Improving Learning for Embodied Agents in Dynamic Environments by State Factorisation

David Jacob, Daniel Polani, Chrystopher L. Nehaniv
Adaptive Systems Research Group, University of Hertfordshire
College Lane, Hatfield, Herts AL10 9AB, UK
D.Jacob, D.Polani, C.L.Nehaniv@herts.ac.uk

Abstract

A new reinforcement learning algorithm designed specifically for robots and embodied systems is described. Conventional reinforcement learning methods intended for learning general tasks suffer from a number of disadvantages in this domain including slow learning speed, an inability to generalise between states, reduced performance in dynamic environments, and a lack of scalability. Factor-Q, the new algorithm, uses factorised state and action, coupled with multiple structured rewards, to address these issues. Initial experimental results demonstrate that Factor-Q is able to learn as efficiently in dynamic as in static environments, unlike conventional methods. Further, in the specimen task, obstacle avoidance is improved by over two orders of magnitude compared with standard Q-learning.

1. Introduction

Reinforcement Learning (hereafter RL), in its various forms, has long proven to be useful in many areas of Machine Learning, in particular, robot control of various kinds (Maes and Brooks, 1990, Morimoto and Doya, 1998, Smart and Kaelbling, 2002). It works as follows: having decided on a task for the agent to learn, we reward it after every action by sending it a signal (a real value) which tells it how well it is performing. If we correctly design the structure of the reward function, the agent will perform the desired task as a by-product of attempting to maximise its reinforcement reward; the better the task performance, the greater the reward. However, RL's versatility (an agent can be trained by RL to any task which can be represented as a Markovian decision process) is obtained at a significant cost: classical RL methods require a very large number of training examples to arrive at a good policy, which presents practical difficulties for many real-world applications.

It has been shown that learning time increases as a (low-order) polynomial with the size of the state-space (Koenig and Simmons, 1996), which is in turn exponential in the number of state-variables. One basic approach to improving performance is therefore to attempt to reduce the dimensionality of this space. Clearly, in the general case, this is defined by the structure of the agent and environment. But a concrete embodied agent defines in a natural way a structuring of the state-space, in particular the existence of a causal relation between sensor observation, action and reinforcement. Classical RL, being designed for the general case, does not and cannot assume anything of this kind. The identification and exploitation of this causal relation, then, is the basis of the current work.

Generality of application, as we have seen, is one of the chief strengths of the RL paradigm. However it is also the cause of perhaps its greatest weakness, an inability to generalise. This is because a truly general learning process cannot make any prior assumptions about the effects of the same nominal action taken in different states. In other words, if in half the states in the world, action 1 moves the agent north and action 2 rotates it clockwise 90°, and in the remaining states action 2 moves it north and 1 rotates it, it makes no difference to a classical RL algorithm: it will learn a task just as quickly as if all the states behaved the same. Clearly, on encountering a new state, the algorithm cannot predict whether it will behave like the first case, like the second, or indeed neither. Generalisation, on the other hand, requires exactly this predictive ability, based on some concept of similarity between different global states.

It is therefore clear that, so far as robots and embodied real-world agents are concerned, generality *per se* is not really what we want. More useful would be a learning algorithm which could recognise relevant similarities between global states and act accordingly. This would for example enable our robot, visiting a state for the first time, to avoid actions which have yielded reduced reward when taken previously in a 'similar' state. It would also have a beneficial effect on learning in dynamic environments, where local observations may not always be the same for a given location; this is a particular area of weakness for conventional RL algorithms. The question then is how to link perception, action and reward to achieve this outcome.

The central thrust of our approach derives from the regular, orderly nature of physical space, which induces constraints on the behaviour of natural objects. The generalisation we make use of in our work arises only because we are considering learning in the context of robots and other physically embodied agents acting in the real world.

