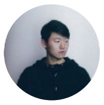
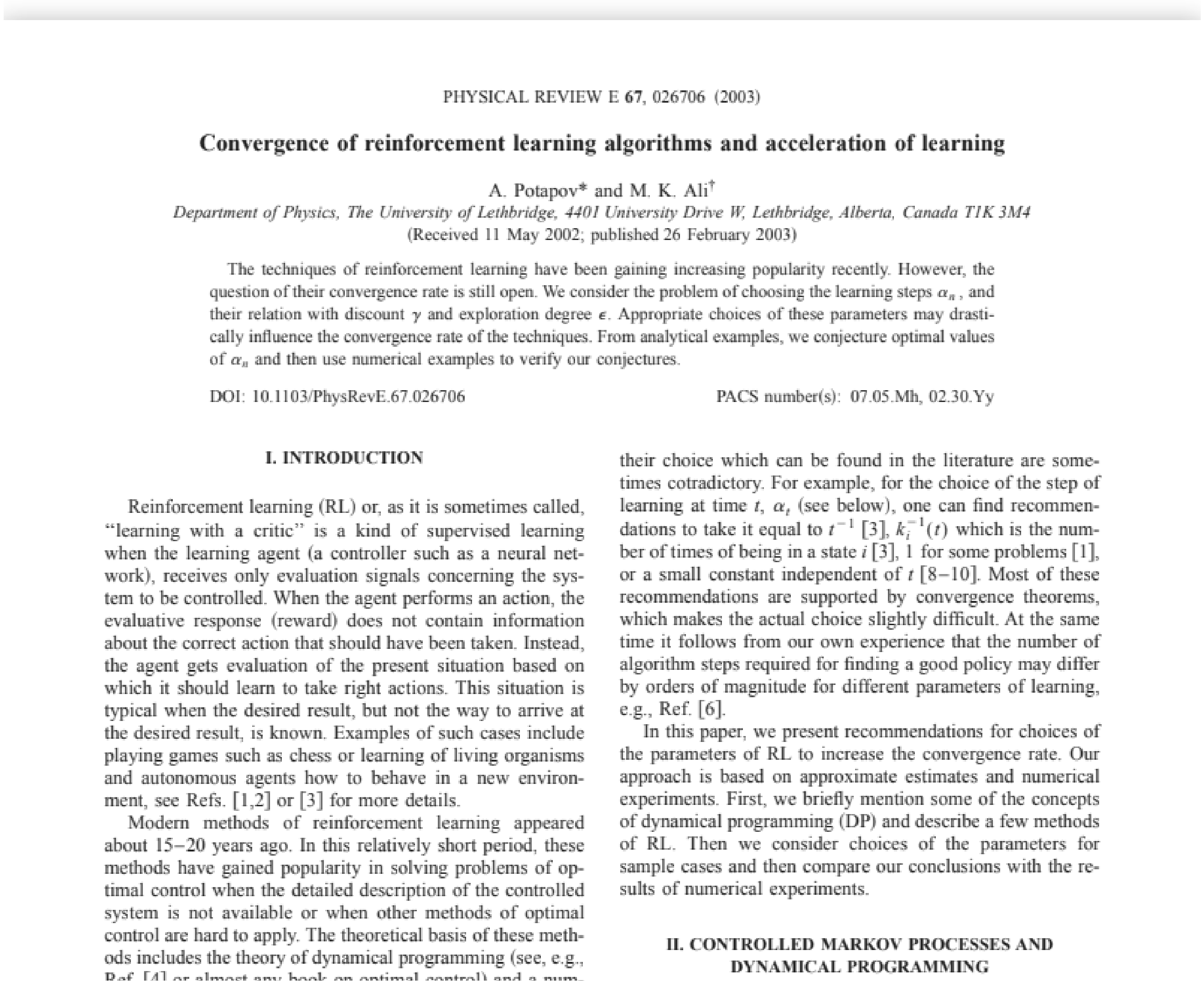
[Archived Post] Convergence of reinforcement learning algorithms and acceleration of learning



Jae Duk Seo Follow
Mar 17 · 6 min read

Please note that this post is for my own educational purpose.

. . .



PHYSICAL REVIEW E **67**, 026706 (2003)

Convergence of reinforcement learning algorithms and acceleration of learning

A. Potapov* and M. K. Ali†
Department of Physics, The University of Lethbridge, 4401 University Drive W, Lethbridge, Alberta, Canada T1K 3M4

(Received 11 May 2002; published 26 February 2003)

The techniques of reinforcement learning have been gaining increasing popularity recently. However, the question of their convergence rate is still open. We consider the problem of choosing the learning steps α_n, and their relation with discount γ and exploration degree ϵ. Appropriate choices of these parameters may drastically influence the convergence rate of the techniques. From analytical examples, we conjecture optimal values of α_n and then use numerical examples to verify our conjectures.

DOI: 10.1103/PhysRevE.67.026706 PACS number(s): 07.05.Mh, 02.30.Yy

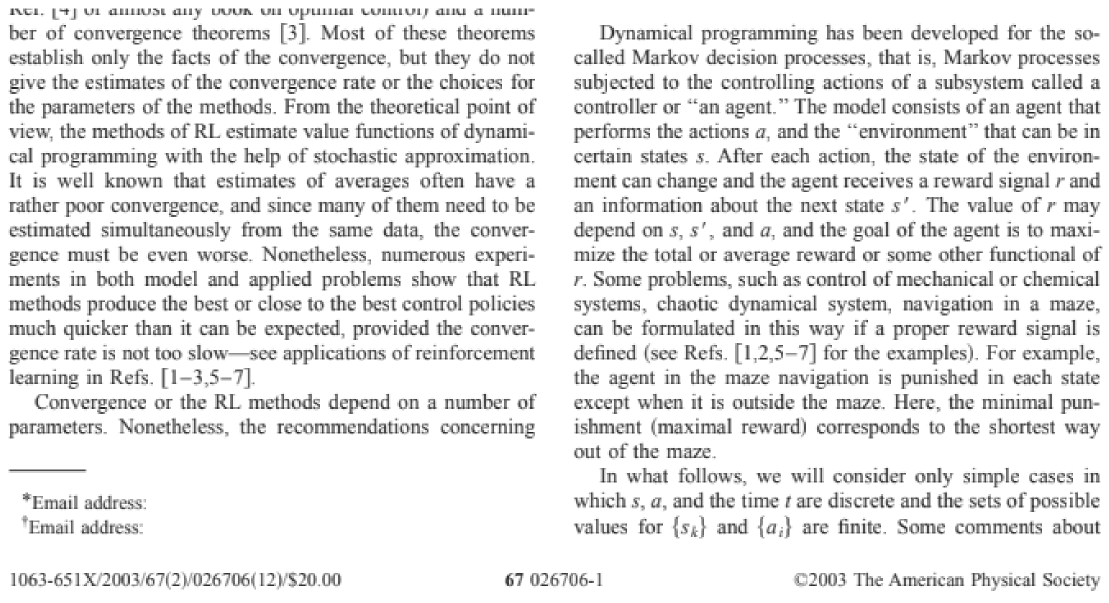
I. INTRODUCTION

Reinforcement learning (RL) or, as it is sometimes called, "learning with a critic" is a kind of supervised learning when the learning agent (a controller such as a neural network), receives only evaluation signals concerning the system to be controlled. When the agent performs an action, the evaluative response (reward) does not contain information about the correct action that should have been taken. Instead, the agent gets evaluation of the present situation based on which it should learn to take right actions. This situation is typical when the desired result, but not the way to arrive at the desired result, is known. Examples of such cases include playing games such as chess or learning of living organisms and autonomous agents how to behave in a new environment, see Refs. [1,2] or [3] for more details.

