Incremental and Mutual Adaptation in Multiagent Systems

Claudia V. Goldman * Jeffrey S. Rosenschein
Insitute of Computer Science
The Hebrew University
Givat Ram, Jerusalem, Israel
ph: 011-972-2-658-5188 fax: 011-972-2-658-5439
email:clag@cs.huji.ac.il, jeff@cs.huji.ac.il
url: http://www.cs.huji.ac.il/∼ *clag*

Technical Report, CS96-15

*Supported by the Eshkol Fellowship, Israeli Ministry of Science

Incremental and Mutual Adaptation in Multiagent Systems

Claudia V. Goldman Supported by the Eshkol Fellowship, Israeli Ministry of Science
Jeffrey S. Rosenschein
Insitute of Computer Science
The Hebrew University
Givat Ram, Jerusalem, Israel
ph: 011-972-2-658-5188 fax: 011-972-2-658-5439
email:clag@cs.huji.ac.il, jeff@cs.huji.ac.il
url: http://www.cs.huji.ac.il/$\sim clag$

Topic Areas: Distributed AI, Learning and Adaptation in multiagent systems.

Abstract

There are two streams of research when combining multiagent systems and learning. One regards multiagent systems in which the agents have learning capabilities and they learn from the environment in which they operate. The feedback the agent receives from the environment includes knowledge about the other agents in the system. The second direction investigates the issue of multiagent learning where the focus is on the interactions among the learning agents. Each agent learns directly from the other agents that operate in the same environment.

In this work, we investigate how agents can learn strategic behavior in a teacher-learner model. The notion of the teacher here should be understood as a "trainer." In certain research on automated learning, a "teacher" is presupposed to be benevolent with his student. In a general multiagent learning system, however, we might have benevolent agents as well as manipulative agents. A manipulative agent, acting as a trainer, will instruct the learner in a way that causes it to act in the future for the trainer's benefit.

Each agent computes a function that expresses the agent's satisfaction level. This function depends on two terms. One represents the learner's gain from the action it has performed, and the state to which the world has moved. The second term represents the degree of approval of the trainer according to the learner's behavior so far. The general model enables an agent to learn from many trainers, considering the weighted degree of approval of its trainers in its environment. Each agent adjusts its behavior to increase its satisfaction level in order to adapt to one another. Our aim is that the agents learn to coordinate their behaviors, i.e., the agents will learn to adapt themselves.

We present preliminary results of experiments we have conducted. We also propose an algorithm for incremental adaptation in a multiagent system, based on the trainer-learner model where the agents behave in order to maximize their satisfaction levels.

1 Introduction

Two different research directions can be distinguished when combining learning and multi-agent systems.

One might be called *learning in a multiagent system*. Following this stream, the agent learns by choosing its actions according to feedback it receives from the environment in which it operates. This environment includes other agents and thus the learning agent is learning in a multiagent system. The behavior of the other agents in the system is interpreted and encoded in a single value. This is the feedback that the learning agent receives. The agent decides upon its next actions in a way such that they will maximize this value. In this case, the environment is influenced by the actions of all the agents operating in it. The learning agent "knows" about the existence of the other agents in the same system indirectly through the reinforcement value it gets from the world.

For example, in [3], a central critic gives the payoff to each agent in the system. Each agent learns about the dynamic structure of its environment in [6]. Reinforcement techniques are widely used in the DAI literature. Agents that learn according to the Q-learning algorithm [11] and also cooperate with other agents by exchanging information (partial solutions, plans of action) can learn more quickly than agents that do not cooperate [10]. In [8], agents learned complementary policies without sharing any information with the other agent. Another example of achieving global coordination without sharing any knowledge can be found in [9]. Experiments were conducted in multiagent scenarios using the Q-learning algorithm in [7]. Another multiagent scenario, in which the agents don't learn from each other appears in [1]. The agent, facing the environment, regards the other agents as a single unit.

The second interesting paradigm is concerned with the issue of a *multiagent learning system*. The focus in these systems is on the interactions among the agents while they learn. Each agent learns directly from each of the others.

Markov games were proposed as a framework for multi-adaptive agents [4], although this framework is not suitable for analyzing the emergence of cooperation in such systems. Learning social rules [5] requires three types of reinforcement, including the reinforcement from other agents that operate in the same world. This work deals with homogeneous societies, i.e., every agent's benefit is the system's benefit.

