LEARNING BY OBSERVATION IN SOFTWARE AGENTS

Paulo Costa[1], Luis Botelho[1]
[1] *Instituto de Telecomunicações/ISCTE-Instituto Universitario de Lisboa, Lisbon, Portugal.*
{*paulo.costa, luis.botelho*}*@iscte.pt*

Keywords: machine learning, learning algorithms, learning by observation, software image, software agents

Abstract: In a society of similar agents, all of them using the same kind of knowledge representation, learning with others could be achieved through direct transfer of knowledge from experts to apprentices. However, not all agents use the same kind of representation methods, hence learning by direct communication of knowledge is not always possible. In such cases, learning by observation might be of key importance. This paper presents an agent architecture that provides software agents with learning by observation capabilities similar to those observed in superior mammals. The main contribution of our proposal is to let software agents learn by direct observation of the actions being performed by expert agents. This is possible because, using the proposed architecture, agents may see one another.

1 INTRODUCTION

Learning by observation is one of the most powerful socialisation mechanisms for superior mammals and also one of the most important means of knowledge acquisition (Ramachandran, 2003; Bandura, 1977; Meunier et al., 2007). The capacity to observe and imitate the movements of others is among the least common and most complex forms of learning (Moore, 1992). Research in neurology and psychology shows that learning by observation may well be one of the causes of the exponential growth of human capabilities in the last centuries (Ramachandran, 2006).

Learning by observation can be classified under the human and superior animals social interaction mechanisms. Bandura's social learning theory emphasizes the advantages of learning by observation (Bandura, 1977). Dautenhahn (Dautenhahn, 1994) claims that some intellectual capacities evolve out of the social domain. Animals and humans take benefit from the experiences of others by learning what they observe from them.

This provides motivation for further research on adapting a similar learning mechanism for software agent societies. Learning by direct observation of the expert agent performing its actions (as opposed to merely relying on observing their effects) can be advantageous in situations where the effects of those actions are not directly visible in the environment (e.g. communication between agents), when the representation of world states requires too much memory making it impossible to know the effects of all actions (e.g. social simulations with multiple interactions), in situations in which the same effects could be achieved by different alternative actions but using one of them is clearly better than using others (e.g. using a set of sums instead of a simpler multiplication), and especially when the agent does no know the effects of its actions (e.g. virtual character manipulation).

Learning by observation is also advantageous for knowledge transmission between agents developed with different programming languages and also between software and robot agents. Using learning by observation it is possible to make robot agents learn by observing a software simulated version of them.

We propose a software agent architecture providing learning by observation capabilities similar to those of superior mammals. This architecture allows apprentice agents to acquire control rules through the observation of an expert agent in action. In this paper, we focus on the learning by observation algorithm that is part of the proposed architecture. The algorithm depends on the agent software image (Costa and Botelho, 2011) as a means to identify similar agents and to observe those agents as they act. The observed agent software image provides apprentices with the necessary training sequences for their learning algorithms. It also supplies the necessary means to evaluate the acquired knowledge.

We have tried two learning by observation approaches. The memory based approach uses the observed data to create sequences of actions that are

applied to specific circumstances. The mirror based approach uses machine learning algorithms, trained with data from expert observations, to determine the correct behaviour for the perceived environment state. Apprentices take the observed expert behaviours as if they were their own, enabling an easier identification of the observed actions.

Since our memory based approach has not been completed, the experimental results focus on the mirror based approach. These results show that the best suited algorithms for the mirror based approach are KStar (classification algorithm) and NNGE (rule association algorithm). Results also show that after a few observations, apprentices are able to learn the correct behaviours for a large set of environment states. The results also seem to show that, for certain kinds of problems, learning by observation is advantageous over other learning methods.

The next section presents a survey of research on learning by observation. Section 3 presents a description of the learning by observation algorithm. The results of initial tests made on the learning algorithm are presented in section 4. Section 5 presents conclusions and future work.

2 RELATED WORK

Learning is an essential characteristic of intelligent beings. A computer program is able to learn if its performance on a set of tasks improves with experience (Mitchell, 1997). Learning algorithms can be organized in two types: supervised learning and unsupervised learning.

Supervised learning creates mappings between inputs and outputs, whose correct values are provided by a supervisor, which is usually a person. Supervisors can interact directly with the learner agent providing the necessary training sequences for the learning algorithm or building a reward system that compensates the agent each time it chooses the correct action. Unsupervised learning determines how a dataset is organized. These algorithms try to find regularities in the input data to extract knowledge from them (Alpaydin, 2004).

