Behavioural priors:
Learning to search efficiently in action planning

Aapo Hyvärinen
Dept of Computer Science and HIIT
University of Helsinki, Finland

Abstract

We propose that an important part of planning (model-based reinforcement learning) in a biological agent is to use information on the statistical structure of near-optimal action sequences. Such statistical information can be collected simply by observing the action sequences that the agent has actually performed, assuming that the agent only performs action sequences that are the most optimal ones in a search set. Learning such a prior model for behaviour can be used to direct the search of action sequence to the most relevant parts of the action sequence space, and thus enables more efficient planning. The resulting "behavioural priors" are the behavioural counterpart of perceptual priors that are widely used in Bayesian models of perception.

Introduction

An agent must continuously decide which action to take. Actions move the agent to new states, and the states bring rewards (or punishments, which are considered negative rewards here). The agent is supposed to choose its actions so as to maximize the sum of obtained rewards.

A frequently used framework for learning to act based on rewards is reinforcement learning (RL). A well-known problem with the basic formulation of (model-free) RL is that the reinforcement function is assumed to remain constant as a function of time. This may be quite unrealistic for an agent since the locations of food, water, sexual partners and other reinforcing stimuli do change; in fact, they are often consumed and thus disappear from the environment. In particular, it can be argued that the reinforcements change *more quickly* than other parts of the environment, including the set of possible states and actions and the transition rules from one state to another (Foster and Dayan, 2002; Engel and Mannor, 2001). Basic model-free RL methods need to learn the new policy from scratch if the states where reinforcements are received change.

An alternative approach for action selection an agent is based on planning, or model-based reinforcement learning. Here, it is assumed that the agent has a model of the world and its interaction with the world. In a basic approach, the agent considers a number of different action sequences that it might take in the present state. The agent then predicts the consequences of those actions, and evaluates the reinforcement signals associated with each action sequence considered, finally choosing the action sequence with the highest (expected) reward.

Planning is a classical problem in artificial intelligence, although it is usually formulated by defining a single goal state instead of a reward function. The fundamental problem in planning is that the number of possible action sequences grows exponentially as a function of the number of future time steps considered, and thus planning many action steps in advance is computationally impossible. Modern neuroscience models emphasize dynamic programming, i.e. iterative computation of the future "value" of all states, as proposed in model-based reinforcement learning literature.

However, in the case of a biological agent or "animat", the planning problem might be actually much easier because the agent is often faced with similar action selection problems many times over its lifetime, and thus its action sequences have strong regularities. Thus, the agent could learn from its experience, as well as from its evolutionary history. This is in stark contrast to classical planning problems where the environment may be unique for each planning problem to be solved. For example, some work in RL has proposed methods for improving the performance of an agent by learning chunks of actions, called macroactions (Iba, 1989; McGovern, 2002), options (Sutton et al., 1999), or skills (Thrun and Schwartz, 1995).

Here we propose a simple framework for *learning to plan future actions* more efficiently. The agent observes the sequences that it has previously chosen, and learns a statistical model of the sequences — a *behavioural prior* model. Using statistical models for learning of environmental regularities has been a very succesful approach in perception (Knill and Richards, 1996; Olshausen and Field, 1996; Hyvärinen and Hoyer, 2000), and enables the application of the highly sophisticated machinery of Bayesian inference, which is the main motivation for our work. The effect of learning is then to bias the search towards sequences that have the same statistical regularities as the previously chosen ones. This enables the planning system to concentrate on more meaningful action sequences, thus searching the space of possible action sequences more efficiently, and eventually increasing the length of planned sequences to look further ahead. We suggest that this enables the agent to obtain larger average rewards. The framework includes learning of macroactions as a special case, and it can be incorporated in schemes that use a value function approximated by methods related to dynamic programming.

Basic Setting

At every time point $t = 1, 2, \ldots$, the agent finds itself in a state $s(t)$ which belongs to the discrete state space $S = \{s_1, s_2, \ldots, s_n\}$, and has the choice of a number of actions $a(i), i = 1, \ldots, I$ chosen from the set of actions A. Every action changes the state in a deterministic way. (For simplicity, we don't consider random state changes, but they would not change the basic idea.). Every action gives a reward which depends on the combination of the present state and the action taken therein. The agent attempts to obtain maximum reward.