We are interested in heterogenous agents that learn to coordinate their courses of actions. We have already suggested [2] the model of a teacher and learner, where both agents are involved in a mutually supervised learning process. The agents went through a training period, and then through a generalization period in which they generalize what they have been taught and act accordingly.

In this work, each agent gets feedback directly from the agent with whom it is interacting. In a heterogeneous environment, each agent can grade the others' behaviour separately. The same agent might regard its partners differently. Thus, each agent in the environment can have a particular influence on specific agents. For example, agents might behave cooperatively or not based on their mutual past history. The agent might behave more friendly with an agent that has cooperated with it in the past than with an uncooperative agent. The cooperative agents will receive a higher feedback from the agents they have helped, and lower feedbacks from those that got less help.

Another subject we want to point out in this work is the need to define and look for *fair* adaptive solutions and the difference between them and the cooperative solutions. We will just refer to the fairness issue with regard to the example we present in this paper. Nevertheless, we think that more formal work needs to be done in order to define what it means to be "fairly adapted" for problems that are solved within a multiagent system.

In a previous paper [2] we gave several learning rules and a theoretical proof for the learnability of a cooperative solution for a multiagent system. The model used was based on a teacher-learner model, where the agents went first through a training stage and then generalized what they have learned. In this paper, we will be using the same teacher-learner model, but the agents will be involved in a continous adaptation process. They will also execute their actions, while learning to adapt their behaviors to the other.

Once again, the notion of the teacher in our model differs from the usual meaning of a teacher in the artificial intelligence learning literature, who is presupposed to be benevolent to his student. In our case, a general multiagent system, the teacher should be regarded as a trainer that intends to instruct his student how to behave. The difference lies in enabling manipulative trainers in the environment that might train their learners for the trainers' own benefit.

We also implemented this model in a traffic scenario, where each agent represents a traffic signal controller.

In the following sections, we will formally describe the model and the domain. The agents decide upon their actions based on the feedback received from the other agent in the system. Each agent learns to adapt to the other agent by evaluating the feedback values given by its teacher. We will show several results we have gotten from experiments, and we will also present a new algorithm for incremental mutual learning.

2 A Multiagent Adaptive System

The main research question we are investigating in this work regards how heterogeneous agents could learn to coordinate their behaviors when interacting in the same environment. We propose a trainer-learner model to answer this question. Each agent in the system will play both roles, the trainer and the learner. The trainer tries to train the learner to behave for its (i.e., the trainer's) own benefit, while the learner behaves as a *provoking agent*. Intuitively, the agents will stick to their decisions until they have "irritated" the other agent too much, and then they will change their choice until their own gain from choosing their actions drops below a threshold, and then they will start to provoke again.

Time is discrete. Each agent has goals to achieve, and the agents' goals might interfere; therefore, there is need for coordination in the system. Each agent is able to characterize the state of the world, and therefore it can evaluate how far the current state is from the state in which its goals are achieved. At each period of time, the agents evaluate their last action (or actions), the state in which the world has been moved to, and their trainer's feedback, and then decide upon their next action.

An agent acting as a trainer will grade its learner behavior according to how the learner's actions affect its state. A trainer, being one of the agents in the multiagent system, has

```
t=0
Initialize A_i's state at t=0
Initialize a^0_{A_i} randomly
While (¬ StopCondition) {
   Compute S(A_i, t)
   Compute A_i's state at t+1
   Apply an Adaptive Criterion on S(A_i, t) to
   Decide on a^{t+1}_{A_i}
   t = t+1
}
```

Figure 1: The basic mutual learning algorithm

its own goals to achieve and the other agents' actions might interfere with them. Hence, each agent will send the others (its learners) feedback that represents how happy it is with the others' choice of action. An agent, when playing the role of the learner, computes its satisfaction level based on the state of the world, its goals, and the feedback it got from its trainers. The learner will choose the action that increases its satisfaction level.