Several authors (Argall et al., 2009; Billard and Dautenhahn, 1999) define learning by observation as a subset of supervised learning, where policies are generated by observing, retaining and replicating the behaviour executed by an expert. Meunier and his colleagues (Meunier et al., 2007) proved that the use of learning by observation in rhesus monkeys improved the learning speed when compared with other types of learning such as trial and error.

The application of learning by observation techniques in robot and software agents opens the learning process to experts other than robot or software agents. It enables the creation of an intuitive communication medium between humans and computers (Argall et al., 2009).

Research in learning by observation is usually associated to robotics. Since software agents are usually not visible to themselves and to others, they are mainly restricted to observing the effects of actions, instead of the actions themselves (Kerstin et al., 1999). Software agent learning algorithms are often directed to the use reinforcement learning techniques, which in turn are learning methods that rely exclusively on environment observations.

One of the main causes of the lack of learning by observation of actions in software agents is the question on how to represent those actions when they have no physical body to make them visible (Kerstin et al., 1999). To overcome this problem, we have developed and integrated the software image into our agent architecture (Costa and Botelho, 2011). The software image is a mechanism through which software agents can observe their bodies and actions.

According to Argall and her colleagues (Argall et al., 2009) two aspects must be taken into account when building a learning by observation solution: gathering and interpreting examples from the expert agent and deriving a policy from those examples. The software image takes care of the first aspect, the proposed learning algorithm relates to the second aspect. Policy derivation can be achieved through several approaches. Our approach uses classification algorithms and policy representation as a sequence of behaviours.

Classification algorithms categorize and group similar perceptions and behaviours. In these cases the policy is derived by mapping perceptions to behaviours. This is best suited when environment states change at random and behaviours do not depend on each other (i.e. learning a package routing algorithm).

Representing the policy as a sequence of behaviours provides apprentices with the necessary steps to go from one state to the goal state. It is best suited for situations where behaviours are correlated and a chain of events can be determined (i.e. learning a sort algorithm).

Other approaches for learning by observation take inspiration from neurological research of brain structures such as the mirror neurons. They allow observers of some action to feel almost the same as if they would if they have performed it, providing the means for easy action identification (Ramachandran, 2006; di Pellegrino et al., 1992; Rizzolatti et al., 1996).

According to Demiris and Hayes (Demiris and Hayes, 2002), research in human brain and mirror neuron activation suggests that learning by observation allows the apprentice to put itself in the experts place. To understand the observed behaviour, apprentices generate several alternatives before the observation process is finished, allowing them to improve the quality of response.

Based on these facts, Demiris and Hayes proposed a biologically inspired computational model for learning by observation. They used forward models to create estimates for specific behaviours regardless of their source (observed or generated by the apprentice). Forward models allow the apprentice to predict the next state without actually performing the behaviour. In a sense, they allow the apprentice to generate and test a set of possible behaviours without affecting the environments state (Demiris and Hayes, 2002).

A similar approach is used in our learning algorithm. A mirror mechanism provides agents with the ability to generate a set of possible behaviours for a given state, retrieved from observation or from the apprentices perception. Instead of using estimations and forward models, the possible behaviours are evaluated, providing the apprentice with quality measures.

Evaluation is an important feature in learning by observation since it allows the apprentice to know its ability to accomplish the goal. Evaluation can be assisted or unassisted. Milstein and her colleagues realized that, unlike humans, chimps and gorillas do not get assisted by their parents when learning new tasks. They have to resort to trial and error to perfect the learnt skills (Milstein et al., 2008).

The use of guidance from an expert is a singular feature in learning by observation amongst humans. This interaction allows the apprentice to ask for guidance whenever it has doubts on the actions to take. This provides us great advantages over trial and error. Our learning algorithm allows the use of both assisted and unassisted evaluation, since assisted evaluation requires the use of specialized experts, the teachers, which might not be always available.

3 LEARNING BY OBSERVATION ALGORITHM

The learning by observation algorithm follows a modular design, operating in two stages: the learning stage and the execution stage. The learning stage includes observation, storage, reflection and evaluation of behaviours. The execution stage includes the application and evaluation of the learnt skills.

Each of the learning algorithm's modules addresses a specific task in the learning process. Figure 1 shows how these modules are arranged and how information flows at each stage.