The agent has a perfect model of the environmental dynamics, i.e. what is the next state and the obtained reward when taking a given action in a give state. In principle, it could then compute the values of all states, or the Q-values [1] of all state-action pairs, but it is assumed that the rewards change too fast for this to be computationally useful. Thus, the agent uses explicit planning to choose actions.

The agent has a planning system that searches the space of action sequences or plans $(a(i_1), a(i_2), \ldots, a(i_n))$ of length n. It is assumed that the agent needs to plan its actions a relatively large number of time steps ahead in order to find meaningful action sequences. This means that n is fixed to a relatively large value so that an exhaustive search of the plan space A^n is not possible due to excessive computational requirements. Long plans are necessary, for example, if rewards are *sparse*, that is, the reward is zero for most states, regardless of action taken. We assume this in the following.

Since exhaustive search is not possible, the agent uses a stochastic search strategy that consists of sampling a number of candidate plans from the space A^n of plans of length n. In the basic case, the agent simply samples plans so that each plan in the space A^n has equal probability. For the sampled plans, the agent evaluates the reward to be obtained by following them. The plan leading to the largest reward is then executed. The agent then finds itself in a new state, and starts a new planning process. See, for example, (Kearns et al., 1999) for a related approach.

Learning to Find Better Plans

Learning statistics of executed plans

We propose here that the planning system can learn from experience to constrain the search to a smaller set of plans that are likely to be better than others. This is based on the assumption that *good plans have statistical regularities* in the sense that any action $a(i)$ is typically followed by certain actions and not others. That is, we can use preceding actions *in a good plan* to predict the following actions. A "good" plan means, loosely, a plan that gives an above-average reward.

Thus, we learn to associate a probability to each plan in the plan space A^n: It is the probability that the plan is "good". How do we learn the probability that a plan is good? Since the agent considers many plans before executing the best among them, we assume that all executed plans are good. Thus, the probabilities are learned by building a statistical model of the executed plans. This learning is possible in an unsupervised way. In Bayesian terminology, these probabilities can be called *behavioural priors.*

Improving planning

The statistical model of executed plans, which is based on experience, is then used in planning of future actions in the following way. All candidate plans are sampled from the space of possible plans *according to the probabilities given by the statistical model.* Thus, the system considers only plans which follow the same statistical regularities as the plans that were previously found best. This speeds up planning because it constrains the search to useful parts of the plan space. This learning is domain-independent and can be applied on any kinds of actions.

If the model is such that it can be naturally extended from n-dimensional to m-dimensional sequences, the computational saving due to learning can be used to extend the time horizon of the planning.

Markov models

A simple and useful concrete model that we can use is Markov models. Markov models are simple to estimate (at least for low model orders) and sampling of new plans is very easy, at least if the action space is discrete. Let us denote by $a(t)$ the action taken at time step t.

The k-order Markov model basically predicts the current action $a(t)$ given preceding actions $a(t-1), a(t-2), \ldots, a(t-k)$. The parameters of the model consists of the probabilities $P(a(t)|a(t-1), \ldots, a(t-k))$. Typically, Markov models are first-order, i.e. $a(t)$ is predicted simply by its predecessor $a(t-1)$.

The probabilities can be simply learned by just counting how often an action is followed by another action (in the first-order case) or a sequence of k preceding actions (in the general case). The Markov model also needs the initial probabilities. For a k-order model, we need a probability distribution of the k initial actions taken. Again, such probabilities can be learned by just counting the occurrences of different initial action sequences.

Sampling (generation) of data from the model is straighforward: first, pick up a sequence of k initial actions from the distribution of initial sequences, and set $a(1), \ldots, a(k)$ equal to the obtained sequence. Set $t = k + 1$. Recursively, pick up a new action from the conditional distribution $P(a(t)|a(t-1), \ldots, a(t-k))$, increase t by one, and repeat until the action sequence is as long as desired.[2]

Macro-actions (Iba, 1989; McGovern, 2002) can be easily incorporated in the framework. They correspond

[1]That is, expected discounted future reinforcement when performing a given action in a given state and following the optimal policy thereafter (Sutton and Barto, 1998)

[2]Alternatively, the k first actions could be sampled by using lower-order Markov models, so that first $a(1)$ is picked randomly form the distribution of initial sequences, then $a(2)$ is sampled using a 1st-order Markov model, and so on.

to a statistical model of plans that is based on recoding certain typical sequences as new actions by including them in the action space A. After such recoding, we can build a Markov model for the new, extended set of actions.