Modern methods of reinforcement learning appeared about 15–20 years ago. In this relatively short period, these methods have gained popularity in solving problems of optimal control when the detailed description of the controlled system is not available or when other methods of optimal control are hard to apply. The theoretical basis of these methods includes the theory of dynamical programming (see, e.g., Ref. [4] or almost any book on optimal control) and a num- their choice which can be found in the literature are sometimes cotradictory. For example, for the choice of the step of learning at time t, α_t (see below), one can find recommendations to take it equal to t^{-1} [3], $k_i^{-1}(t)$ which is the number of times of being in a state i [3], 1 for some problems [1], or a small constant independent of t [8–10]. Most of these recommendations are supported by convergence theorems, which makes the actual choice slightly difficult. At the same time it follows from our own experience that the number of algorithm steps required for finding a good policy may differ by orders of magnitude for different parameters of learning, e.g., Ref. [6].

In this paper, we present recommendations for choices of the parameters of RL to increase the convergence rate. Our approach is based on approximate estimates and numerical experiments. First, we briefly mention some of the concepts of dynamical programming (DP) and describe a few methods of RL. Then we consider choices of the parameters for sample cases and then compare our conclusions with the results of numerical experiments.

II. CONTROLLED MARKOV PROCESSES AND DYNAMICAL PROGRAMMING


Ref. [4] or almost any book on optimal control) and a number of convergence theorems [3]. Most of these theorems establish only the facts of the convergence, but they do not give the estimates of the convergence rate or the choices for the parameters of the methods. From the theoretical point of view, the methods of RL estimate value functions of dynamical programming with the help of stochastic approximation. It is well known that estimates of averages often have a rather poor convergence, and since many of them need to be estimated simultaneously from the same data, the convergence must be even worse. Nonetheless, numerous experiments in both model and applied problems show that RL methods produce the best or close to the best control policies much quicker than it can be expected, provided the convergence rate is not too slow—see applications of reinforcement learning in Refs. [1–3,5–7].

Convergence or the RL methods depend on a number of parameters. Nonetheless, the recommendations concerning

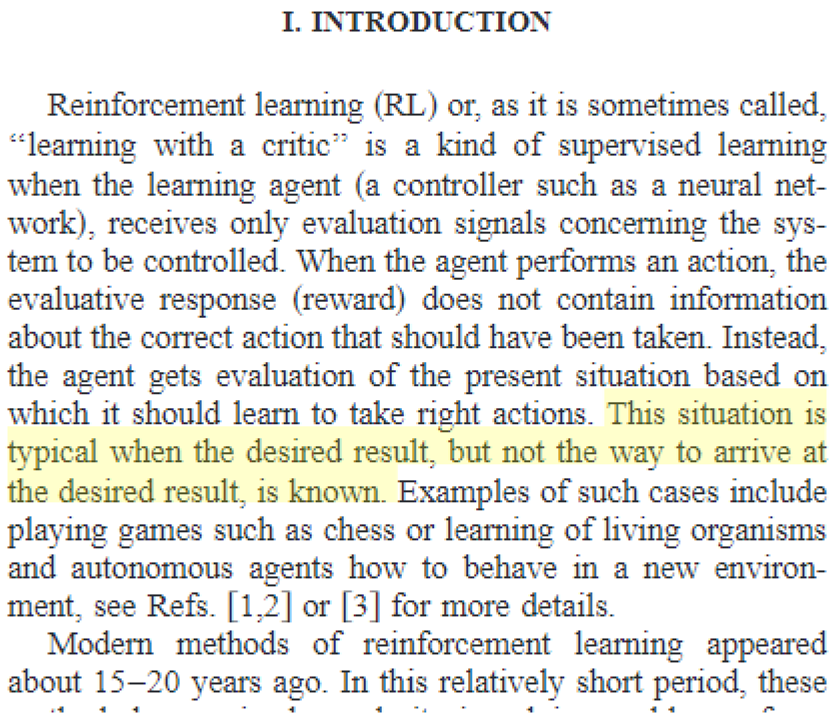
*Email address:

†Email address:

Dynamical programming has been developed for the so-called Markov decision processes, that is, Markov processes subjected to the controlling actions of a subsystem called a controller or "an agent." The model consists of an agent that performs the actions a, and the "environment" that can be in certain states s. After each action, the state of the environment can change and the agent receives a reward signal r and an information about the next state s'. The value of r may depend on s, s', and a, and the goal of the agent is to maximize the total or average reward or some other functional of r. Some problems, such as control of mechanical or chemical systems, chaotic dynamical system, navigation in a maze, can be formulated in this way if a proper reward signal is defined (see Refs. [1,2,5–7] for the examples). For example, the agent in the maze navigation is punished in each state except when it is outside the maze. Here, the minimal punishment (maximal reward) corresponds to the shortest way out of the maze.

In what follows, we will consider only simple cases in which s, a, and the time t are discrete and the sets of possible values for $\{s_k\}$ and $\{a_i\}$ are finite. Some comments about

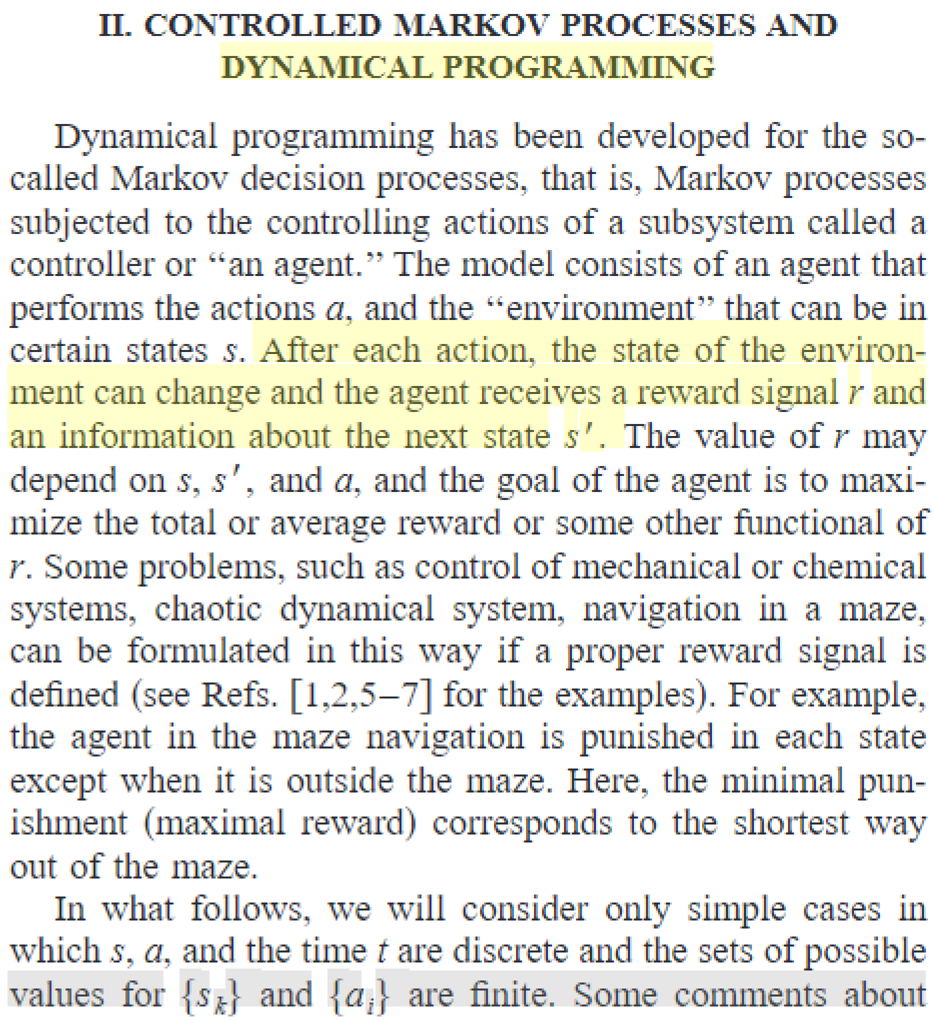
1063-651X/2003/67(2)/026706(12)/$20.00 **67** 026706-1 ©2003 The American Physical Society


I. INTRODUCTION

Reinforcement learning (RL) or, as it is sometimes called, "learning with a critic" is a kind of supervised learning when the learning agent (a controller such as a neural network), receives only evaluation signals concerning the system to be controlled. When the agent performs an action, the evaluative response (reward) does not contain information about the correct action that should have been taken. Instead, the agent gets evaluation of the present situation based on which it should learn to take right actions. This situation is typical when the desired result, but not the way to arrive at the desired result, is known. Examples of such cases include playing games such as chess or learning of living organisms and autonomous agents how to behave in a new environment, see Refs. [1,2] or [3] for more details.