The remainder of this paper introduces a new learning algorithm, Factor-Q, which is designed specifically for this case. Conventional RL is task-based, which is to say that the agent starts each task from scratch, knowing nothing about the world and the effects of its own actions. By contrast, Factor-Q makes the quite different assumption, applicable in many real-world scenarios, that given a set of local observations by an agent, some actions are inappropriate whatever the task: this knowledge once learned is bound to the agent and can be applied from the outset in unseen states and new tasks. In the example experiment reported in this paper, we use this idea to provide obstacle-avoidance in a simple navigation task. This has a dramatic effect on learning outcomes, particularly in dynamic environments with non-stationary obstacles, where obstacle-avoidance is improved by at least two orders of magnitude over the Q-learner's performance.

2. Related Work

2.1 Reducing the size of the state-space

Recognising that the sheer size of the state space in classical RL formulations causes learning to be slow, many methods have been proposed for reducing the size of this state-space. This has often been done by exploiting task structure: for example, McCallum's *utile distinctions* (McCallum, 1993) distinguishes between states only on the basis of their utility in the context of the current task, and to that extent generalises between spatially-distinct states with the same utility. The process used however is computationally expensive and does not make use of any intrinsic properties of the problem under consideration. It starts with no distinction between states and its early learning is therefore entirely random. As a consequence it is too general in application to be particularly suited to embodied systems; further, being explicitly task-based, it cannot help us when the task is changed.

2.2 Multiple sources of reinforcement reward

The principle of combining multiple sources of reinforcement reward is treated in (Shelton, 2000), but this differs from the current work in that the sources are themselves considered to be agents, competing to influence the outcomes of an overall policy. Shelton's work therefore fits better within the established framework of multi-agent RL, for example (Hu and Wellman, 1998).

2.3 Reward shaping through multiple rewards

Multiple sources of reward are also used in the form of subsidiary rewards to bias system behaviour, so-called 'reward shaping' (Ng et al., 1999). However, the authors here are aiming more towards the introduction of heuristic reward to provide dense reward functions, which although difficult to construct may give performance advantages (Smart and Kaelbling, 2002). Although superficially similar to some aspects of the current work, there is no direct intervention in the action selection process of the learning algorithm, and thus no a priori generalisation between unvisited global states.

2.4 Hierarchical task decomposition

Hierarchical RL is another area which has attracted much research. The term "Modular Reinforcement Learning" is often used in this context: a large overall task is decomposed into smaller tasks which can be individually learned, for example (Dietterich, 2000), the module-based RL of (Kalmár et al., 1998), and the options framework introduced by (Sutton et al., 1999). This approach can give benefits in speed of learning and the availability of training examples for sub-tasks which occur multiple times. Unfortunately the decomposition is hard to achieve autonomously, although some success has been attained by state occupation frequency analysis (McGovern and Barto, 2001). In general, however, it appears that domain or world knowledge may be required: the Hierarchy of Autonomous Machines (Parr and Russell, 1997) is an example. Also notable are sequential decompositions (Morimoto and Doya, 1998), where intermediate sub-goals assist in the performance of larger tasks by effectively limiting divergence from a desired trajectory.

Factor-Q, the approach presented here, differs from most of the above in that it is explicitly designed for embodied agents acting in the real world (or models thereof). It attacks the problem of the combinatorial explosion of state space by maintaining, throughout the learning process, a separation between learning a particular task, and learning basic competences which facilitate any task. Whilst in classical reinforcement learning all the state information is 'condensed' to an atomic index variable, Factor-Q maintains this information as a vector throughout the learning process, enabling partial matching between states; on the assumption that locally-similar states require similar actions, it is able to use these similarities to select appropriate actions in states not previously visited. This ability to generalise arises from natural constraints on a physical system, and is particularly valuable in dynamic environments.

3. Reinforcement Learning

Before moving to a detailed description of the Factor-Q algorithm, we recapitulate conventional (tabular) reinforcement learning to motivate the ensuing discussion[1]. Note that for the purposes of the current model, we make the standard assumption that the agent has full knowledge of its global state.

3.1 Conventional Reinforcement Learning

As an agent explores its world, it maintains for each state a table of reward for each possible action in that state.

1. Agent in state s selects an action a according to some policy π. An example of such a policy is the ϵ-greedy policy, which selects the greedy (highest valued) action with probability $(1-\epsilon)$ and an exploratory (random) action with probability ϵ (where $0 < \epsilon < 1$).

