Using a Case Base of Surfaces to Speed-Up Reinforcement Learning

Chris Drummond

Department of Computer Science, University of Ottawa
Ottawa, Ontario, Canada, K1N 6N5
cdrummon@csi.uottawa.ca

Abstract. This paper demonstrates the exploitation of certain vision processing techniques to index into a case base of surfaces. The surfaces are the result of reinforcement learning and represent the optimum choice of actions to achieve some goal from anywhere in the state space. This paper shows how strong features that occur in the interaction of the system with its environment can be detected early in the learning process. Such features allow the system to identify when an identical, or very similar, task has been solved previously and to retrieve the relevant surface. This results in an orders of magnitude increase in learning rate.

1 Introduction

One important research issue for case based learning is its combination with other learning methods. As Aamodt and Plaza [1] point out, generally the machine learning community aims to produce "a coherent framework, where each learning method fulfills a specific and distinct role in the system." This paper discusses one such approach, combining case based and reinforcement learning. An advantage of the latter method is that it learns even when the information available is very limited. It requires only knowledge of its present state and infrequent real values of reward to learn the actions necessary to bring a system to some desired goal state. As often occurs in achieving this level of generality, the learning rate is slow. It is important then to exploit the results of prior learning and it would be a definite advantage to quickly recognise that the task, or a similar one, has been solved previously.

This paper explores the idea of storing the results of learning in a case base. A critical issue, as in most case based systems, is how to recognise and index these cases. The central intuition of this work is that there are strong features in an environment. Here "strong" means that the features are stable and easy to recognise accurately early in the learning process. The environment here includes the system. The features are not "in the world" but in the system's interaction with the world. This interaction can be represented by a surface, a multi-dimensional function. For the notion of strong features to be effective, they must largely dictate the shape of this surface. If the features differ by a small amount one would expect the surface to differ by a small amount. Research [9] has shown that a function can be accurately reconstructed from a record of its steep slopes. In this work, tasks producing similar features result in the same surfaces, at some qualitative abstraction.

This paper takes much of its insight from work done in vision and is closely related to Marr's early work [10]. The features are essentially Marr's zero crossing points. A popular technique in object recognition, the snake [14], will be used to locate and characterise these features. One important difference in this research, is that recognising the task is not an end in itself but is a means of initialising another learning process. Thus the recognition process need only be accurate enough to produce a significant speedup in the learning process, averaged over many learning episodes.

In the rest of the paper, Section 2 discusses the case base, giving a brief introduction to reinforcement learning and snakes. Section 3 presents experiments demonstrating feature extraction and how this leads to a substantial reduction of learning time. Following sections deal with limitations, future work and related research.

2 The Case Base

The case base consists of two parts: the cases and the indices. A *case* is a surface obtained using reinforcement learning on a particular task. An *index* is a curve associated with strong features of these surfaces extracted through the use of a snake. The rest of this section discusses how the cases and indices are produced.

2.1 Reinforcement Learning

A simulated robot environment of a main room and an inner room containing the goal is shown as the bold lines in figure 1 (the significance of the dashed lines will be made clear in Section 3). The robot's actions are small steps in any of eight directions. The learning task is to determine the shortest path to the goal from any start location. Here the location, or state, is simply the robot's x and y coordinates. The legal range of the coordinates, defined by the walls of the main room, is the state space. Under the simplifying assumption that the effect of an action is not dependent on previous actions, the task becomes one of learning the best action in any state. The best overall action would be one that takes the robot immediately to the goal. But this is only possible in states close to the goal. The problem is how to evaluate actions in other states, which may result in progress towards the goal without reaching it. This is often termed the credit assignment problem.

