings with different sizes. The scattered points are rates predicted by the neural network on the same test data. Different experiments using single agent result similar outcomes. It suggests that learning by single agent is not a good approach. It is partially due to agent’s limited view. It is difficult for one agent to see a wide range of buildings on fire. Therefore it is difficult for the learning process to generalize an accurate function due to limited examples.

3.3 Multi Agent Perspective

Figure 3 shows the results of an experiment in which the prediction model was trained on 24 data points (after filtering) from observations of 10 randomly distributed agents. The graph compares the prediction model to actual observations from one of the test datasets: for each data point in the test dataset, the learned model predicts the intensity rate for that building, given its area.

As can be seen, the predicted rates are closely aligned to the values in the test dataset (which were obtained directly from the simulator). This alignment is statistically significant: for the first set of results, the Pearson coefficient r is 0.99, indicating a very strong correlation between predicted values and actual values. The other two test sets give similar outcomes, each with correlation 0.98. Table 1 shows the Mean Squared Error (MSE) against the three test sets.

Compare to single agent perspective, the multi-agent approach shows a clear advantage. This result suggests that predicting in Robocup Rescue domain should use multiagent learning which is “learning that is done by several agents

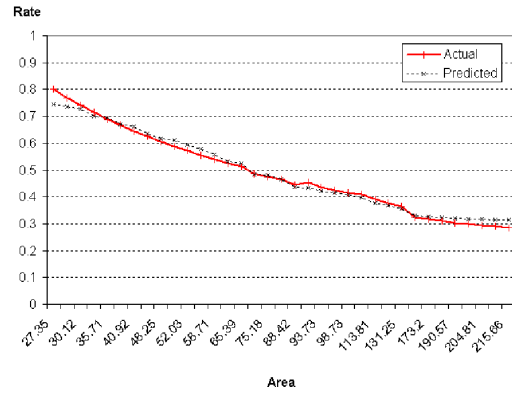


Figure 3: Learning from Multi Agents’ Perspective

| | |
|--------------------|---------|
| Test Set 1 | 2.87E-4 |
| Test Set 2 | 2.12E-3 |
| Test Set 3 | 5.26E-3 |
| No. of data points | (24) |

Table 1: Mean Squared Error on Three Test Sets

and that becomes possible only because several agents are present” [20].

3.4 Buildings with Different Materials

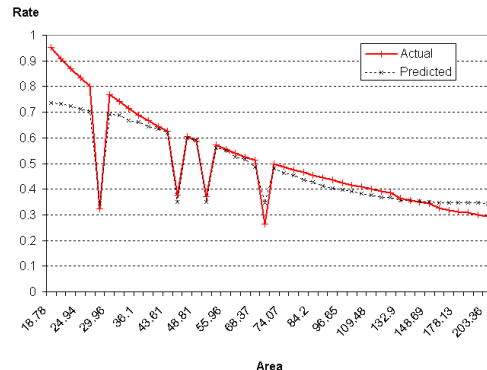


Figure 4: Mixing Buildings With Different Materials

The training data in previous experiment contains same type of buildings. Separating buildings with different material simplifies the learning tasks. Of course it is feasible only if there are not many varieties in building types. This part of experiments show that it is still possible to train on mixed type of buildings. Figure 4 shows one of the test results. Due to the different building materials, the actual rate is not a smooth curve any more. The sharp drops in the graphs are the minority steel frame and concrete buildings. The neural network can still manage to give a reasonable prediction also the match between predicted values and the actual values is not as good as that of using only one type of building.

4. DISCUSSION

In this section we discuss a number of issues to do with quality and quantity of training data, in order to provide greater understanding of the nature of the data needed for effective predictive learning in this scenario.

4.1 Positioning of Data Collection Agents

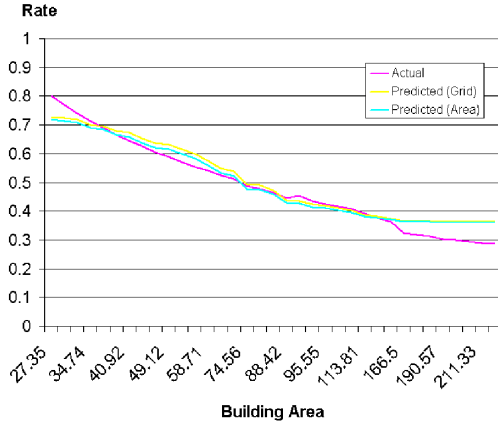


Figure 5: Distributing Agents in Grid

In stead of randomly distributing agents across the simulated city, we distributed 10 agents evenly in grid to see whether such distribution would bring any benefits to learning. The learning outcome is presented in figure 5 on which the line labeled “Predicted (Grid)” is the prediction by training on grid approach, while the line labeled “Predicted (Area)” shows that by method discussed in section 3.3. The result suggests that there is no obvious improvement by distributing agents evenly.