2. Agent makes transition to state s' and receives immediate reward r.

3. Agent, now in state s', can see the reward table in that state, and also knows r. This information can be combined and incorporated in the existing entry in the reward table for state s under action a. The rule governing this combination and incorporation is known as the update rule; it constitutes the means by which the agent is able to use past experience as a guide to future action, since the agent is able to consult the table for its current state to help it decide what to do next.

In this way the agent builds a statistical, probabilistic model of reward for each action in each state. There exist convergence proofs for the established RL methods (for Q-learning itself for example see (Watkins and Dayan, 1992)): under certain technical conditions, if we continue the process for long enough we will get arbitrarily close to the optimum policy, which is obtained from the table by selecting the highest-valued action in each state.

An important point to note about conventional RL is its abstract, dimensionless quality: s and a are index values, and r is a single scalar real. While this allows for the maximum generality in the learning process, it is inevitable that dimensionally structured inputs such as the sensors on a mobile robot lose potentially valuable information when they are represented this way. Likewise, a single scalar reward is not the most useful indicator of outcomes in an embodiment situation since it cannot indicate any 'reason' for the reward assigned.

[1]A comprehensive treatment of RL techniques including Q-learning is provided by (Sutton and Barto, 1998); here we describe the basic principle by which discrete RL methods operate.

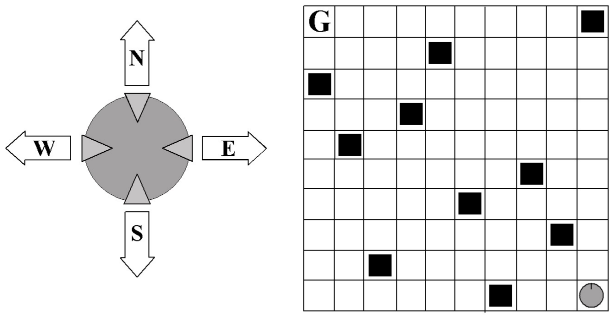

Figure 1: *shows, left, the gridworld agent, and right, the gridworld with obstacles and goal. Agent is shown in its starting position*

Conventional RL builds a retrospective statistical model of the relationship between state index, action index and reward, but it is a fundamental assumption in Factor-Q that in real-world interactions the local effect of the same action will often be similar whatever the global state. In this scenario, for example, if the agent detects an obstacle and moves towards it, it will collide with it; this will happen regardless of where in the world the agent happens to be. This natural causal relationship between local action and local observation is characteristic of real-world systems.

With the addition of locally-generated reward, we may be able to learn this local relationship separately from, but at the same time as, the global task. This task is in turn made easier to learn by the corresponding reduction in the dimensionality of the underlying process. Factor-Q provides a framework within which the learned local and task-based reward functions may be combined to produce actions which take account of both local conditions (which in a dynamic environment are transient) and global goals.

4. Purpose and description of experiment

The experiment to be described demonstrates the advantages to be gained from maintaining this dimensional information throughout the learning process. An agent (see figure 1) is placed in a 4-connected gridworld, which in this experiment can be considered actually to represent a physical agent on a 10×10 chess-board with a perimeter wall, rather than an abstraction of a general process. The agent has four actuators each of which can move it one square in one of the four possible directions north, west, south, east (it moves by translation only and does not rotate). The actions are shown as arrows in the diagram. Exactly one of the actuators fires at each action step. The grid has randomly-placed obstacles (shown as black squares) in it each of which occupies a single square: in some experiments these obstacles are static, in others they move at random. The task is for the agent to move from the starting position in one corner of the world to the goal (G in the figure) in the corner diagonally opposite, without hitting the walls or the obstacles. To achieve this the agent is equipped with four proximity sensors which indicate the presence or absence of an obstruction in the four adjoining squares. Bump sensors on each face detect when contact has occurred. The position of these pairs of sensors is indicated by the triangles in the figure.