Modern methods of reinforcement learning appeared about 15–20 years ago. In this relatively short period, these

RL → learn from experience → and this experience is gathered overtime → not given from the start. (it appeared 15 -20 years ago → but optimal control is possible → very powerful method of controlling). (Solutions include dynamical programming with control → convergence). (RL → estimation → rather have poor convergence → but empirical results → show the RL method gives good policy.). (there seems to be some contradiction within the convergence analysis). (recommendation → supported by convergence theorem). (this paper → make a recommendation of parameters).

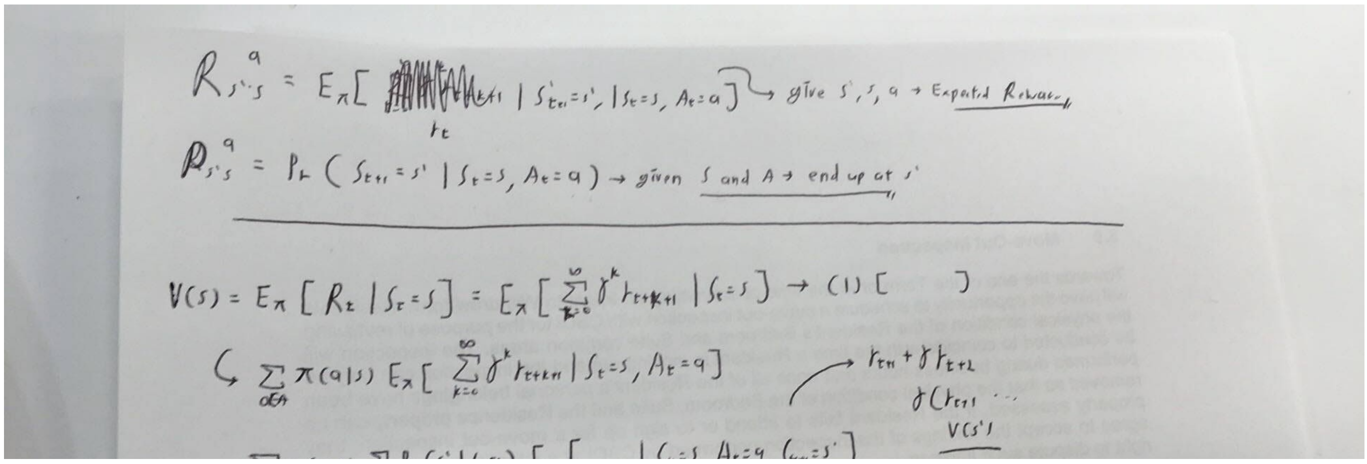


II. CONTROLLED MARKOV PROCESSES AND DYNAMICAL PROGRAMMING

Dynamical programming has been developed for the so-called Markov decision processes, that is, Markov processes subjected to the controlling actions of a subsystem called a controller or "an agent." The model consists of an agent that performs the actions a, and the "environment" that can be in certain states s. After each action, the state of the environment can change and the agent receives a reward signal r and an information about the next state s'. The value of r may depend on s, s', and a, and the goal of the agent is to maximize the total or average reward or some other functional of r. Some problems, such as control of mechanical or chemical systems, chaotic dynamical system, navigation in a maze, can be formulated in this way if a proper reward signal is defined (see Refs. [1,2,5–7] for the examples). For example, the agent in the maze navigation is punished in each state except when it is outside the maze. Here, the minimal punishment (maximal reward) corresponds to the shortest way out of the maze.

In what follows, we will consider only simple cases in which s, a, and the time t are discrete and the sets of possible values for $\{s_k\}$ and $\{a_i\}$ are finite. Some comments about

DP → good for the markov decision process → control action of the sub-system and more → it can be in certain state s → receive a reward → maximize total or average reward. (the dynamic system → can be formulated in this method).



$R^{a}_{s's} = E_\pi[\underset{r_t}{[illegible]}{}_{t+1} \mid S'_{t+1}=s', \mid S_t=s, A_t=a] \hookrightarrow$ give $s', s, a \rightarrow$ Expected Reward

$P^{a}_{s's} = P_r(S_{t+1}=s' \mid S_t=s, A_t=a) \rightarrow$ given S and $A \rightarrow$ end up at s'

$V(s) = E_\pi[R_t \mid S_t=s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t=s] \rightarrow (1)\,[\quad]$

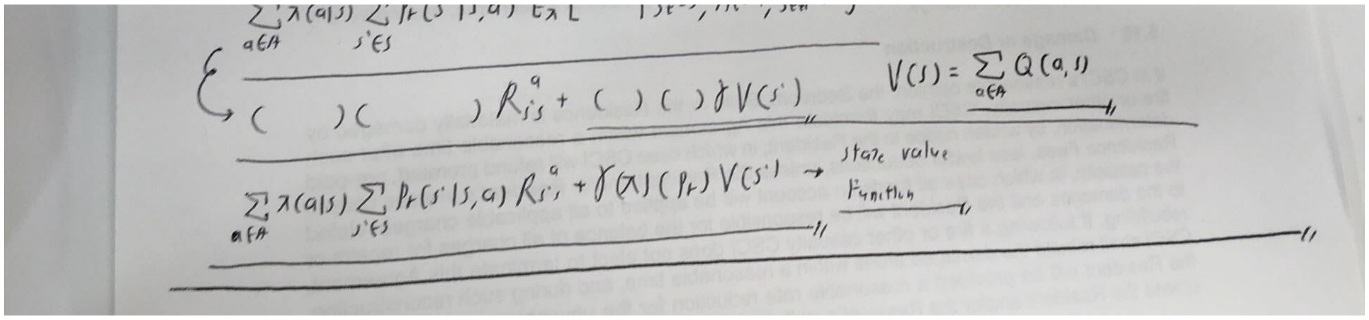
$\hookrightarrow \sum_{a\in A} \pi(a|s)\, E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t=s, A_t=a]$

$\rightarrow r_{t+1} + \gamma r_{t+2}$

$\gamma(r_{t+1} \cdots$

$V(s')$

[illegible]