4.2 Number of Agents with Time Factor

In order to investigate to what extent learning would be possible during the timeframe of a single simulation, we ran experiments to collect training data with differing length of simulation time, and differing numbers of agents collecting data. We investigate whether it is possible to collect data and perform learning within a same simulation.

Table 2 presents the results of running the simulator for 100, 150 and 200 minutes with different numbers of agents, 10, 50 and 100. The data collected at each run are used for training (with filtering by the Transformation component). The MSE of each trained model on three test data sets are listed in the corresponding cell of Table 2. Underneath the three MSEs, a number in brackets indicates how many patterns are actually collected. The data collected before 100 minutes are not shown, because no reasonable learning was achieved based on these data.

In general, the longer the time for data collection, the smaller the resulting errors although it is not strictly the case always. The results at 100 minutes are slightly worse than those at 150 minutes and at 200 minutes. However the difference is small, which suggests that collecting data for 100 minutes should give a reasonable prediction.

As discussed in Section 3.3, data is collected by multiple agents to ensure sufficient data points. However more agents do not necessarily result in better learning. We can see from

| | 100 min | 150 min | 200 min |
|------------|------------------|--------------------|-------------------|
| 10 Agents | 8.40E-4 | 7.11E-4 | 7.30E-4 |
| | 3.0E-3 | 2.99E-3 | 2.92E-3 |
| | 6.17E-3 (8) | 6.098E-3 (12) | 6.07E-3 (13) |
| 50 Agents | 9.42E-4 | 5.3 9E-4 | 5.93E-4 |
| | 3.29E-3 | 2.75E-3 | 2.88E-3 |
| | 6.34E-3 (8) | 5.8 1E-3 (14) | 5.93E-3 (18) |
| 100 Agents | 9.15E-4 | 8.8 3E-4 | 6.371E-4 |
| | 3.25E-3 | 3.39E-3 | 2.98E-3 |
| | 6.3E-3 (8) | 6.4 4E-3 (14) | 6.0E-3 (18) |

Table 2: Impact of Number of Agents with Time Factor

Table 2 that 50 agents gives better coverage than 10 agents, perceiving more fire events at 150 and 200 minutes (14 vs 12 and 18 vs 13). However the prediction performance over the three test data sets is quite similar. 100 agents did not result in more patterns than 50 agents, as the number of fires is not affected by the number of agents, and only the best data is used.

5. LEARNING IN ROBOCUP RESCUE

While there is much work on learning and action-selection in agent systems, particularly in the area of reinforcement learning, there is very little work on learning in the context of BDI agent architectures. [6] and [14] describe frameworks for incorporating learning capabilities into BDI agents, and [11] use a decision tree learner to learn plan selection strategies. Our approach is to learn a model of the dynamics of the environment, which can then be incorporated into the plan-selection to react to future developments rather than simply current conditions.

A number of learning tasks have been addressed within the Robocup Rescue domain itself, using a variety of techniques. [12] identified a number of potential learning tasks within this domain, including: learning how to use communication channels more efficiently; how to manage a disaster better; and being able to anticipate other agent’s action and the environmental changes to form long term plans—our approach addresses the final one of these.

Fire management in RoboCup Rescue has been addressed by a number of authors. [9] learns to predict the time it takes fire to spread between buildings. They also use a C4.5 decision tree to classify civilians into “survivors” and “victims”, and use regression trees (CART) to predict the lifetime of civilians, in order to maximise survival rate. [2] propose a layered learning structure to predict fire spread between buildings in one layer and the effectiveness of firefighting in another. A third layer then uses fuzzy logic to decide the priority in which to extinguish buildings.

[13] learns to perform task allocation, enabling rescue agents to learn how to choose a buliding for action so as to maximise the rescue outcome; learning features include buiding material, fire intensity, building size, and number of agents in the vicinity. [1] also consider the task of distributing fire-fighting agents, using neural reinforcement learning to obtain an optimal priority scheme in group population

management; this involved using a neural network to learn to predict ratios of dying civilians.

[10] used Q learning for police agents to determine optimal choices in clearing roads. They use a “virtual center” to learn, coordinate agents, and control decision making. [16] also learn action strategies for police agents, learning priorities for different activities.

6. CONCLUSION AND FUTURE WORK

A learning framework based on neural network is presented. This study shows that it is suitable for agent to learn from past observation to predict future states. Due to the large amount of information required it makes sense for agents to share their observations so that learning can be done on the basis of multiple agents’ observations. One approach is to centralize the observation reports from agents by a coordinator agent, who can store and appropriately filter the information, perform the learning, and then plan and coordinate the assignment of agents to fires based on the output of predictive model.