This problem has been the focus of much work in the reinforcement learning community. To address it an action is evaluated not by waiting until the final outcome of the task, but rather by correlating it to the evaluation of the best action in the next state [15]. For the robot learning task, at each state an estimate of the distance to the goal could be kept. This estimate would be the result of the best local action, one moving the robot to the new state with the smallest estimate of distance to goal. Early in the learning process only states close to the

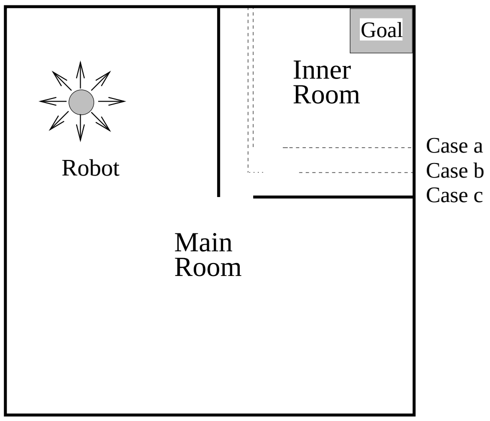

Fig. 1. A Room Within a Room

goal are likely to have accurate estimates of true distance. Each time an action is taken, the estimate of the new state is used to update the estimate of the old state. Eventually this process will propagate back accurate estimates from the goal to all other states.

Rather than directly recording the distance to goal, this paper uses the more usual representation of reinforcement learning, the expected discounted reward for each state $E[\sum_{t=1}^{\infty} \gamma^t r_t]$. The influence of rewards, r_t, are reduced progressively the farther into the future they occur by using a γ less than one. In this work, the only reward is for reaching the goal. So the farther the state is from the goal the smaller the value. This forms a *surface* over the state space. Figure 2a shows such a surface produced by the learning process using the environment of figure 1.

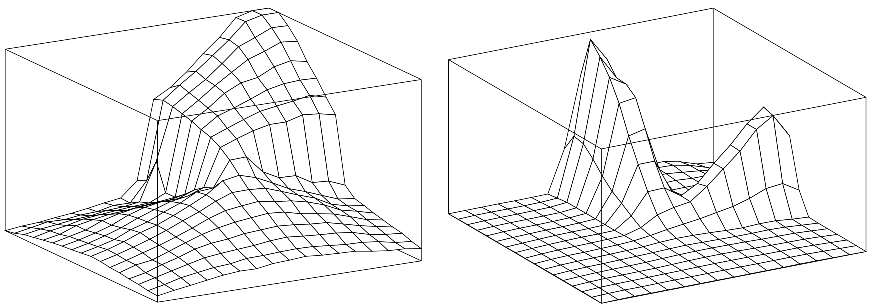

Fig. 2. a) A Surface b) Its Gradient Vector

The two algorithms used most frequently in reinforcement learning are Q-learning [19] and TD(λ) [16]. This research adopts the Q-learning algorithm, the simpler of the two. This algorithm assumes the world is a discrete Markov process thus both states and actions actions are discrete. For each action a in each state s, Q-learning maintains a rolling average of the immediate reward r plus the maximum value of any action a' in the next state s', equation 1. The surface discussed in previous paragraphs and shown in the figures represents this maximum value.

$$Q^{t+1}_{s,a} = (1-\alpha)Q^{t}_{s,a} + \alpha(r + \gamma max_{a'} Q^{t}_{s',a'}) \tag{1}$$

Watkins and Dayan [19] proved that this will converge to the optimal value with certain constraints on the reward and the learning rate α. The optimal solution is to take the action with the greatest value in any state. Thus in this robot problem, a greedy algorithm will take the shortest path to the goal. The extension to continuous spaces can be done using function approximation. The simplest method, and the one used here, is to divide both the state and action dimensions into intervals. Each resulting cell then represents the average of a range of actions taken from a region in the state space. In many applications this method is successful but there exists no general proof of its convergence.

2.2 Feature Extraction

In the plot of figure 2a, one can see two steep slopes separating a raised area from the rest of the plot. Comparing this to the environment shown in figure 1 it is apparent that these correspond to the walls of the inner room. These are the *strong features* discussed in this paper. They exist because the distance to goal is much less when the robot is at the side of the wall inside of the room than when it is outside the room. These features are visually readily apparent to a human, so it seems intuitive to use vision processing techniques to locate them.