4.1 Conventional representation

In the conventional formulation of this task, the reward function would typically have three components:

- A reward for reaching the goal (typically $+1$) added to
- A reward for each step taken (typically -1) – this encourages the agent to complete the task in the minimum possible time, since the longer the task continues, the greater the negative reward
- A reward for collisions (typically -1) – this in addition to the step reward

The state-space for this representation would have the dimensions 100 (cells in the grid) times 16 (to represent every possible combination of the four binary proximity sensors), and for each of these 1600 states there would be 4 action values, giving a total size for the reward table of 6400 entries.

However, if the task changes so that another state becomes the goal, only the first reward function component is affected, as it is now triggered in a different state: the second and third components are unaltered. These latter are therefore not so much related to a particular task as to tasks in general. Thus we can factor them out of the task and apply them separately: this is the idea underlying Factor-Q.

4.2 Factor-Q representation

In this formulation, the task is defined only in terms of the goal and step rewards. The avoidance of obstructions (walls, obstacles) is achieved using a separate reinforcement mechanism which is local to the agent. This reflects the relationship between the agent's sensors, its actions, and reward signals generated internally in response to collisions in the environment.

One effect of this representation is greatly to reduce the size of the state-space. Obviously there is the same number of cells (100) in the grid, but this grid now has only to represent the task reward function; the agent's sensors need not be included here. The cells now represent physical locations in the world on a one-to-one basis: with four possible actions in each state, the grid now contains 400 action-values.

To this number has to be added the number of action-values needed by the agent's obstruction-avoidance mechanism. In this deliberately simple example, sensors and actions grouped in pairs which are orthogonal and therefore mutually independent. We can therefore deal with each sensor/actuator pair separately as follows. The state-space for each pair has cardinality 2 (representing the binary sensor input); for each of these states there are 2 action-values. The size of the reward table associated with each pair is therefore 4. (We use the term 'local' for these tables and rewards, and 'task' to denote the main table and its associated action-values.) To assess the overall reward for each action, the agent combines the projected reward from local and task tables as will be described in section 5.2 below; it uses this information to select its next action.

The number of action-values to represent the whole task using Factor-Q is thus 100 cells times 4 actions, plus 4 instances of 2 states times 2 actions, a total of 416. This represents a reduction of 93.5% over the conventional representation. The practical advantages of this more compact representation will be clearly seen in the results of the experiment.

One further way in which Factor-Q differs from the conventional representation is that the agent preserves the dimensionality of the collision reward by representing it as a binary vector (of length 4 in this case). The four actions and the four sensor observations are represented in the same way, as vectors. In the current case, this vector representation is not strictly necessary for the actions, nor for the local reward provided it is kept separate from the task reward signal. However, it must be borne in mind that this is a simple example intended for didactic purposes and proof of concept. More complex situations would require dependencies between sensors, actuators and rewards to be learned[2]. Moreover, in future work we plan to investigate the possibility of multiple independent simultaneous actions, necessitating the use of an action vector; clearly in this case correct reward assignment is made much easier if the various rewards arising from the actions are kept separate rather than aggregated into a whole, discarding valuable dimensional information which must then be imperfectly

[2]In the worst case, where no factorisation is possible, all possible combinations of local observation and action would need to be separately learned. There are, however, strong indications that real-world cases allow for at least partial factorisation, and that these dependencies can be learned using a more modest extension of the system described here. Naturally, for a given embodiment, this would only have to be done once (since it relates to the agent and is independent of any particular task). For this reason, and because of its ability to treat task and local rewards separately, thereby reducing the size of the overall state-action space, Factor-Q would still confer worthwhile benefits.

reconstructed statistically.

5. Details of algorithm

For the reader's convenience and for purposes of comparison we begin with details of the Q-learning algorithm and extend them to Factor-Q. (We consider only the episodic case here: the formalism extends without modification to the infinite-horizon case where the agent tries to maximise the *rate* of receipt of reward.)