The information predicted so far, while useful, is fairly simple. More complex information for prediction would be the rate at which fire intensity changes as agents are involved with fighting the fire. This information is more realistic for agents to collect (as they would not generally stand and watch buildings burn in order to obtain data for learning!), as well as being more useful in guiding the agents’ actions, as what is desired is an assignment of an appropriate number of firefighters to control the fire.

In a disaster scenario such as robocup rescue there is not just a single fire, but many. A more difficult, but useful prediction task is to predict the rate at which new fires are going to break out, in order to try and ensure sufficient numbers of fire brigades available to manage them.

This work is also motivated by work in other areas. For example, Thangarajah et al. [19] describe the importance of recognising complex situations that have significance for a particular domain, such as intense storm activity. Predictive ability could enable agents to not just recognise when a situation has arisen, but predict that such a situation is impending, and either take action to avoid it, or, as in the meteorological domain, to send warnings and thus avoid damage.

Operating in dynamic worlds, predicting how the outside environment is likely to evolve is extremely useful for agents to plan and act effectively. The current work is a small start.

Acknowledgment

The authors would like to thank the Agent Oriented Software Group and the Australian Research Council for supporting this research project “Learning and Planning in BDI Agents” (number LP0560702).

7. REFERENCES

- [1] Saman Ampirpour Amraii, Babak Behsaz and Mohsen Izadi. S.o.s 2004: An attempt towards a multi-agent rescue team. In *Proc. 8th RoboCup Int’l Symposium*, 2004.
- [2] Ali Bitaghsir, Fattaneh Taghiyareh, Amirhossein Simjour, Amin Mazloumian and Babak Bostan. Uteternity’s team description: Layered learning in robocup rescue simulation. In *Proc. 8th RoboCup Int’l Symposium*, 2004.
- [3] Paolo Busetta, Ralph Rönquist, Andrew Hodgson and Andrew Lucas. JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty. Ltd, 1998. <http://www.agent-software.com>.
- [4] Cortes and Vapnik. Support-vector networks. *MACHLEARN: Machine Learning*, Volume 20, 1995.
- [5] M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proceedings of the IEEE Special Issue on Knowledge Representation*, Volume 74, pages 1383–1398, 1986.
- [6] A. Guerra-Hernandez, A. E. Fallah-Seghrouchni and H. Soldano. Learning in bdi multi-agent systems. In *CLIMA IV*, pages 218–233, 2004.
- [7] Koen V. Hindriks, Frank S. De Boer, Wiebe Van der Hoek and John-Jules Ch. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, Volume 2, Number 4, pages 357–401, 1999.
- [8] H. Kitano, S. Tadokor, H. Noda, I. Matsubara, T. Takhasi, A. Shinjou and S. Shimada. Robocup-rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In *In Proc. of the IEEE Conference on Systems, Men, and Cybernetics*, 1999.
- [9] Alexander Kleiner and Michael Brenner. Resq freiburg: Team description and evaluation. In *Proc. 8th RoboCup Int’l Symposium*, 2004.
- [10] Eslam Nazemi, Mohammad Ali Fardad and Mohammad Mehdi Saboorian. Message management system in sbce_saviour team. In *Proc. 8th RoboCup Int’l Symposium*, 2004.
- [11] A. Nguyen and W.R Wobcke. An adaptive plan-based dialogue agent: Integrating learning into a BDI architecture. In *Proc. AAMAS*, pages 786–788, 2006.
- [12] Sébastien Paquet. Learning coordination in RoboCupRescue. In Yang Xiang and Brahim Chaib-draa (editors), *Proc. Canadian Conference on AI*, pages 627–628, 2003.
- [13] Sébastien Paquet, Nicolas Bernier and Brahim Chaib-draa. From global selective perception to local selective perception. In *AAMAS*, pages 1352–1353, 2004.
- [14] T. Phung, M. Winikoff and L. Padgham. Learning within the bdi framework: An empirical analysis. In R. Khosla, R. Howlett and L. Jain (editors), *Knowledge-Based Intelligent Information and Engineering Systems, Part III*, pages 282–288, 2005.
- [15] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.
- [16] Maziar Ahmad Sharbafi, Caro Lucas, Abolfazel Toroghi Haghighat, Omid AmirGhiasvand and Omid Aghazade. Using emotional learning in rescue simulation environment. *Transactions On Engineering, Computing and Teachnology*, Volume 13, May 2006.
- [17] Andreas Steiner and Moira C. Norrie. Implementing temporal databases in object-oriented systems. In *Database Systems for Advanced Applications*, pages 381–390, 1997.

- [18] Peter Stone and Manuela M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, Volume 8, Number 3, pages 345–383, 2000.
- [19] John Thangarajah, Lin Padgham and Sebastian Sardina. Modelling situations in intelligent agents. In *Proc. AAMAS*, Hakodate, Japan, May 2006.
- [20] Gerhard Weiß. Distributed reinforcement learning. *Robotics and Autonomous Systems*, Volume 15, Number 1-2, pages 135–142, 1995.