$\sum_{a\in A}\pi(a|s)\sum_{s'\in S}Pr(s'|s,a)E_\pi[\quad|S_t=s',$

$V(s)=\sum_{a\in A}Q(a,s)$

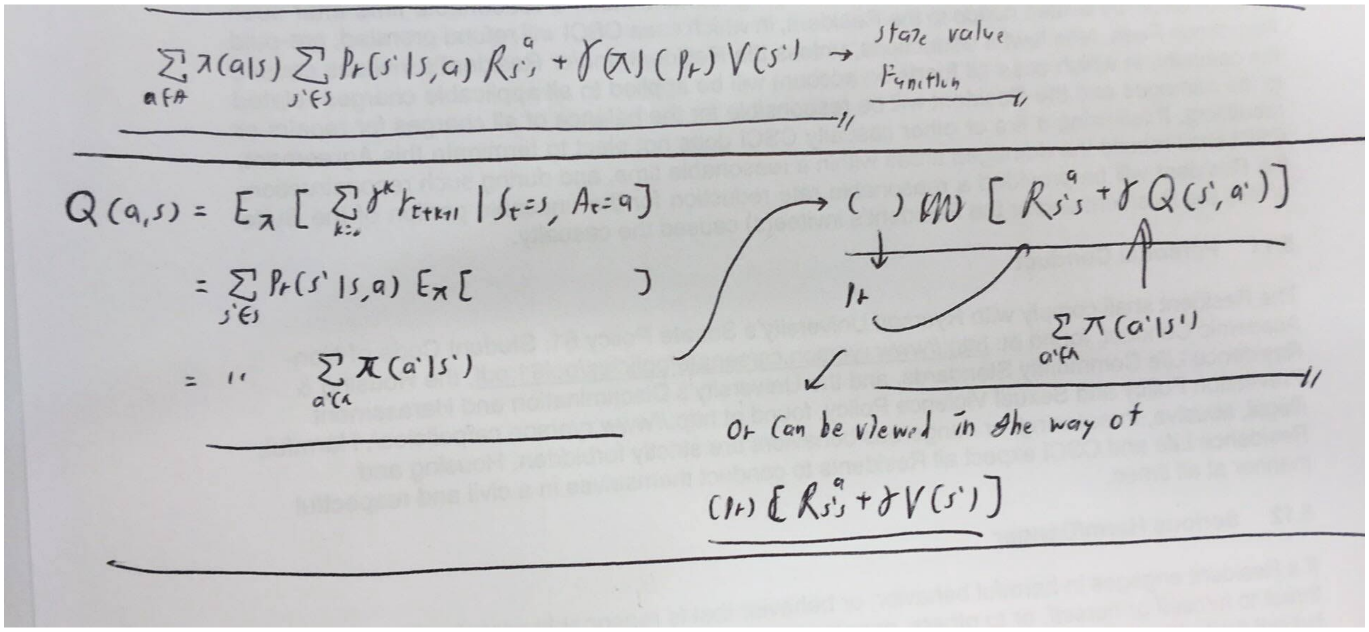
$(\quad)(\quad)R^a_{ss'}+(\quad)(\quad)\gamma V(s')$

$\sum_{a\in A}\pi(a|s)\sum_{s'\in S}Pr(s'|s,a)R^a_{ss'}+\gamma(\pi)(Pr)V(s')\rightarrow$ state value Function

A policy is a solution and state value function can be formulated like below.

$$V(s)=\sum_{a}\pi_a(s)\sum_{s'}P(s,a,s')R(s,a,s')$$
$$+\gamma\sum_{a'}\pi_{a'}(s')\sum_{s''}P(s',a',s'')$$
$$\times\left(R(s',a',s'')+\gamma\sum_{a''}\cdots\right)$$
$$=\sum_{a}\pi_a(s)\sum_{s'}P(s,a,s')[R(s,a,s')+\gamma V(s')]. \quad (1)$$

And this can be seen as a system of linear equations. (for math to work out → need discount). (and called bellman's equation → solutions are unique → now we can be more specific → add the actions for each case).



$\sum_{a\in A}\pi(a|s)\sum_{s'\in S}Pr(s'|s,a)R^a_{ss'}+\gamma(\pi)(Pr)V(s')\rightarrow$ state value Function

$Q(a,s)=E_\pi[\sum_{k=0}\gamma^k r_{t+k+1}|S_t=s,A_t=a]\rightarrow(\quad)(m)[R^a_{ss'}+\gamma Q(s',a')]$

$=\sum_{s'\in S}Pr(s'|s,a)E_\pi[\quad]$

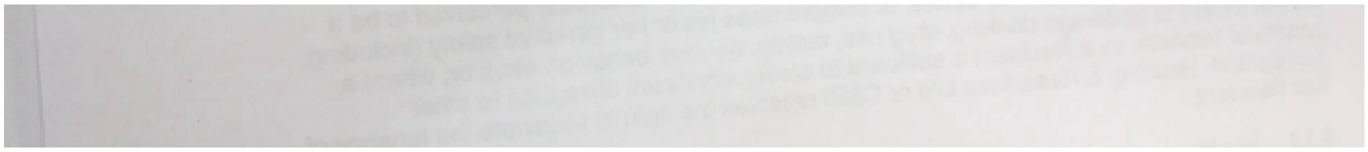
$=\ ''\ \sum_{a'\in A}\pi(a'|s')$

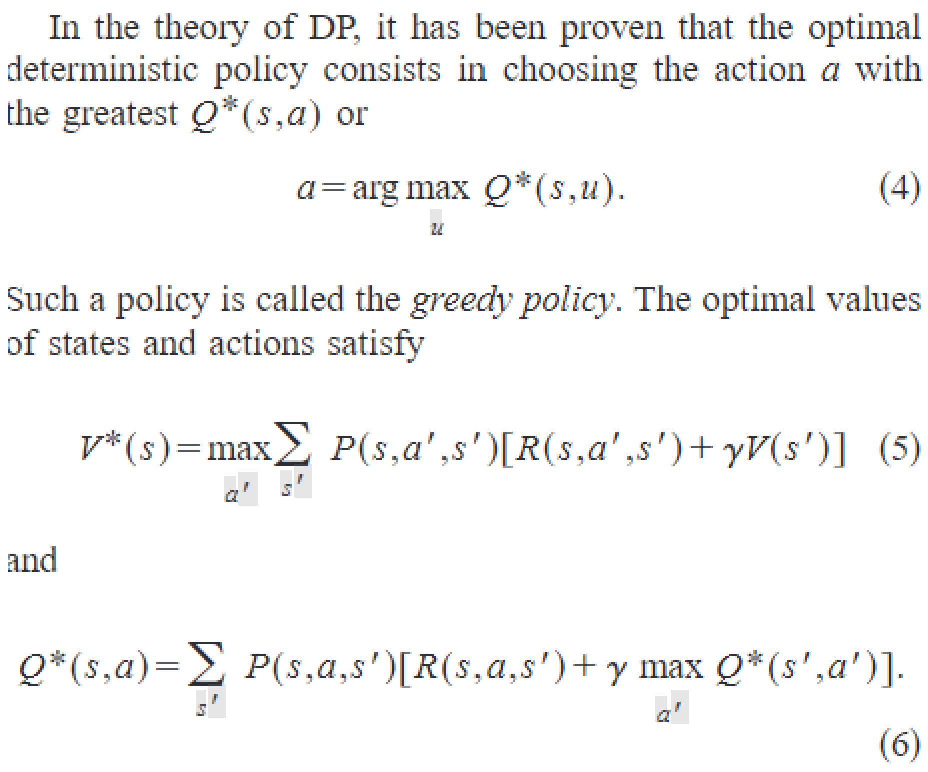
Pr

$\sum_{a'\in A}\pi(a'|s')$

Or Can be viewed in the way of

$(Pr)[R^a_{ss'}+\gamma V(s')]$





In the theory of DP, it has been proven that the optimal deterministic policy consists in choosing the action a with the greatest $Q^*(s,a)$ or