5.1 Q-learner

Assume the agent is in state s. It selects its next action a with probability $1 - \epsilon$ according to:

$$a = \text{argmax}_a Q(s,a)$$

where $Q(s,a)$ is the expected future discounted reward which will be obtained from taking action a in state s and following the optimum policy thereafter, and $0 < \epsilon < 1$ is the probability of an exploratory (random) action. If a is not unique, one of the contenders is selected at random. The agent moves to state s' and receives immediate reward r according to the outcome of the action. It now updates $Q(s,a)$ according to the Q-update rule:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') \right] \quad (1)$$

(where $0 < \alpha \leq 1$ is the learning rate and $0 < \gamma < 1$ is the future reward discount rate) and repeats until the goal is reached.

Note particularly that nothing is known a priori about any unvisited state, and actions in that state are therefore random.

5.2 Factor-Q

Assume the agent is in state s and has local observation vector $\vec{o}$. The local expected reward for action i under observation $\vec{o}$ is $\rho_i(\vec{o})$: we write ρ_i by abuse of notation.

The underlying idea here is to maintain a (small) table of immediate *local* reward for each action under different local observations; sufficient observations should be included to make these rewards as far as possible deterministic. Action selection is performed using these reward tables in conjunction with a conventional Q-table (which now deals only with task rewards) to predict the overall outcome of an action.

(In the current work, actions are orthogonal, each action has an associated observation, and index i is tied to action a. However, the fundamental assumption of Factor-Q is that *in real circumstances* in the best case there will be a one-to-one relationship between action and observation, and in many other cases few-to-few. There are strong indications that it will be possible to learn these more complex relationships autonomously, and this will be explored in future work.)

The vector $\vec{C}(s)$ of combined expected future discounted reward on which the agent will base its action selection is given by

$$\vec{C}(s) = \vec{Q}(s) + G\vec{\rho} \quad (2)$$

where $\vec{\rho}$ is the vector of projected immediate local reward and G is a normalisation factor. To obtain G we first find the greedy action $\hat{a}$ (for which the task reward is highest):

$$\hat{a} = \text{argmax}_a Q(s,a)$$

and the mean $\bar{Q}$ of the expected future discounted rewards of all the other actions:

$$\bar{Q} = \frac{1}{N-1} \sum_{a \neq \hat{a}} Q(s,a),$$

N being the number of actions from which to select. Similarly, we find the action $\check{a}$ for which the local reward is the *minimum*:

$$\check{a} = \text{argmin}_i \rho_i$$

and the mean $\bar{P}$ of the local rewards for all other actions:

$$\bar{P} = \frac{1}{N-1} \sum_{a \neq \check{a}} \rho_a$$

Finally we can combine these elements to give G:

$$G = \frac{Q(s,\hat{a}) - \bar{Q}}{\bar{P} - \rho_{\check{a}}} \times (1+\eta) \quad (3)$$

G is needed because the relationship between the task and local action values is of its nature arbitrary (all we require of the local reward is that worse outcomes are punished harder) but some means must be found to combine them for action selection. G achieves this by scaling the size of variation in the local reward to the size of variation in task reward for the current state.

It is necessary for the local reward term slightly to dominate the reward sum, since we do not want actions to be selected which will result in collisions – these will not further task fulfilment. A value of $\eta = 0.1$ has been found by experiment to work well, but the algorithm does not appear sensitive to the particular value chosen, so long as it is greater than zero. For the same reason, G is subject to a small minimum value so that it dominates 'empty', previously unvisited, states.

After action we update the task reward table, but this time using the same combined rewards used for action selection:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \\ \alpha \left[r + \gamma \max_{a'} \Big(Q(s',a') + G\rho_{a'} \Big) \right] \quad (4)$$

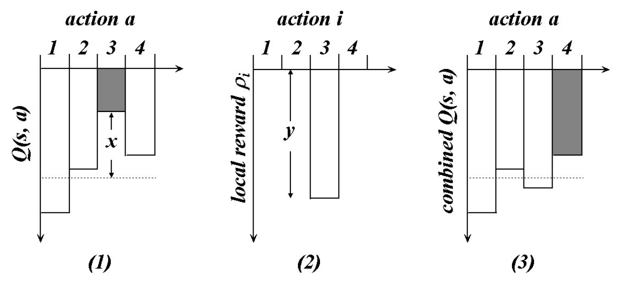

Figure 2: *demonstrates the calculation of G and its application in combined reward action selection.* $x = Q(s, \hat{a}) - \bar{Q}$ *and* $y = \bar{P} - \rho_{\hat{a}}$ *in equation (3). See text for details.*

The local reward tables too are updated, according to

$$\vec{\rho} \leftarrow (1 - \alpha_L)\vec{\rho} + \alpha_L \vec{r_L} \quad (5)$$

where α_L is the local learning rate and $\vec{r_L}$ is the immediate local reward vector generated in response to action a.