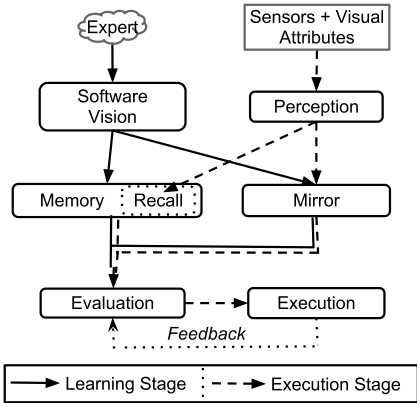

Figure 1: The learning modules.

As figure 1 shows, the memory, mirror and evaluation modules participate in both learning and execution stages. The learning stage begins in the software vision module with the location and observation of a similar expert. The execution stage begins in the apprentice's sensors with the acquisition of the apprentice's perception.

Evaluation represents the last step for both learning and execution stages. This is an important feature since it ensures the apprentice's learning quality. Evaluation is a continuous process, allowing the apprentice to go back to the learning stage whenever it is necessary.

The evaluation process influences the apprentice's confidence level. Confidence is an internal quality measure that allows switching between the learning and execution stages. When confidence is under a certain threshold the algorithm is on the learning stage, above that threshold the algorithm is on the execution stage.

3.1 The Learning Stage

The software vision module provides apprentices with the necessary tools to observe experts. It makes use of a special property that allows experts and apprentices to describe their capabilities and perceptions, the software image (Costa and Botelho, 2011). The vision module focuses the apprentices attention on experts that can be imitated (whose static image is contained in the apprentice's static image). The expert's dynamic image provides apprentices with the expert's behaviour and perception as snapshots.

Software vision converts this information into ap-

prentice experiences by replacing all references to expert structures in the snapshot with references for the corresponding apprentice's structures. This conversion allows apprentices to use the experts state description and behaviour as if they were their own.

Software vision is able to read both current and historical information on what the agent is doing and what were the conditions holding just before it has done each action. Apprentices are able to load a limited amount of snapshots from earlier times. Each time an apprentice starts observing a new expert it reads its historical data and, only after that, it starts collecting snapshots of on-going behaviour.

Two approaches are used to handle data obtained from observation of an expert's software image. The memory approach focuses on storing and recalling the observed snapshots according to their place in time. The mirror approach focuses on making the act of observing an expert to produce the same effects as preparing for executing those actions.

The memory approach uses the memory module to store observed snapshots and provide solutions for perceived environment states. As it is the case in superior mammals procedural memory, the memory module stores each snapshot as a chain of steps. These are later recalled, by the recall mechanism, and used as reference paths for moving from one environment state to another. Thus the memory module is able to provide a set of possible behaviours for the currently perceived state through association with previously executed behaviours.

The mirror approach uses the memory module to store observed snapshots and the mirror module to provide solutions for perceived environment states. For the mirror module, like in mirror neurons, the act of observing an expert produces the same effects as preparing for executing those actions. The mirror module uses a collection of machine learning algorithms, such as KStar (classification), ID3 (decision trees), Naive Bayes (Bayesian networks) and NNGE (rule association), that are trained with the data stored in the memory module to build a list of possible behaviours from an environment state. Environment states can come from both agent perception or expert observation, as they are treated in the same way. Developers are free to use any of the algorithms but, as section 4 shows, KStar and NNGE are the best choices.

The apprentice's choice between these two approaches involve the use of a weight mechanism. The memory and mirror modules are fitted with weight factors that enhance the apprentices adaptability to different learning circumstances. Each time a module produces a solution that, through evaluation, is proven to be the appropriate one, the module's weight is increased.

In the learning stage, evaluation happens each time the apprentice makes an observation. For each observed snapshot, the apprentice produces a solution for the environment state described in the snapshot. This solution is compared, in the evaluation module, with the behaviour provided by the snapshot to determine if the apprentice is making the correct choices.

Apprentice confidence increases whenever the apprentice's solution matches the behaviour provided by the snapshot. When that is not the case, the apprentice's confidence decreases.

3.2 The Execution Stage

In the execution stage the apprentice's perception is used as input for the mirror module and the recall mechanism. Each of these modules produce a solution according to the underlying approach (mirrror or memory approach). The execution module picks the most fitted solution from the module with the highest weight value. If the solution has a positive evaluation the execution module carries out with execution.

The execution module is only active when the confidence level is above a certain threshold. Bellow that threshold the agent is unable to perform any action, it only has the ability to observe. Execution makes the necessary arrangements to call the actions required for the solution. Throughout execution, a special mechanism collects information on the problems and achievements that might be encountered. This mechanism is responsible for providing the feedback to the evaluation module.