The plot of figure 2b represents the magnitude of the gradient vector of this surface. This is the absolute value of the largest gradient in any direction, it forms hills corresponding to the steep slopes of the surface. To locate the features a curve is found that lies along the ridge of the two hills. The method used here is called a snake. In figure 3 the dashed lines represent contour lines for the hills of figure 2b, the dark lines the initial and final position of the snake. To simplify the exposition, we can imagine that the snake consists of a number of individual hill climbers spread out along the line representing the initial position of the snake. But instead of being allowed to climb independently their movement relative to each other is constrained. The main advantages of this approach is that the curve produced is relatively smooth and it tends to prevent individual points getting trapped in local minima. Additional characteristics of the process prevent the points collecting at the hills' apices. When the snake reaches the top of the ridge it will tend to oscillate around an equilibrium position, by limiting the step size the process can be brought into a stationary state. A more detailed mathematical treatment of this approach is given in [6].

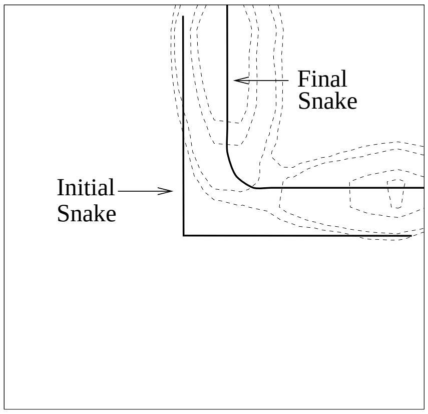

Fig. 3. Fitting the Snake

2.3 Constructing and Using the Case Base

As discussed at the beginning of this section, the case is exactly the surface produced by reinforcement learning. Multiple cases are produced by allowing the process to converge (when changes in the action values are less than some small amount) for different room configurations. Associated with each surface is a curve, the snake. The curve is positioned in the (x, y) plane, defining the bounds of the inner room. Of course, another critical aspect of the room is the position of the doorway. Looking at the gradient plot of figure 2b, the doorway is the valley separating the ends of the two hills. The width and position of the doorway can be determined from the magnitude of the gradient along the snake. The index consists of the (x, y) coordinates of sixty-four points along the snake and the surface gradient at these points.

When a new room configuration is being learnt, at fixed intervals the snake fitting procedure is repeated on the new surface. All snakes for the existing cases are retrieved. Their distances, $D(.,.)$, from the new snake are determined using the weighted sum of two Euclidean norms, equation 2 where the superscript n is the new snake and c the case index. The actual magnitude of the differential was not used in the norm, as it varies considerably during the learning process. But the position of the snake and the position of the valley are largely static. To remove the effect of this variation, the magnitude values are thresholded, $Th()$, by the mean magnitude along the snake producing a vector of ones and zeros. The difference between thresholded magnitude values is much greater than that between location values. So the location norm is multiplied by a scaling factor of ten to normalise the sum.

$$D(s^n, s^c) = 10(\sum_i (x_i^n - x_i^c)^2 + (y_i^n - y_i^c)^2)^{1/2} + (\sum_i (Th(m_i^n) - Th(m_i^c))^2)^{1/2} \quad (2)$$

Providing this match exceeds a threshold, the best match is selected. The case associated with the best match is then copied from the case base and used to initialise the learning system with the surface that closely matches the task in hand. If the match is exact, learning is complete. If not, further refinement is necessary but much less than in a normal learning process.

3 Experiments

The experiments, presented in this section, use the environment discussed in Section 2. The goal remains fixed, but ten different sized inner rooms with different door positions and widths were used. Three of these room configurations, denoted cases a, b and c, are shown in figure 1. The basic Q-learning algorithm was used, with a discrete function approximator as discussed in Section 2.1. The function approximator divides the state space into a 17 by 17 array of discrete cells. The algorithm was run for each of the ten room configurations until there was no appreciable change in the surface. An initial position for the snake was established and then the method discussed in Section 2.2 used to locate the features. The snake's position and and gradient information was extracted to form the index, the surface the case. One of these room configurations (case b) was designated as the test case. The resultant surface and its gradient vector are shown in figure 4.