Here we note that Factor-Q is able to base its action selection on both the global (task-based) and the learned local reward functions; the advantages that this confers are best illustrated by consideration of the results obtained in this experiment.

5.2.1 Update rule

The update rule of (4) is equivalent to the normal Q-update rule of (1) here because in both cases the maximum expected future discounted reward of the greedy action (which for Factor-Q is calculated by (2)) is used for the update. It is important not to use the raw value from the task reward table since this will be an over-estimate of the future discounted reward from that state (under the current conditions); this promise of high reward will attract the agent and may cause endless loops, particularly in states adjacent to permanent obstacles.

5.3 How it works

Figure 2 shows a possible action-selection scenario from the gridworld experiment. In Q-learning the greedy action corresponds to the highest value for $Q(s, a)$ in the global reward table. This is shown in grey in fig. 2.1 . Suppose that for the state s shown in 2.1 the expected rewards for the observation vector $\vec{o}$ corresponding to s is as shown in 2.2 . The mean $Q(s, a)$ for actions other than the greedy action is shown by the dotted line in 2.1 . Thus the numerator in equation (3) is x in 2.1 and the denominator is y in 2.2 . Applying these values in equation (2) results in the situation shown in 2.3, where the new greedy action is shown in grey. The value of this action as shown in 2.3 corresponds to the term $\max_{a'}(Q(s', a') + G\rho_{a'})$ in the update equation (equation (4)). Thus action 3 is suppressed and action 4 is substituted.

6. Experiments

The experimental set-up is described in section 4. We assume ϵ-greedy action selection in all cases, with $\epsilon = 0.01$. All reward tables are pre-initialised to 0, task learning rate $\alpha = 0.2$, local learning rate $\alpha_L = 1/\kappa$ (where κ is the number of times this action has been taken, in other wordsthe expected local reward is the mean of all rewards so far received) and $\gamma = 0.9$. (In these experiments we always consider a fixed learning rate for the task, which may be advantageous in some dynamic environments. A control experiment in the static world using a task learning rate which decays such that the expected future reward is always the mean of all the rewards so far received produced statistically identical results to those reported.) An experiment comprised ten runs of 200 episodes each. One series of experiments was performed using standard Q-learning; the reward table was re-initialised to 0 before each run of 200 episodes. In the other experiments, Factor-Q was used; in this case only the task reward table was re-initialised before each run, the local tables being initialised only at the start of each experiment[3]. One experiment in each series was performed in each of the following worlds:

6.1 Static world

Ten stationary obstacles, each occupying one square, were placed on the grid in the positions shown in figure 1.

6.2 Dynamic world: changes every ten moves

The ten obstacles were reset to new random positions after every ten actions by the agent. The only constraint on the position of the obstacles was that they could not occupy the start cell, the goal cell or the cell occupied by the agent. In the results this world is referred to as 'semi-dynamic'.

6.3 Dynamic world: changes every move

The ten obstacles were reset to new random positions after every action by the agent.

7. Results

The results of the experiments are summarised in figure 3. Each of the three graphs shows the mean of

[3]Because the local reward is learned so quickly in this simple example, the agent could very quickly have learned it anew for each run of 200 episodes. However it is fundamental to the operation of Factor-Q that the agent can learn local rewards and is able to retain that knowledge and apply it to new tasks.