$$a = \arg\max_{u} Q^*(s,u). \tag{4}$$

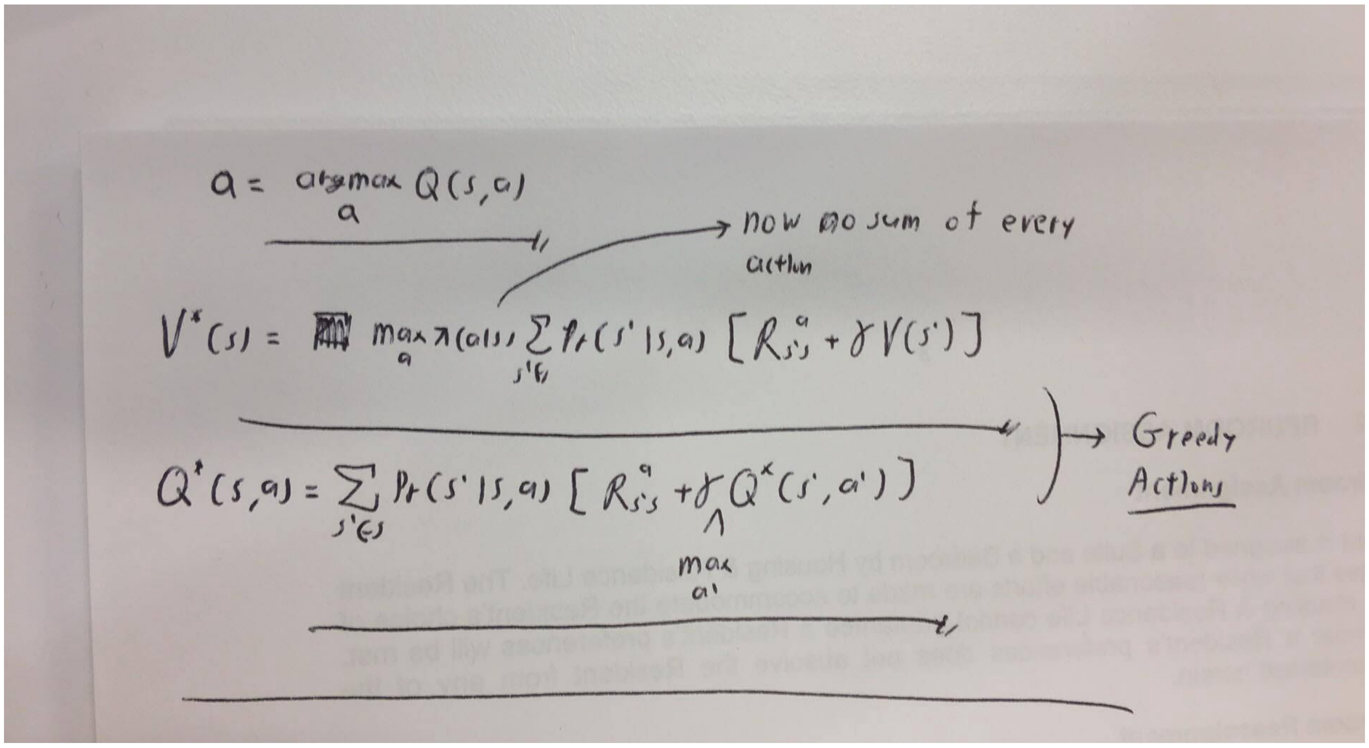
Such a policy is called the *greedy policy*. The optimal values of states and actions satisfy

$$V^*(s) = \max_{a'} \sum_{s'} P(s,a',s')[R(s,a',s') + \gamma V(s')] \tag{5}$$

and

$$Q^*(s,a) = \sum_{s'} P(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')]. \tag{6}$$

And with the greedy policy, → we would always choose the action that gives us the best reward.



$$a = \arg\max_{a} Q(s,a)$$

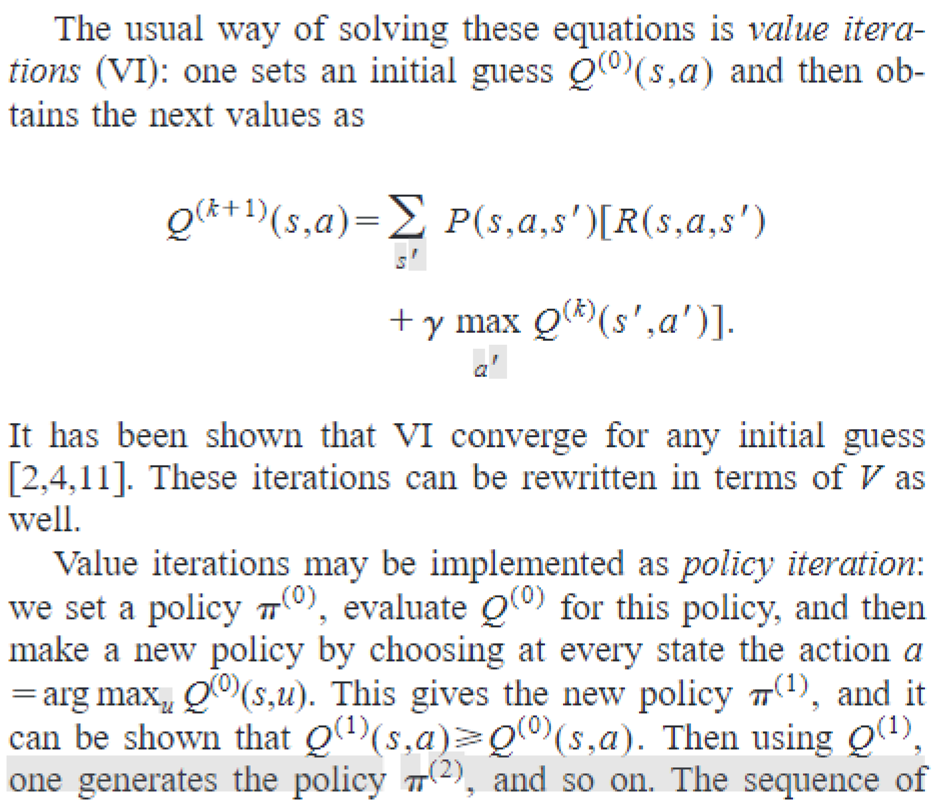
→ now no sum of every action

$$V^*(s) = \max_{a} \pi(a|s) \sum_{s' \in} Pr(s'|s,a)\left[R^{a}_{ss'} + \gamma V(s')\right]$$

$$Q^*(s,a) = \sum_{s' \in S} Pr(s'|s,a)\left[R^{a}_{ss'} + \gamma \overset{\wedge}{\underset{a'}{\max}} Q^*(s',a')\right]$$

→ Greedy Actions

And now → we do not have to consider → every action.



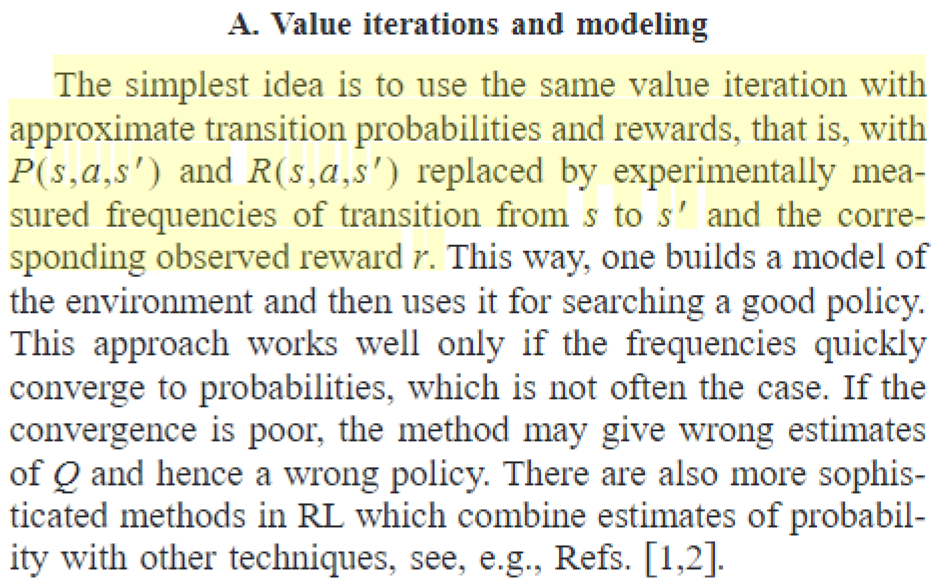
The usual way of solving these equations is *value iterations* (VI): one sets an initial guess $Q^{(0)}(s,a)$ and then obtains the next values as

$$Q^{(k+1)}(s,a)=\sum_{s'} P(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q^{(k)}(s',a')].$$

It has been shown that VI converge for any initial guess [2,4,11]. These iterations can be rewritten in terms of V as well.