In general, the satisfaction level of an agent A_i ($1 \leq i \leq N$, N is the number of agents in the system) at time t, $S(A_i, t)$, is given by:

$$S(A_i, t) = \alpha_{ii} \times OwnGain(A_i, t) + \Sigma_{1 \leq j \leq N, j \neq i} \alpha_{ij} \times Feedback(A_j, t)$$

$$\Sigma_{j=1}^{N} \alpha_{ij} = 1$$

α_{ii} expresses how much A_i weights its own gain, and α_{ij} is how much A_i cares about A_j (i.e., how much A_i obeys A_j). By giving different values to these parameters, we can simulate different types of agents. For example, if A_i is a selfish agent, then α_{ii} will be one while α_{ij} will be zero. On the other hand, if A_i is an altruistic agent, it will only care about what the other agent's A_j tells it to do. Any other combination of weights that sum up to one will represent different levels of cooperation that exist between the agents. The satisfaction level of A_i at time t depends on its own gain after having executed $a^t_{A_i}$ (i.e., how happy it is with its own choice, and therefore with the state the world was moved to as a consequence of the action), and it also depends on the feedback sent by A_j at time t (i.e., what A_j thinks of A_i because of the action it has chosen and the current state of the world). The learner gets from its teacher a value, but the learner does not need to know how this function was computed.

The basic learning algorithm is shown in Figure 1.

3 Experiments

3.1 The Traffic Lights Domain

We consider a single traffic intersection, as shown in Figure 2.

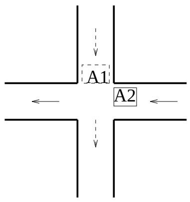

Figure 2: A simple traffic intersection

There are two agents; i.e., two traffic signal controllers. A_1 is responsible for the traffic flow in the vertical direction, while A_2 is in charge of the horizontal road. Time is considered to be discrete. The traffic at each road is represented by a simple queue. Each agent can choose to perform one out of two possible actions, *red* (encoded as 1) and *green* (encoded as 0). At each period of time, a new car might be added to each agent's queue, and a car might leave the queue if the agent chose to execute the green action. The size of each agent's queue at time t, $|Q_t|$, can remain unchanged at time $t+1$ when the action performed was 0 and also a car entered the queue, or when the agent chose 1 and no car was added to its queue. $|Q_t|$ can increase by one when a car was added to the queue and the agent chose 1, and it will decrease by one when the agent performs 0 and no new car entered the queue. A collision occurs at time t, if both agents performed 0 at this time.

The agents' goals are to maintain their queue as short as possible (the optimal state is when $|Q_t| = 0$) and to avoid collisions. These goals conflict but there exists a solution that requires the agent to coordinate. We want the agents to learn to adapt to each other, through continuous interactions.

A cooperative, non-adaptive, solution for both agents to achieve the goals mentioned above is given by the following algorithm (it is written for A_1; A_2 needs to run the same program when the corresponding indexes are exchanged):

$$if\ |Q(A_1)_t| > |Q(A_2)_t|\ then \quad action(A_1)_t = 0\ else\ \ action(A_1)_t = 1$$

This solution assumes both agents know about their queue sizes. By following this algorithm, the agents will achieve a global goal, that is the average size of both queues will be minimal. From the individual point of view, each agent will not collide. We might say that they are *cooperating* with each other, although considering each agent's individual goal, this solution might lead to an unfair solution. This happens when one agent (e.g., A_1) always has the larger queue. In this case, it will always choose the green action while the other agent, that is cooperative, will have to choose the red action.

A possible cooperative and fair, non-adaptive, solution to the same problem is to let the agent that holds the larger queue choose the green action but only for a period of time equal to the difference of the queue sizes. Then, the other agent will be allowed to perform a green action, and then again they can compute who has the larger queue. As a consequence of this algorithm, if the same agent holds the shorter queue all the time, it will seldom be allowed to perform the green action, and thus ease the emptiness of its queue. Letting one agent choose the green action for a time period equal to the difference between both queue sizes causes the agents to get to a fair state where both hold queues of equal size.