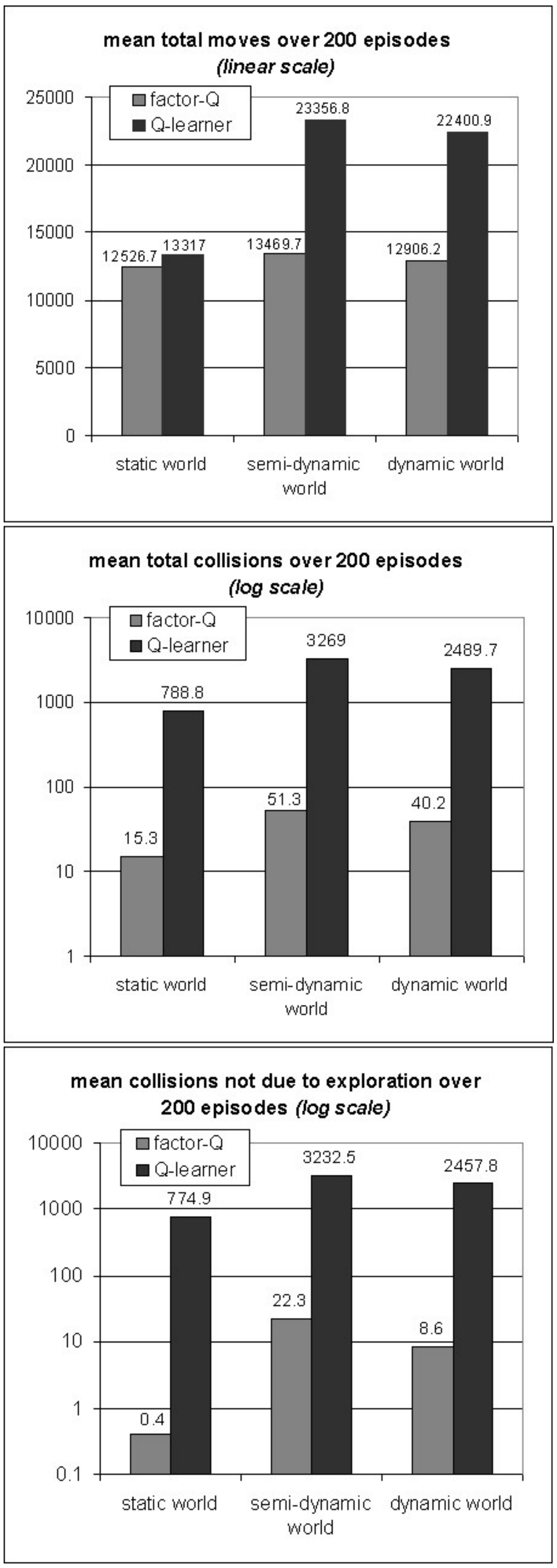

Figure 3: *summary of experimental results*

the results obtained over ten runs each comprising 200 episodes. The following points are worth noting:

- On every performance measure, Factor-Q outperformed Q-learning.
- Learning rates in static environments are not significantly different, as one would expect: the underlying task learning process in both systems is the same. However, collision avoidance in the static world was dramatically better in Factor-Q
- Factor-Q learns the dynamic environments nearly as quickly as the static one, whereas Q-learning's learning rate is badly affected by the moving obstacles
- Factor-Q improves on Q-learning's collision performance by over two orders of magnitude when collisions due to exploratory moves, when a non-greedy action was selected,are discounted.

8. Discussion

Factor-Q's performance in the dynamic environments is the most interesting aspect of these results. Taking into account the increases in mean path length which arise because of the non-static obstacles, the performance was not significantly degraded over the static case. The collision performance also was better than it appears from the results, since nearly all collisions occurred when the agent was completely surrounded by obstacles and then had no choice but to collide with them.

Two factors contribute to this improved performance: the ability of a vectorised representation to attribute reinforcement reward directly to the actions and observations which give rise to it, and the predictability of the local effect of actions irrespective of global state which we can assume only because we are dealing with embodied, real-world agents. This predictability is exploited by the new action-selection mechanism which is able to avoid actions which it predicts to be bad, unlike Q-learning which can only avoid actions which it has tried before and *found* to be bad. This is clearly highly advantageous in dynamic environments, where local observations (and therefore optimal actions) may vary for the same global state (position) and need to be taken into account the first time they are encountered.