Value iterations may be implemented as *policy iteration*: we set a policy $\pi^{(0)}$, evaluate $Q^{(0)}$ for this policy, and then make a new policy by choosing at every state the action $a = \arg\max_u Q^{(0)}(s,u)$. This gives the new policy $\pi^{(1)}$, and it can be shown that $Q^{(1)}(s,a) \geqslant Q^{(0)}(s,a)$. Then using $Q^{(1)}$, one generates the policy $\pi^{(2)}$, and so on. The sequence of

And these methods → can be solved by value or policy iterations. (very sexy convergence). (but what if we do not know the transition probability and the rewards).



A. Value iterations and modeling

The simplest idea is to use the same value iteration with approximate transition probabilities and rewards, that is, with $P(s,a,s')$ and $R(s,a,s')$ replaced by experimentally measured frequencies of transition from s to s' and the corresponding observed reward r. This way, one builds a model of the environment and then uses it for searching a good policy. This approach works well only if the frequencies quickly converge to probabilities, which is not often the case. If the convergence is poor, the method may give wrong estimates of Q and hence a wrong policy. There are also more sophisticated methods in RL which combine estimates of probability with other techniques, see, e.g., Refs. [1,2].

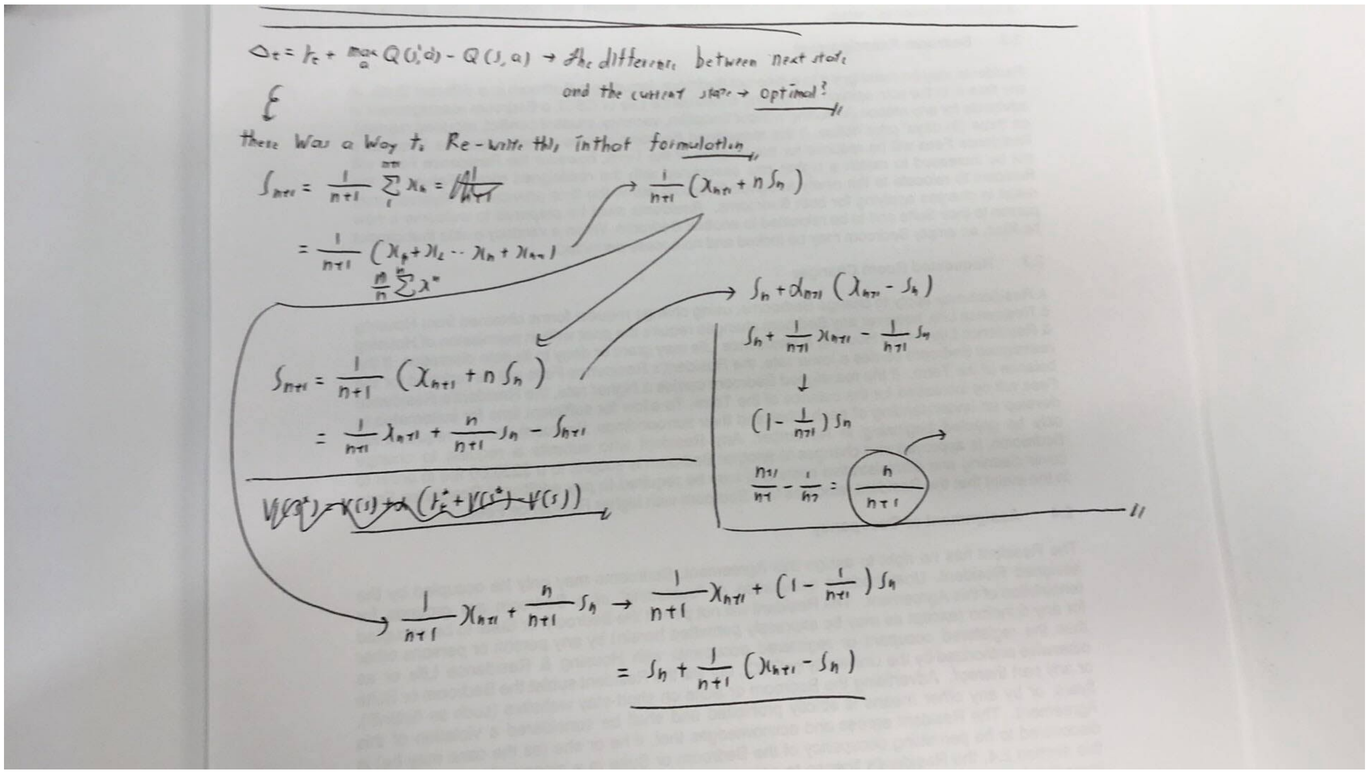
We can build the model of the world → and see how it goes from there → and build on top of it → bootstrapping.

B. Q learning or use of stochastic approximation to solve Eq. (6)

Equation (6) can be rewritten as $\langle R(s,a)\rangle + \gamma\langle \max_{a'} Q^*(s',a')\rangle - Q^*(s,a) = 0$. Current estimates of $Q(s,a)$ differ from Q^*, therefore observations do not provide mean values, and only the values

$$\Delta_t = r_t + \gamma \max_{a'} Q^{(t)}(s_{t+1},a') - Q^{(t)}(s_t,a_t)$$

The above actually does lead to optimal policy.



$\Delta_t = r_t + \max_a Q(s',a) - Q(s,a)$ → the difference between next state and the current state → optimal?

there was a way to re-write this in that formulation

$S_{n+1} = \frac{1}{n+1}\sum_{1}^{n+1} x_n = \cancel{\frac{1}{n+1}}$

$= \frac{1}{n+1}(x_1 + x_2 \cdots x_n + x_{n+1})$

$\frac{n}{n}\sum^{n} x^n$

$\to \frac{1}{n+1}(x_{n+1} + nS_n)$

$S_{n+1} = \frac{1}{n+1}(x_{n+1} + nS_n)$

$= \frac{1}{n+1}x_{n+1} + \frac{n}{n+1}S_n - S_{n+1}$

~~$V(s^*) = V(s) + \alpha(r_t + V(s^*) - V(s))$~~

$\to S_n + \alpha_{n+1}(x_{n+1} - S_n)$

$S_n + \frac{1}{n+1}x_{n+1} - \frac{1}{n+1}S_n$

$\downarrow$

$(1 - \frac{1}{n+1})S_n$

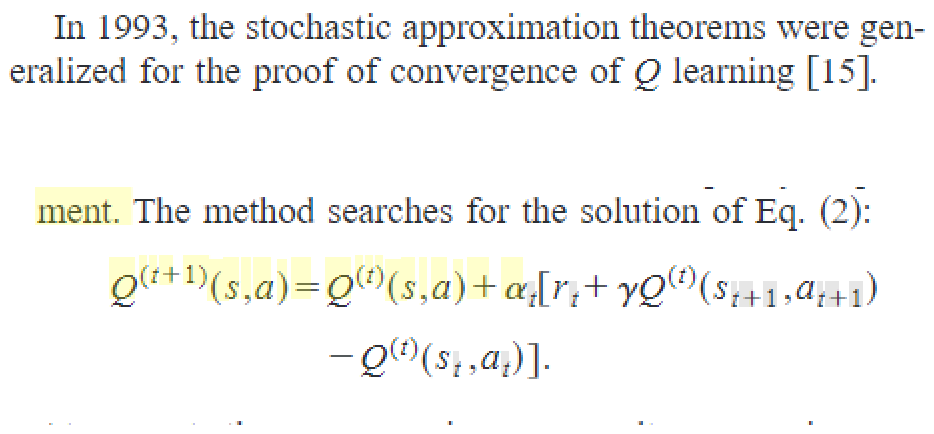
$\frac{n+1}{n+1} - \frac{1}{n+1} = \boxed{\frac{n}{n+1}}$

$\frac{1}{n+1}x_{n+1} + \frac{n}{n+1}S_n \to \frac{1}{n+1}x_{n+1} + (1 - \frac{1}{n+1})S_n$