The case study we face in this paper consists of two agents, A_1 and A_2, each one choosing an action $a^t_{A_i}$, at each period of time t. At time t, each agent knows its state $x^t_{A_i}$ (i.e., how many cars A_i has at time t and which are the actions performed by itself and the other agent at time t), and what is the reaction of its teacher (i.e., the feedback received for the actions performed at time t). After getting the feedback from its teacher, each agent is able to compute its level of satisfaction based on this feedback, the action it has performed itself, and the state that it is in (i.e., $x^t_{A_i}$). An adaptive criterion will be applied to this satisfaction level, that will instruct the agent to choose an action at time $t+1$. At time $t+1$, A_i recomputes its new queue, and it performs $a^{t+1}_{A_i}$ based on the adaptive criterion. At the same time, A_i receives A_j feedback value, and the whole cycle restarts.

Informally, we'd like to capture in this function the following (in this example we consider A_1):

- If A_1 chose to perform red then its own gain will be rather low because it is moving further from its goal of minimizing its queue size. The feedback it receives will be rather high because A_2 wants to encourage A_1 to keep choosing red.

- If A_1 chose to perform green, its own gain will be high and A_2 will want to punish A_1, proportionate to A_2's queue size. We can think of different feedback functions that can be chosen for the teacher. For example, the teacher might want to punish his learner each time his own queue's size increases.

3.2 The Learning Algorithms

In this section we present the learning algorithm that we have implemented. We have experimented with two different functions for computing the satisfaction level of the adaptive agents. The learning algorithm is based on a criterion applied to the level of the agents' satisfaction. The agents were programmed to adapt to each other by evaluating these criteria, and based on this they choose the action to perform. We also present different adaptive criteria we have implemented.

The first satisfaction function we implemented is given by Eq. 1.

$$S(A_i, t) = \left(\frac{\alpha}{Q^t_{A_i}} + \frac{\beta * collisions?}{Q^t_{A_i}}\right) + \left(\frac{\delta_1}{MaxQ} + \frac{\delta_2}{MaxQ}\right) \tag{1}$$

Both terms, the OwnGain, and the trainer's feedback, were weighted equally. *collisions*? will be 0 when no collisions occurred at time t, and -1 otherwise, thus decreasing the S value of agent A_i. δ_1 and δ_2 represent the level of agreement and disagreement of the other agent (A_j). More formally, δ_1 will decrease the S value, whenever A_j chose red, and δ_2 will add a positive value whenever A_i chose red, too.

The adaptive criteria we are testing using the S function shown in Eq. 1 are the following:

1. $if\ S(A_i,t) < \frac{\Sigma_{l=1}^{|window|} S(A_i,t-l)}{|window|}\ then$
   $a_i^{t+1} = 1 - a_i^t$
   $else\ a_i^{t+1} = a_i^t$

   i.e., change your last choice of action if your current satisfaction level has dropped off from the average level you had up to time t. Notice the special case when the history we let the agent remember (i.e., the window) is of length one. The agent A_i will change its last action if it has caused a decrease in the level of satisfaction it was in, at time t. We tested different lengths of windows in our experiments. In case $t < |window|$, then we regard there to be a window of size $t-1$.

2. $if\ S(A_i,t) < S(A_i,k)\ then$
   $a_i^{t+1} = a_i^k$
   $else\ a_i^{t+1} = 1 - a_i^k$
   where $k < t$, is the last time that $|Q|_{A_i}^k = |Q|_{A_i}^t$.

The second satisfaction function we have studied is given by Eq. 2.

$$\frac{\Sigma_{l=1}^4 \alpha_l * f_l}{\Sigma_{l=1}^4 \alpha_l} \tag{2}$$

where the first term represents the learner's own gain:

$$OwnGain = \alpha_1 * \frac{MaxQ - Q_{A_i}^t}{MaxQ} \tag{3}$$

and the teacher's feedback is given by

$$A_j'sfeedback = \alpha_2 * \frac{MaxQ - Q_{A-j}^t}{MaxQ} + \alpha_3 * \frac{collisions}{MaxQ} + \alpha_4 \frac{f_4}{MaxQ} \tag{4}$$

f_4 in Eq. 4 will be $MaxQ$ whenever both agents cooperate, 0 if a collision occurred and $\frac{MaxQ}{2}$ when both agents chose the red action.