Combination with Dynamic Programming

Although in the preceding section we proposed learning behavioural priors in a basic framework where planning is done by sampling from a distribution over action sequences, the same idea carries over to the setting where more sophisticated planning methods are used. In particular, methods based on dynamic programming and value iteration can be easily combined with behavioural priors.

One of the key ideas in dynamic programming, and many RL methods, is that the agent only needs to consider actions one step ahead, since the value function contains all relevant information about the effect of later actions. However, this is only true if the value function is known exactly, which is not true during learning or if the reward structure changes — which was one of our motivations in the first place. As soon as we assume that the values assigned by the agent to different states are not necessarily correct, it is useful to combine planning with RL methods (Daw et al., 2005), and behavioural priors become useful.

In fact, the situation where the value functions are only known approximatively is formally closely related to the present case. The computed value function can be used as reinforcement in the framework described above (where only the reinforcement in the final state of the sequence is obtained). If the agent plans how to get to states having the maximum value, it will move towards the goal, approximately. Learning behavioural priors can be performed in the same way as above, and they will give meaningful sequences to evaluate.

There are many different reasons why the computed values could be far from the true values of the state (for the optimal policy). One case is where some reinforcement learning has been performed, but the value iteration has been carried out incompletely due to computational restrictions. Another possibility is that the reinforcements may have changed without the agent knowing it. Also, if the states are not discrete but points in a multidimensional real space, some function approximation scheme needs to be used to approximate the value function, and errors are likely to occur in the approximation. In all these case, the computed value function gives unreliable information on which action is the best to take. Combining the information they contain with the information given by behavioural priors would be very useful.

In the case of approximately known value function, using behavioural priors has a very similar function to perceptual priors in Bayesian theories in perception: since the sensory input is noisy and incomplete (corresponding to inexact value function), prior information can help infer the real values.

Deterministic planner	32.1
Random stochastic planner	53.7
Bayesian 1st-order planner	84.1
Bayesian 2nd-order planner	84.4

Table 1: Percentage of cases where each planner was the best in the first simulation. There were many ties so the percentages don't add to 100%.

Simulations

Planning by simple forward search

We made simulations in a very basic gridworld of 20×20 states. The state of the agent consists of x and y coordinates of integer values, and the actions are "left", "right", "up" and "down". Rewards are dependent on the states only: the reward values for each state are obtained by taking a random variable uniformly distributed in $[0, 1]$ and raising it to the 10th power. This gives very sparse rewards which are practically zero for most states. Reward was only given at the state where the action sequence terminated.

The simulations can be thought in terms of foraging where the agent moves in a 2D environment, and then implicitly performs an "eat" action at the end of every sequence. The agent tries to eat as much as possible.

First, we used deterministic full search in the action space. The depth of planning was fixed to 4 time steps (actions). Starting from a random initial point, all possible $4^4 = 256$ plans were considered and the best was chosen for execution. This was repeated 10,000 times, each time from a random initial position (state).

Both a first-order and second-order Markov models were learned from the sequences chosen by the deterministic planner. These were used as behavioural priors. Random planning based on sampling from the priors was then performed. Here, the action sequence length was fixed to 8, which gave a large space of 65536 action sequences to choose from. The sampler only considered a small number of choices. The number of considered candidate plans was fixed to 256 which was equal to the deterministic planning case to allow a fair comparison. The same 10,000 initial states as with the deterministic planner were used.

As another baseline comparison we considered the case where the planner takes 256 completely random sequences (uniform initial probabilities and transition probabilities from action to another) of length 8 and chooses the best one. This means planning parameters were similar to those one used with behavioural priors, but the statistical structure of typical action sequences was not used.

We computed the percentage of cases where each of the planners performed the best in terms of obtained reward. The results are shown in Table 1.

We see that the best performance was clearly obtained by the Bayesian planners. Going to a 2nd order Markov model did not really improve the performance.