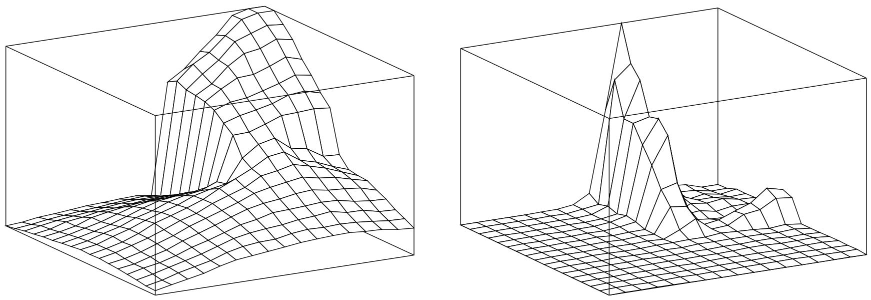

Fig. 4. a) Test Case Surface b) Its Gradient Vector

Each new learning episode started with a surface for the same goal but without an inner room, effectively the default case in the case base. The algorithm was then run in increments of 1000 actions up to 100000. To establish a base-line for comparison the algorithm was first run without using the case base. The algorithm was then rerun but this time, the snake was fitted to each of the resulting surfaces at each increment and the information extracted. If a good match to an existing case was found, the surface was reinitialised with the stored surface.

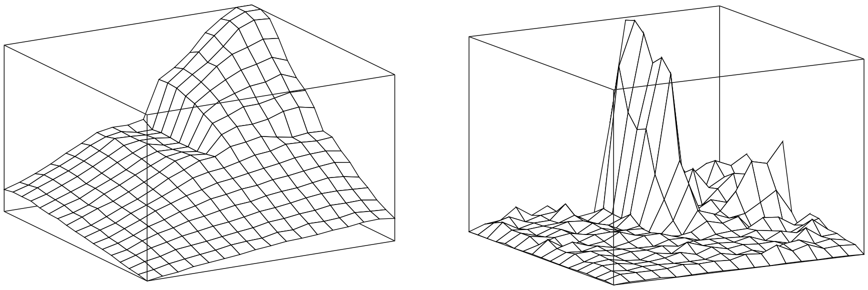

Fig. 5. a) Surface After 2000 Actions b) Its Gradient Vector

In the first experiment the test surface was left out of the case base. The algorithm chose as the best match a slightly smaller room with a door in a similar position, case a in figure 1. In the second experiment the test surface was included in the case base and on repeating the procedure the actual surface was selected. In both experiments, a case was selected after 2000 actions, based on features extracted from the surface shown in figure 5. The gradient vector shows roughly the same form as the test case and this allowed the successful extraction of the features so early in the learning process.

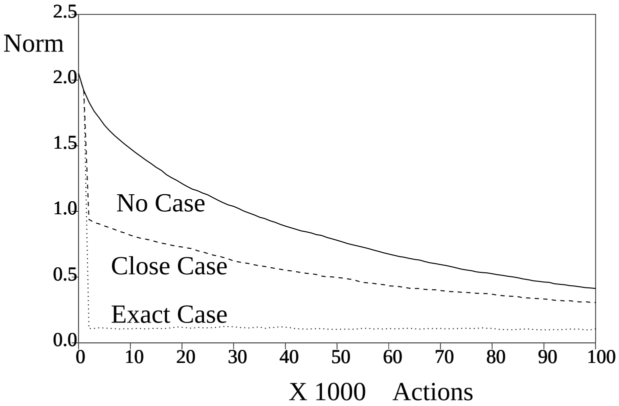

Fig. 6. The Learning Curves

Figure 6 shows learning curves from the experiments. These curves were produced by taking the Euclidean norm of the difference between the surface at each increment and the converged surface for the test case. The topmost curve shows the learning curve when not using the case base. As can be seen at the far end it is slowly converging to an asymptotic value close to zero. It may not actually reach zero as there is some variation in the surface due to the stochastic selection of the actions, but it will oscillate around a very low value.