Evaluation in the execution stage follows two directions depending on the existence of a special kind of expert, the teacher. If teachers are available the apprentice is able to evaluate directly the provided solution through teacher appraisal. The apprentice asks the teacher if its solution is correct and if the teacher answers positively the apprentice's confidence increases. If the answer is negative the apprentice's confidence decreases and the solution is not executed.

When no teachers are available, evaluation only produces an outcome after execution. Whenever a problem is found when executing the solution or the apprentice realizes that it has made a step back (e.g. needs to re-achieve a sub-goal), evaluation decreases the apprentice's confidence level. If, by any chance, the execution provides some type of reward, like for example the achievement of a sub-goal, evaluation increases the apprentice's confidence. In all other cases, confidence cannot be changed since it is not possible to assume if the solution was the appropriate one.

4 EXPERIMENTAL RESULTS

In this section, we describe an experiment on the mirror approach that tests the capabilities of several machine learning algorithms in learning by observation environments. These algorithms are used by the apprentice's mirror module to provide solutions for the perceived environment states. With this experiment we intend to see what types of algorithms are best suited for the mirror approach and how agent complexity affects the algorithm's performance.

A test framework was developed to allow the creation of small test scenarios involving a single expert and a single apprentice agent. In this framework, the environment state is described by four variables with a limited set of possible values: two numerical, one string and one enumerate. Expert agents react to controlled changes in the environment state, executing specific behaviours when specific combinations of the four environment variables are presented.

Table 1 presents the agent features in two of the tested scenarios. They allow us to see how the number of rules influences the learning efficiency. The complexity of agent behaviours rises with the number of rules used to describe them. When the number of rules increases, an increase in the time required to observe an expert is expected, since learning by observation works efficiently when apprentices observe a wide range of possibilities.

Table 1: Agent features for the test scenarios.

Scenario	Parts	Sensors	Actions	Rules
1	1	2	4	4
2	1	3	5	10

Both scenarios present a total of 100 apprentice observations (meaning the expert is presented with 100 different environment states). For each observation the apprentice stores the observed snapshot and calculates the proper behaviours for a set of ten different environment states. Apprentice behaviours are calculated by a specific algorithm in the mirror module. The algorithm makes use of the previously observed snapshots as train sequences.

Apprentice behaviours are compared with the correct responses to determine how the number of observations influences the apprentice's accuracy rate. The accuracy rate determines the number of correct behaviours of the ten performed. Figure 2 presents the apprentice's error rate for the best two algorithms, KStar and NNGE, in both scenarios. Although Bayesian networks are widely used for these kind of learning solutions, the tested algorithm (Naive Bayes) presented a lower performance on the tested scenarios.

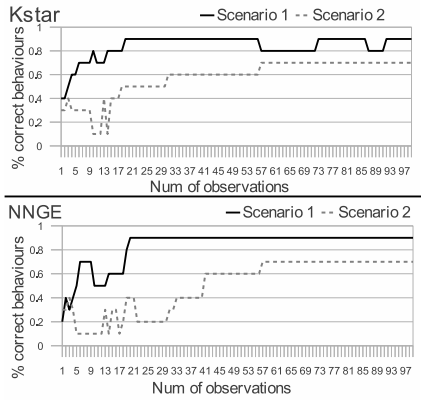

Figure 2: Apprentice error rate for the tested scenarios.

Figure 2 shows that using fewer rules allows apprentices to achieve high accuracy rates with a smaller number of observations. As another important finding, on both scenarios, the accuracy rate stabilizes after achieving a certain value, which is lower in the second scenario. For 100 observations, it is impossible to achieve full accuracy. Even though the apprentice is able to perform correctly in most of the cases, under certain circumstances incorrect behaviours are still being performed. Further research on this matter revealed that under these circumstances, the algorithm's confidence (the probability values of possible behaviours) is quite low, meaning the algorithm's decision capacity is prone to errors.

One possible reason for the lack of a complete accuracy after 100 observations may be the small range of observed behaviours. Some of the environment states presented to the apprentice may require the execution of yet unobserved behaviours. This results in the non inclusion of the correct behaviour in the algorithm's list of possibilities.