What is the structure learned by the 1st-order Markov model? The transition matrix $P(a(t)|a(t-1))$, when the actions are in the order ("left", "right", "up", "down") has the following form:

$$\begin{bmatrix} 0.62 & 0 & 0.19 & 0.19 \\ 0 & 0.60 & 0.20 & 0.20 \\ 0.19 & 0.22 & 0.59 & 0 \\ 0.20 & 0.20 & 0 & 0.60 \end{bmatrix}$$

This shows two kinds of structure. First, a very simple and intuitive structure: after going up, one should not go down and vice versa, after going right one should not go left and vice versa. Second, there is a strong tendency to *repeat* the same action in a sequence. This is not intuitively so obvious, but it is due to the fact that by repeating the movement in the same direction, the agent can go farther. Otherwise, it could take sequences such as "left","up", "right", "up" which does not really get the agent any further than two "up" actions, but consumes two more actions.

Planning combined with dynamic programming

In the second simulation, we combined simple search with dynamic programming techniques. The motivation was that the dynamic programming was incomplete and provided only approximations of the true values as discussed above, and thus it was useful to plan many steps ahead (Daw et al., 2005).

The setting was the same kind of gridworld as in the preceding simulation. Now, there was one goal state (in the middle, which bounced the agent to a new random initial position) and the agent performed 10,000 steps of Q-learning. The Q-values were initialized as small random values which provided a limited exploration mechanism in the initial stage of the learning, but the agent always chose the action greedily with no randomness. The step size was 0.5 and the the discount factor 0.9.

The obtained approximation of the optimal value function is given in Fig. 1. Clearly, the learning did not find a very good approximation of the optimal value function due to a limited number of steps, as well as the very primitive learning method that included no proper exploration or such sophisticated mechanims as eligibility traces.

Now, we consider an agent which is only given this rough approximation of the optimal value function. (The value of the goal state is further defined to be infinite). Initially, the agent is place in a random position on the grid. The agent plans a few steps ahead (just as in the previous section) to reach a state of maximum value function, and then executes that plan. This is repeated a maximum of ten times, starting from the state where the agent ended up due to execution of the previous plan, or until the agent reaches the goal state. After ten plans, or after reaching the goal state, a new random initial state is chosen, because often the agent gets stuck and does not move anywhere. The performance is assessed by counting how many times the agent reached the goal.

We used the same four planners as in the preceding section for this planning problem. First, we used the deterministic one and learned the behavioural priors from the sequences chosen for action by that planner.[3]

Deterministic planner	2,422
Random stochastic planner	3,952
Bayesian 1st-order planner	4,961
Bayesian 2nd-order planner	4,733

Table 2: Number of times the agent reached the goal for different planning methods in the second simulation.

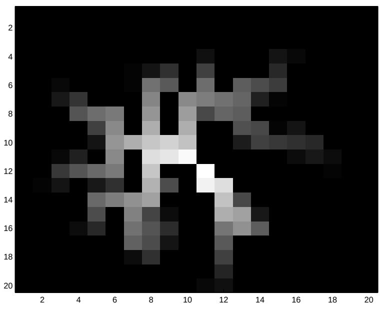

Figure 1: The approximation of the optimal value function, given in greyscale values, used in the second simulation. The approximation is rather bad, which is why it is useful to plan several steps ahead.

The results are given in Table 2. We see that again, the best results were obtained using behavioural priors, modelled as a first-order Markov chain. The learned priors (not shown) were essentially the same as in the preceding section. Interestingly, the 2nd-order Markov model had a performance inferior to the 1st-order model.

Discussion

Related work

Learning motor synergies (Tresch et al., 2006) is closely related to our framework. Obviously, the planning activities can happen on the level of motor control as well. An important difference to our work is that the synergies in (Tresch et al., 2006) considered postures in each time point separately, whereas we emphasize the temporal aspect of planning. However, these two aspects are both easily combined in the framework of behavioural priors. If the action space is a n-dimensional real space, the behavioural priors can be defined on sequences of n-dimensional vectors.

In addition to macroactions already discussed above, researchers in RL have proposed another framework for learning temporally extended actions called options (Sutton et al., 1999). Options are not merely fixed sequences of actions but action selection subsystems that are activated based on some initial conditions, and include a stopping criterion. Options can therefore pro-

[3]In these simulations, there were also many cases where the planner was not able to find a better plan than to stay in the same place. Plans which ended in the initial state were excluded from the plans used in learning priors.

duce arbitrarily long action sequences. A probabilistic interpretation of options, which integrates them in our framework, is an important question for future research.