In both subsequent experiments, after 2000 actions the new snake is considered to be sufficiently close to an index in the case base. The second curve shows the result of using the surface that is close to the desired one. The learning curve drops dramatically as the old surface is introduced. From then on it slowly decreases towards the asymptotic value. The curve is effectively the result of shifting the position at 40000 actions back, so the learning rate is increased by a factor of twenty. When the exact solution is used the learning curve drops to its minimum value. These experiments demonstrate that the features can be detected early in the learning process and are effective in indexing a case base. Even when there is not an exact match, using a surface from a close match results in a considerable increase in learning rate.

4 Limitations and Future Work

There are a number of limitations in the experimental validation of this approach. Firstly, the experiments used only a small case base representing ten room configurations, with a single fixed goal. More extensive experiments are needed with much larger case bases representing a variety of goals. Secondly, the learning curves represent the accuracy of the value surface rather than the average distance to goal. Although some informal testing using the latter measure produced similar improvements, more detailed experiments are required to demonstrate this is generally true.

There is also a need for further experiments in other domains. In the experiments of the previous section, the features are due to the robot's interaction with walls. It is the existence of such objects with well defined boundaries that makes the notion of edge extraction central to much of vision processing. But what of other situations where objects with well defined boundaries are not present, an important situation for this type of learning? In Herve et al's paper [5] on robot arm control, there are also strong features, the singularities in the arm's work space. In Aha and Salzberg's paper [2] learning is particularly successful where the control surface is smooth, but encounters difficulties in the area of strong non-linearities. These examples demonstrate that even in spaces that do not contain well defined objects strong non-linearities occur. This work aims to identify such features, not only because they are easy to recognise but also because they have such an impact on learning. But further experimentation is needed to determine whether or not they can be detected sufficiently early in the learning process to provide a significant increase in learning rate.

There are also a number of limitations in the present implementation of the ideas discussed in this paper. Firstly, a single snake is used and it will only locate features that are connected to the boundaries of the state space. In environments of a more complex collection of rooms and walls, it will be necessary to use multiple snakes. This should present no real technical problem but will somewhat complicate the matching algorithm. Secondly, surfaces that approximately match a new task are used without modification. Once detecting the position of the features, it should be possible to modify the case, as is typical in many case based systems, to better initialise the learning process.

Future work will involve solving these problems and investigating the combining of these ideas with a planning system. Each case would be a plan operator. The learning process would be initialised either from a single case or the composition, determined by the planner, of several cases. This would then be further refined through reinforcement learning, ultimately becoming itself a new case.

5 Related Work

The most closely connected work, to that presented in this paper, is the various forms of instance based or case based learning that have been used in conjunction with reinforcement learning. Their use has been to address a number of issues: the economical representation of the state space, prioritising states for updating and dealing with hidden state. The first issue is addressed by Peng [12] and Tadepalli [17] who use learnt instances combined with linear regression over a set of neighbouring points. Sheppard and Salzberg [13] also use learnt instances but they are carefully selected by a genetic algorithm. The second issue is addressed by Moore and Atkeson [11] who keep a queue of "interesting" instances. These are updated most frequently to improve the learning rate. The third issue is addressed by McCallum [7] who uses trees which expand the state representation to include prior states removing ambiguity due to hidden states. In further work [8] he uses a single representation to address both the hidden state problem and the general problem of representing a large state space by using a case base of state sequences associated with various trajectories. Unlike this other research, in the work presented here the case is not an example of the value function during learning. Instead, it is the result of a complete learning episode, so the method should be complementary to these other approaches.