$= S_n + \frac{1}{n+1}(x_{n+1} - S_n)$

This method is called stochastic approximation → damn that's sexy. (find a solution x star → of an equation f() → above method can be deployed.

$$Q^{(t+1)}(s_t,a_t) = Q^{(t)}(s_t,a_t) + \alpha_t[r_t + \gamma \max_{a'} Q^{(t)}(s_{t+1},a') - Q^{(t)}(s_t,a_t)].$$

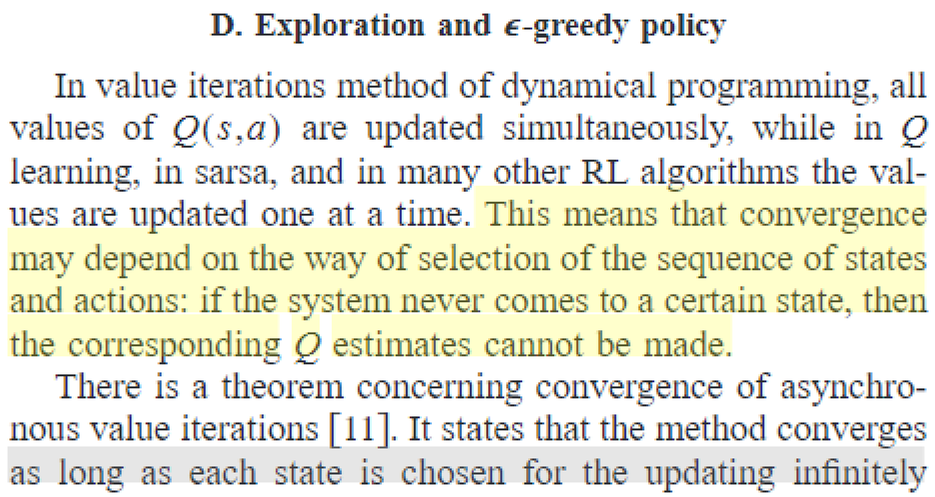


In 1993, the stochastic approximation theorems were generalized for the proof of convergence of Q learning [15].

ment. The method searches for the solution of Eq. (2):

$$Q^{(t+1)}(s,a)=Q^{(t)}(s,a)+\alpha_t[r_t+\gamma Q^{(t)}(s_{t+1},a_{t+1}) - Q^{(t)}(s_t,a_t)].$$

Now we can extend this solution to SARSA. (super cool). (there are so many problems to solve).

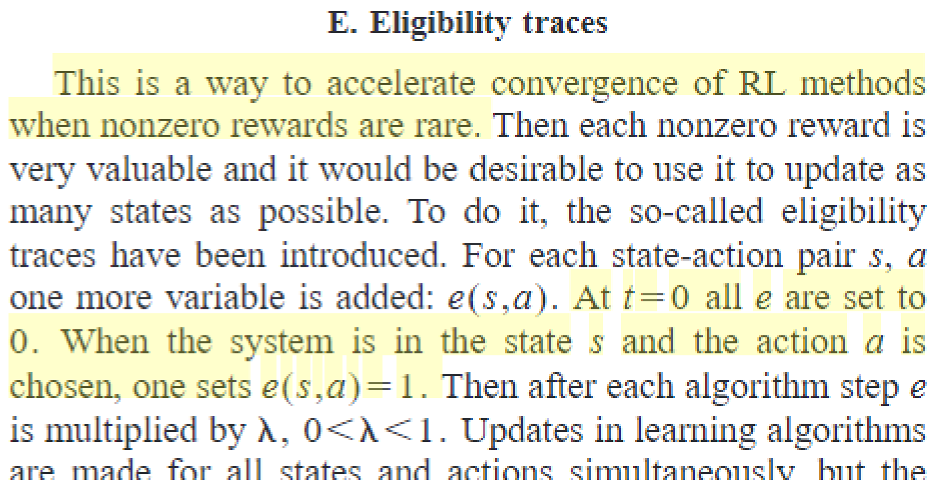


D. Exploration and ϵ-greedy policy

In value iterations method of dynamical programming, all values of $Q(s,a)$ are updated simultaneously, while in Q learning, in sarsa, and in many other RL algorithms the values are updated one at a time. This means that convergence may depend on the way of selection of the sequence of states and actions: if the system never comes to a certain state, then the corresponding Q estimates cannot be made.

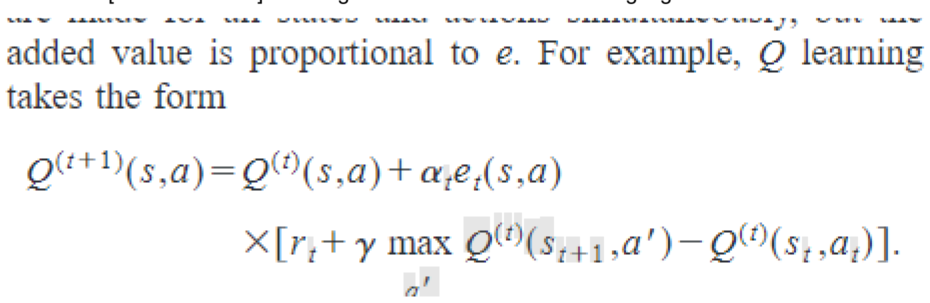
There is a theorem concerning convergence of asynchronous value iterations [11]. It states that the method converges as long as each state is chosen for the updating infinitely

All values are updated → at the same time → or it can be one at a time → this can affect convergence. (so how can we make sure that the exploration is good). (exploration is one solution for optimal convergence).



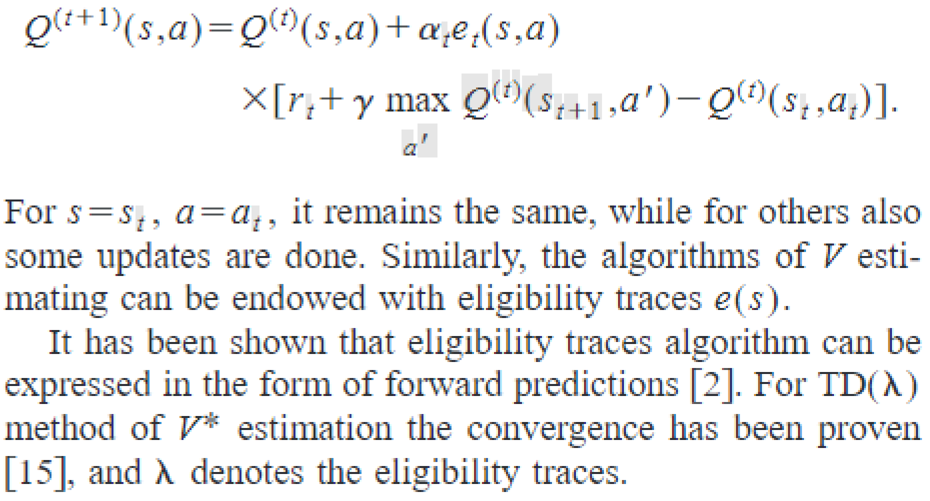
E. Eligibility traces

This is a way to accelerate convergence of RL methods when nonzero rewards are rare. Then each nonzero reward is very valuable and it would be desirable to use it to update as many states as possible. To do it, the so-called eligibility traces have been introduced. For each state-action pair s, a one more variable is added: $e(s,a)$. At $t=0$ all e are set to 0. When the system is in the state s and the action a is chosen, one sets $e(s,a)=1$. Then after each algorithm step e is multiplied by λ, $0<\lambda<1$. Updates in learning algorithms are made for all states and actions simultaneously, but the


added value is proportional to e. For example, Q learning takes the form

$$Q^{(t+1)}(s,a)=Q^{(t)}(s,a)+\alpha_t e_t(s,a) \times [r_t+\gamma \max_{a'} Q^{(t)}(s_{t+1},a')-Q^{(t)}(s_t,a_t)].$$