Case-based planning (Hammond, 1990) provides another closely related framework, together with related work in robotics (Haigh and Veloso, 2000) and AI (Perez and Carbonel, 1994). Our work is different from that framework in that the priors are in the behavioural space only and do not depend on the states; furthermore, we provide a probabilistic framework.

Related work using probabilistic (Bayesian) methods for planning include (Dearden et al., 1999; Attias, 2003). In particular, the methods in (Dearden et al., 1999), as well as (Daw et al., 2005) give estimates of the uncertainty of the value function approximation, and these could possibly be used for determining the optimal depth of planning in a manner similar to (Daw et al., 2005).

Another application of priors on actions or behaviour is in observation of action sequences of other agents in a multi-agent environment (Antonini et al., 2006). This is also an essential subtask in imitation learning. Also, perception of agency (Wegner, 2002) or may also be based on a statistical model of typical action sequences.

Assumption of world models

Planning requires a model of the environment. This may be a drawback with respect to some model-free reinforcement learning methods such as Q-learning (Sutton and Barto, 1998). However, one can argue that it is not unreasonable to assume that many animals have such an internal model, and many relevant computational neuroscience models are based on a world model (Dayan and Abbott, 2001). In fact, one of the functions of consciousness might to enable model-based planning in a kind of virtual reality (Revonsuo, 1995).

Conclusion

We have proposed a framework for action planning based on importing the concept of priors from Bayesian theories of perception. The idea is that *action sequences have statistical regularities*, and these regularities can be learned. The regularities can be used to improve planning in a planner that samples possible action sequences from the distribution given by the prior. Thus, the system learns to bias the search towards useful sequences and to plan more efficiently.

References

Antonini, G., Venegas, S., Bierlaire, M., and Thiran, J. (2006). Behavioral priors for detection and tracking of pedestrians in video sequences. *International Journal of Computer Vision*, 69:159–180.

Attias, H. (2003). Planning by probabilistic inference. In *Proc. 9th International Conference on Artificial Intelligence and Statistics*, Key West, Florida.

Daw, N. D., Niv, Y., and Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8:1704–1711.

Dayan, P. and Abbott, L. F. (2001). *Theoretical Neuroscience*. MIT Press.

Dearden, R., Friedman, N., and Andre, D. (1999). Model-based bayesian exploration. In *Proc. 15th Conf. on Uncertainty in Artificial Intelligence (UAI)*.

Engel, Y. and Mannor, S. (2001). Learning embedded maps of markov processes. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 138–145.

Foster, D. and Dayan, P. (2002). Structure in the space of value functions. *Machine Learning*, 49:325–346.

Haigh, K. Z. and Veloso, M. M. (2000). Learning situation-dependent costs: Improving planning from probabilistic robot execution. In *Proc. Second Int. Conf. on Autonomous Agents*, pages 231–238, Minneapolis, MN.

Hammond, K. J. (1990). Case-based planning: A framework for planning from experience. *Cognitive Science*, 14:385–443.

Hyvärinen, A. and Hoyer, P. O. (2000). Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12(7):1705–1720.

Iba, G. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3:285–317.

Kearns, M., Mansour, Y., and Ng, A. (1999). A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence*, pages 1324–1331.

Knill, D. C. and Richards, W., editors (1996). *Perception as Bayesian Inference*. Cambridge University Press.

McGovern, A. (2002). *Autonomous Discovery of Temporal Abstractions from Interaction with an Environment*. PhD thesis, Univ. of Massachusetts.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609.

Perez, M. A. and Carbonel, J. (1994). Control knowledge to improve plan quality. In *Proc. Second Int. Conf. on AI Planning Systems*, pages 323–328, Chicago, IL.

Revonsuo, A. (1995). Consciousness, dreams, and virtual realities. *Philosophical Psychology*, 8:35–58.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211.

Thrun, S. and Schwartz, A. (1995). Finding structure in reinforcement learning. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems (NIPS) 7*, Cambridge, MA. MIT Press.

Tresch, M. C., Cheung, V. C., and d'Avella, A. (2006). Matrix factorization algorithms for the identification of muscle synergies: evaluation on simulated and experimental data sets. *J. of Neurophysiology*, 95(4):2199–212.

Wegner, D. M. (2002). *The Illusion of Conscious Will*. The MIT Press.