The adaptive criteria implemented for this case were as follows:

1. $if\ S(A_i,t) < \frac{\Sigma_{l=1}^{|window|} S(A_i,t-l)}{|window|}\ then$
   $a_i^{t+1} = 1 - a_i^t$
   $else\ a_i^{t+1} = a_i^t$

2. let $update = S(A_i,t) - \frac{Sigma_{l=1}^{|window|} S(A_i,t-l)}{|window|}$
$if\ (update < 0)\ then\{$
$pr[a_i^t] += (1 - pr[a_i^t]) * update$
$pr[1 - a_i^t] = 1 - pr[a_i^t]\}$
$else\ \{pr[1 - a_i^t] += (1 - pr[1 - a_i^t]) * update$
$pr[a_i^t] = 1 - pr[1 - a_i^t]\}$
Choose a_i^{t+1} with probability $pr[a_i^t]$

i.e., there is a higher chance to choose an action that has improved the value of the S function during the history tested.

3.3 Preliminary Results

In the experiments we have conducted, we compared the performance of the adaptive agents vs. the performance of agents who choose their actions randomly. We ran the programs for a predefined number of runs. The parameters we evaluated included the number of collisions that occurred, the average size of the queues, the number of times that the agent with the larger queue chose the green action without colliding with the other agent, and the number of times both agents cooperated (i.e., chose complementary actions).

We have preliminary results only. Using the first satsfaction function (see Eq. 1), the agents succeeded in achieving 0% collisions, but at the cost of holding long queues, when the agents recall windows of size one. There weren't any significant differences in the results when we let the agents remember longer past histories (e.g., from size 2 to 1000). The agents cooperated in 49% of the cases. Using the second function (see Eq. 2), in less than 2% of the cases the agents' choices resulted in a collision. However, the agents held very long queues. When their memory was of size one, they only cooperate in less than 2% of the cases. When they recalled longer histories, they cooperate in 45% of the cases.

In general, we have not yet gotten results that we expected, i.e., the agents didn't learn strategies of behavior that will lead them to maintain their queues as small as possible and also avoid collisions. We intend to continue investigating other functions that are composed of the two main terms described in our approach. These terms regard the level of satisfaction of the learner agent due to its own actions, and a feedback value sent to the learner by the teacher, expressing the degree of approval of the teacher to the learner's behavior. We also intend to experiment with agents that are able to learn incrementally. This will be explained in the next section.

4 Incremental and Mutual Learning algorithm

In this section, we present a learning algorithm for a multiagent system in which the agents adapt to each other, and eventually learn to coordinate their behaviors incrementally and dynamically. We will still be considering the trainer-learner model. Hence, at each time, each agent plays the role of a trainer and also it plays the role of a learner. The trainer instructs the learner by giving it a feedback value. This value represents the teacher's opinion about the learner's actions. The learner decides upon its actions based on its satisfaction level, that is computed out of its own gain and the teacher's feedback. In the incremental case, the agents' choice of action will not be as simple as it was when they could choose one action out of the two possible ones. In this case, the agents will also be allowed to try a series of actions, and then, they will need to decide about their quality. That means that each agent computes how much it is satisfied after having performed a whole series of actions. We present the algorithm only. Its implementation remains for future work.

We want to test whether we can overcome the problems pointed out in Subsection 3.3, by letting the agents evolve cooperative behavior after incrementally learning which *series* of actions they should perform in order to please the other agent (the teacher).

Let us consider a series of n actions performed by one of the agents as a word of length n, taken from the alphabet composed of two possible letters 0 and 1. Then, the problem we are trying to solve is equivalent to the problem of finding the two words, in this language, one for each agent, such that they represent the cooperative and fair adaptive solution for the agents that achieves their goals. In other words, these words will be the minimal description of the cooperative adaptive solution.

The algorithm proposed for two agents is shown in Figure 3.

The agents will try iteratively a series of actions (i.e., words) of length *string-length*. The $\overline{S(A_i,t)}$ value for an agent and the word it has performed will be computed as the product of the S values for each action (i.e., letter) in the series. The probability for choosing a specific word will be increased if the $\overline{S}$ value for the word has increased; otherwise it will be decreased. $Ideal_i$ expresses the ideal satisfaction level of agent A_i. Whenever the agents are too far away from their ideal value, they will try a longer word. Notice that the agents will need to coordinate on the length of the series they are performing. We have considered several stopping conditions for this algorithm:

- Evaluate the words the agents are transmitting, after a specific number of iterations. Thus, we could test the relation between simulated time, and the performance of the agents.