This work is also related to case based planning [4, 18], through the general connection of reinforcement learning and planning. This connection will become stronger in future work, as discussed in the previous section, where the combination and modification of cases should have much in common with case based planning. Last but not least, is the connection with object recognition in vision research [14, 3]. In many ways model bases are similar to case bases. In this work, many of the methods if not the final application has come from such work.

6 Conclusions

This paper has shown how the use of a case base of surfaces can significantly improve the speed of reinforcement learning. It introduced the notion of using vision processing techniques to extract features from the evolving surface. The experiments demonstrated that the features are stable and can be extracted early in the learning process.

References

1. Agnar Aamodt and Enric Plaza (1994) *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.* AICom - Artificial Intelligence Communications V. 7 No. 1 pp 39-37
2. Aha, D. W., and Salzberg, S. L. (1993). *Learning to catch: Applying nearest neighbor algorithms to dynamic control tasks.* Proc. Fourth International Workshop on Artificial Intelligence and Statistics. pp 363-368
3. C. H. Chin and C. R. Dyer (1986) *Model-based recognition in Robot Vision.* Computing surveys V. 18 No 1 pp 67-108
4. Kristian J. Hammond (1990) *Case-Based Planning: A Framework for Planning from Experience.* The Journal of Cognitive Science V. 14 no. 3
5. Jean-Yves Herve and Rajeev Sharma And Peter Cucka (1991) *The Geometry of Visual Coordination.* Proc. Ninth National Conf. on Artificial Intelligence pp 732-737
6. Frederic Leymarie and Martin D. Levine. (1993) *Tracking Deformable Objects in the Plane Using an Active Contour Model.* IEEE Trans. Pattern Analysis And Machine Intelligence V. 15 No. 6 pp 617-634
7. R. A. McCallum, (1995). *Instance-based utile distinctions for reinforcement learning.* Proc. Twelfth International Conf. on Machine Learning. pp 387-395
8. R. A. McCallum (1995). *Instance-based state identification for reinforcement learning.* Advances in Neural Information Processing Systems 7. pp 377-384
9. Staphane Mallat and Sifen Zhong (1992). *Characterization of Signals from Multiscale Edges.* IEEE Trans. Pattern Analysis And Machine Intelligence V. 14 No. 7 pp 710-732
10. David Marr (1982) Vision : a computational investigation into the human representation and processing of visual information. W.H. Freeman
11. A. W. Moore and C. G. Atkeson (1993) *Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time.* Machine Learning, V. 13 pp 103-130
12. Jing Peng (1995) *Efficient Memory-Based Dynamic Programming.* Proc. Twelfth International Conf. of Machine Learning pp 438-446
13. John W. Sheppard and Steven L. Salzberg (1996) *A teaching strategy for memory-based control.* To appear in AI Review, special issue on Lazy Learning.
14. P. Suetens and P. Fua and A. Hanson (1992) *Computational strategies for object recognition.* Computing surveys V. 24 No. 1 pp 5-61
15. R.S. Sutton (1988) *Learning to Predict by the Methods of Temporal Differences.* Machine Learning V. 3 pp 9-44
16. R.S. Sutton (1990) *Integrated architectures for learning, planning, and reacting based on approximating dynamic programming.* Proc. Seventh International Conf. on Machine Learning pp 216-224
17. P. Tadepalli and D. Ok (1996) *Scaling up Average Reward Reinforcement Learning by Approximating the Domain Models and the Value Function.* Proc. Thirteenth International Conf. of Machine Learning pp 471-479
18. Manuela M. Veloso and Jaime G. Carbonell (1993) *Derivational Analogy in PRODIGY: Automating Case Acquisition, storage and Utilization.* Machine Learning V. 10 No. 3 pp 249-278
19. Christopher J.C.H. Watkins and Peter Dayan (1992) *Technical Note:Q-Learning* Machine Learning V. 8 No 3-4 pp 279-292

This article was processed using the LaTeX macro package with LLNCS style