Clearly the example given is a simple system, where orthogonal actions and their effects are able to be completely separated from one another rendering the local learning process very quick[4]. However, even in a complex real system, there will as a rule be certain natural associations between actions and consequences which

[4]In these experiments, sensor and actuator noise were not modelled. In the presence of these, the speed of local learning might need to be reduced somewhat to reduce the disproportionate effects of noise early in the learning process; nonetheless, the general point still holds.

can be exploited to reduce the overall burden of learning. Considering that the size of a global state-space is exponential in the number of state variables, and linear in the cardinality of each, any factorisation which can be done will make the learning problem much more tractable. This will be even more the case if simultaneous actions are taken, where there is no temporal separation between actions and their associated rewards, making the reward assignment problem much more acute. All of which, coupled with the ability of the new algorithm to act on predicted reward, holds the promise of improved performance in more realistic scenarios than explored so far in the current work.

9. Conclusions and further work

A new algorithm specifically for reinforcement learning in robots and embodied agents, Factor-Q, was introduced. The algorithm uses vectors for state, action and reinforcement to preserve dimensional information through the learning process and to maintain separation of reinforcement reward due to the task from reward related to the agent's local interaction with its environment (in this case represented by collisions with obstacles). In addition, this vector representation is more scalable than conventional methods because of the factorisation of the state-space which it induces. Experimental results from an example simulated scenario show learning speed remains largely unaffected by a dynamic environment, in contrast to conventional Q-learning, whose learning rate falls to 57% of its static-world rate in the worst case encountered here. Much more serious than this degradation in overall learning rate however is the inability of the Q-learner to act in accordance with local observations. The agent frequently finds itself in a state which it has never previously visited (because a moving obstacle has induced a change in state index for that particular grid cell) and has to act at random, often resulting in collisions.

The authors are currently developing a new methodology for reinforcement learning in robots and embodied systems based on a modular decomposition of the learning process to reflect the structure of the agent's embodiment. The distributed learning algorithm presented here will be an integral part of this methodology, augmented to accommodate positive as well as negative local rewards, and an ability to learn interdependencies between actions, local observations and rewards where these exist. It will also be necessary to provide an arbitration mechanism, probably based on reward structure, to decide when the task should take priority. For example, in the current experiments, the task might require the agent to strike an obstacle – under the existing framework this would not be possible.

As previously mentioned, Factor-Q may also present possibilities for agents to select and perform more than one action at a time and still be able to make the correct assignment of reward: this will also be explored in future work.

References

Dietterich, T. G. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Artificial Intelligence Research*, 13:227 – 303.

Hu, J. and Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Fifteenth International Conference on Machine Learning*, pages 242 – 250.

Kalmár, Z., Szepesvári, C., and Lorincz, A. (1998). Module-Based Reinforcement Learning: Experiments with a Real Robot. *Machine Learning*, 31:55 – 85.

Koenig, S. and Simmons, R. G. (1996). The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, 22(1-3):227–250.

Maes, P. and Brooks, R. A. (1990). Learning to Coordinate Behaviors. In *National Conference on Artificial Intelligence*, pages 796–802.

McCallum, A. (1993). Overcoming Incomplete Perception with Utile Distinction Memory. In *International Conference on Machine Learning*, pages 190–196.

McGovern, A. and Barto, A. G. (2001). Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *Eighteenth International Conference on Machine Learning*, pages 361–368.

Morimoto, J. and Doya, K. (1998). Reinforcement Learning of dynamic motor sequence: Learning to stand up. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1721 – 1726.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Sixteenth International Conference on Machine Learning*.

Parr, R. and Russell, S. (1997). Reinforcement Learning with Hierarchies of Machines. In *Advances in Neural Information Processing Systems*, volume 10.

Shelton, C. R. (2000). Balancing Multiple Sources of Reward in Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pages 1082–1088.

Smart, W. D. and Kaelbling, L. P. (2002). Effective Reinforcement Learning for Mobile Robots. In *International Conference on Robotics and Automation*.

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

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

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.