- Evaluate the agents behavior when a specific length of string has been achieved.

- Evaluate the agents behavior after the difference between $Ideal_i - \overline{S(A_i,t)}$ has not changed for a specific number of times.

- Test the tradeoff between ϵ, that induces a cost of the learning process and the length of the strings the agents are trying.

```
string-length = 1
Actions = {0,1}
initialize probabilities(action): ∀ actions ∈ Actions prob(action)=1/|Actions|
Choose a_t^i with probability=pr(a_t^i)
Compute the current state of A_i, x_t^i
While (¬ StopCondition) {
   Compute S(A_i,t), given x_t^i
   If (Ideal_i − S(A_i,t) ≤ ε) {
      update probabilities of actions in Actions
      Compute x_{t+1}^i
      Choose a_{t+1}^i with probability=prob(a_{t+1}^i)
   }
   else {
     string-length++
     Compute new set of actions, Actions of size 2^{string−length}
     Initialize probabilities(action): ∀ actions ∈ Actions prob(action)=1/|Actions|
     Compute x_{t+1}^i
     Choose a_{t+1}^i with probability=pr(a_{t+1}^i)
    }
}
```

Figure 3: The incremental and mutual learning algorithm

5 Conclusions and Future Work

There are two interesting paradigms to investigate with regard to multiagent systems and learning. *Learning in a multiagent System* refers to agents acting in the same world, that have learning capabilities. The agents learn from the world and thus indirectly from the other agents that influence the environment. *Multiagent Learning Systems* deal with systems composed of different autonomous agents that learn directly from one another. We are interested in this second research direction. We have suggested the use of a trainer-learner model, where the trainer instructs the learner agent how the learner should behave in order to please the trainer. The learner chooses its actions according to its level of satisfaction, which is based on its own gain (resulting from its actions) and on the teacher's feedback.

We conducted experiments in the traffic light domain, with two satisfaction functions, and several adaptive criteria that the agents evaluated to decide upon the action they will execute. We have presented preliminary results. Moreover, we have suggested an algorithm for incrementally carrying out mutual adaptation. This algorithm reflects the idea of adaptive agents that learn which is the minimal word that will express a cooperative and fair adaptive solution.

Another issue that we will investigate in the future concerns multiagent systems composed of different types of agents. Interesting questions we'd like to answer include whether agents of specific types (e.g., altruistic, selfish, fair, etc.) can teach other agents to become agents of a desired type, and how randomly-behaved agents can learn to become agents of a specific type.

References

[1] Minoru Asada, Eiji Uchibe, and Koh Hosoda. Agents that learn from other competitive agents. In *Machine Learning: Proceedings of the Twelfth international conference, Workshop on Agents that Learn from Other Agents*, 1995.

[2] Claudia V. Goldman and Jeffrey S. Rosenschein. Mutually supervised learning in multiagent systems. *Adaptation and Learning in Multi-Agent Systems*, LNAI 1042:85–96, 1996.

[3] John J. Grefenstette. Lamarckian learning in multi-agent environments. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 303—310, 1991.

[4] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning: Proceedings of the Eleventh international conference*, pages 157—163, 1994.

[5] Maja J. Mataric. Learning to behave socially. In *Proceedings From Animals to Animats 3, Third International Conference on Simulation of Adaptive Behavior*, pages 157—163, 1994.

[6] Maja J. Mataric. Reward functions for accelerated learning. In *Machine Learning: Proceedings of the Eleventh international conference*, pages 157—163, 1994.

[7] Tuomas W. Sandholm and Robert H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems Journal Special Issue on the Prisoner's Dilemma*, submitted.

[8] S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, 1994.

[9] Sandip Sen and Mahendra Sekaran. Multiagent coordination with learning classifier systems. In *Workshop on Adaptation and Learningin Multiagent Systems at the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Quebec, 1995.

[10] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Machine Learning: Proceedings of the Tenth international conference*, pages 330—337, 1993.

[11] Christopher J.C.H. Watkins and Peter Dayan. Technical note Q-learning. *Machine Learning*, 8:279—292, 1992.