Figure 2 also shows that KStar performs a little better than NNGE but, as we increase the number of observations, the accuracy rate of the KSTAR decreases slightly. This decline may be due to the fact that the KStar algorithm uses the similarities among the train sequences to calculate the list of possible behaviours. When faced with sets of similar environment states that give rise to different behaviours, the initial observations may lead to mistakes that are fixed with subsequent observations.

Although only a small set of the learning algorithm was focused on these tests, we can say that the apprentice is able to learn almost all the expert's behaviours in a small amount of time. When compared with other learning techniques that need long training periods such as reinforcement learning, results show that learning by direct observation of agent actions al-

lows software agents to achieve results in a faster way.

Further tests proved that, in particular cases, observing agent actions is the only way to understand expert behaviour. If behaviours produce no effects in the environment (affecting only the agent and its internal state), relying on the changes in the environment makes learning ineffective. Testing our learning approach in such a scenario produced similar results to those presented on figure 2, were all behaviours produce visible effects in the environment. This proves the apprentice was able to learn the expert's behaviour even though there were no visible effects. These results show us that learning by observation can be of use to software agents.

5 CONCLUSIONS AND FUTURE WORK

As shown by the experimental results on section 4, the proposed learning algorithm allows agents to learn new skills within a small number of observations. As opposed to reinforcement learning, apprentice agents do not need to test all possibilities to determine the correct behaviour for a given situation. Apprentices are able to see expert's behaviours, reducing the number of iterations necessary for the initial learning stage.

After completing the memory approach, further tests are needed to determine how it interacts with learning and how it influences the algorithm's performance. The learning algorithm also needs to be compared with other learning approaches in scenarios where observation of expert actions is of key importance. This will enable us to see how favourable is learning by observation when compared with other learning approaches.

ACKNOWLEDGEMENTS

This paper reports PhD research work, for the Doctoral Program on Information Science and Technology of ISCTE-Instituto Universitario de Lisboa. It is partially supported by Fundação para a Ciencia e a Tecnologia through the PhD Grant number SFRH/BD/44779/2008 and the Associated Laboratory number 12 - Instituto de Telecomunicações.

REFERENCES

Alpaydin, E. (2004). *Introduction to machine learning*. MIT Press, Cambridge Mass.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). *A Survey of Robot Learning from Demonstration*, volume Robotics and Autonomous Systems, pages 469–483. Elsevier.

Bandura, A. (1977). *Social learning theory*. Prentice Hall.

Billard, A. and Dautenhahn, K. (1999). Experiments in learning by imitation - grounding and use of communication in robotic agents.

Costa, P. and Botelho, L. (2011). Software image for learning by observation. *Proceedings of 15th Portuguese Conference on Artificial Intelligence (EPIA 2011)*. Accepted.

Dautenhahn, K. (1994). Trying to imitate - a step towards releasing robots from social isolation. *Proceedings of From perception to action conference*, pages 290–301.

Demiris, J. and Hayes, G. (2002). Imitation as a dual-route process featuring predictive and learning components. a biologically plausible computational model. In Dautenhahn, K. and Nehaniv, C., editors, *Imitation in animals and artifacts*, pages 327–361. Cambridge, MIT press edition.

di Pellegrino, G., Fadiga, L., Fogassi, L., Gallese, V., and Rizzolatti, G. (1992). Understanding motor events: a neurophysiological study. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 91(1):176–180.

Kerstin, T. Q., Dautenhahn, K., and Nehaniv, C. L. (1999). The essence of embodiment: A framework for understanding and exploiting structural coupling between system and environment.

Meunier, M., Monfardini, E., and Boussaoud, D. (2007). Learning by observation in rhesus monkeys. *Neurobiol Learn Mem*, 88(2):243–8.

Milstein, M., Linick, S., Lonsdorf, E., and Ross, S. (2008). A comparison between chimpanzee (pan troglodytes) and gorilla (gorilla gorilla gorilla) social tolerance at an artificial termite mound. *AMERICAN JOURNAL OF PRIMATOLOGY*.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Moore, B. (1992). Avian movement imitation and a new form of mimicry: tracing the evolution of a complex form of learning. *Behaviour*, 122:231–263.

Ramachandran, V. S. (2003). *The emerging mind: the Reith Lectures 2003*. Profile Books.

Ramachandran, V. S. (2006). Mirror neurons and imitation learning as the driving force behind "the great leap forward" in human evolution. *Edge Foundation*.

Rizzolatti, G., Fadiga, L., Gallese, V., and Fogassi, L. (1996). Premotor cortex and the recognition of motor actions. *Brain research. Cognitive brain research*, 3(2):131–141.