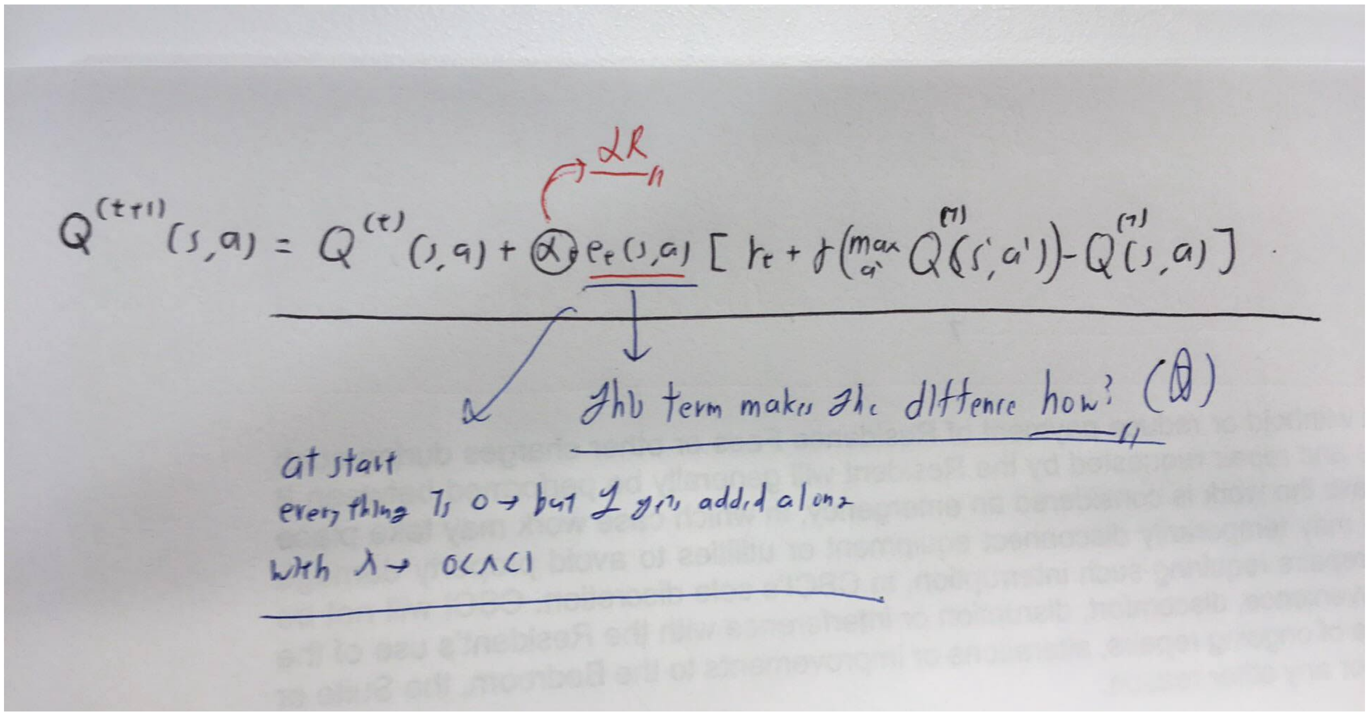
If non-zero rewards are rare → most rewards have to value → this is good information.


$$Q^{(t+1)}(s,a)=Q^{(t)}(s,a)+\alpha_t e_t(s,a) \times [r_t+\gamma \max_{a'} Q^{(t)}(s_{t+1},a')-Q^{(t)}(s_t,a_t)].$$

For $s=s_t$, $a=a_t$, it remains the same, while for others also some updates are done. Similarly, the algorithms of V estimating can be endowed with eligibility traces $e(s)$.

It has been shown that eligibility traces algorithm can be expressed in the form of forward predictions [2]. For TD(λ) method of V^* estimation the convergence has been proven [15], and λ denotes the eligibility traces.

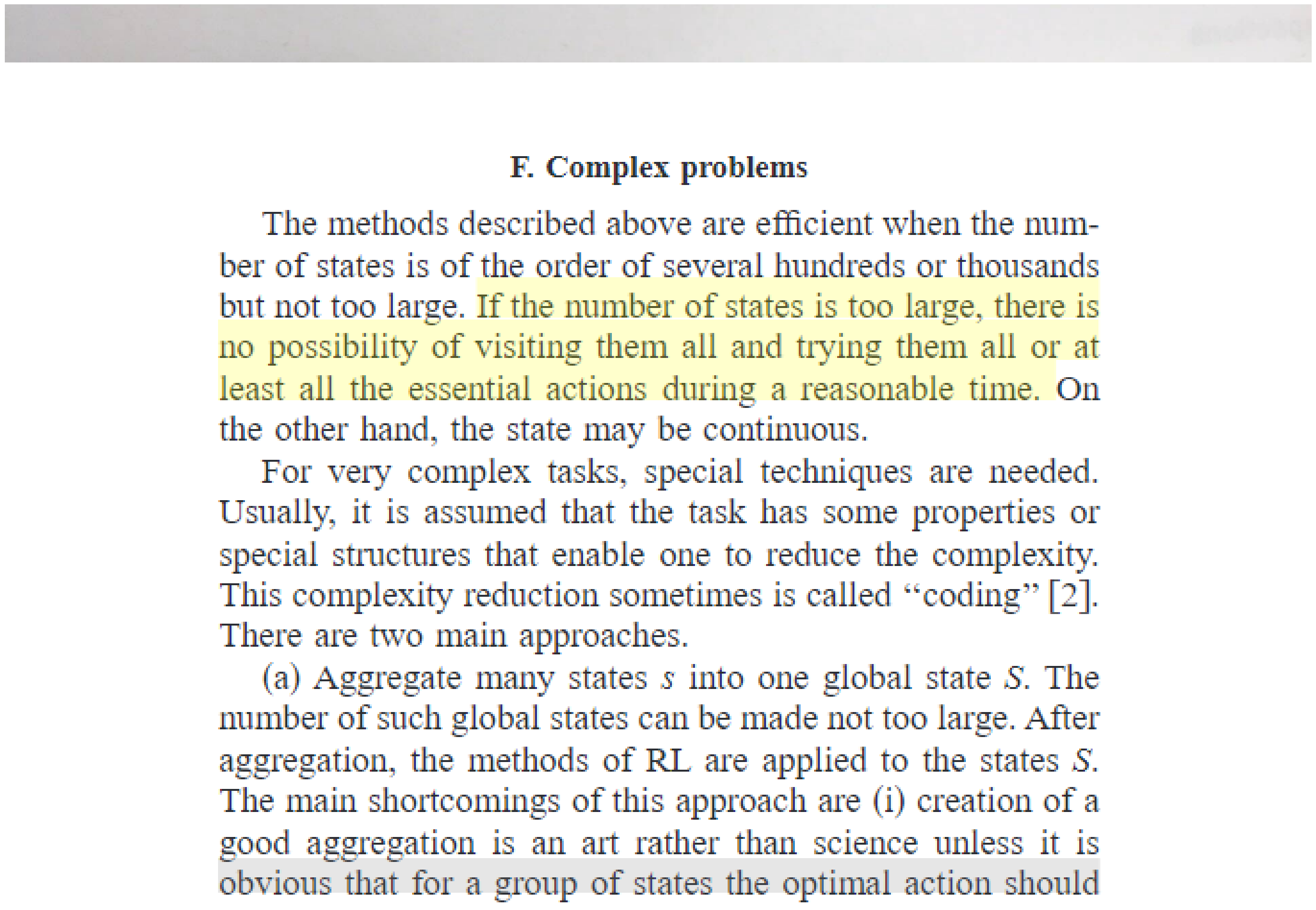
And adding one element → can make all of the difference → with how the optimization process is made. As well as → which part of the action was wrong → does not seem so scalable.


$$Q^{(t+1)}(s,a) = Q^{(t)}(s,a) + \boxed{\alpha}\, \underline{e_t(s,a)} \left[r_t + \gamma\left(\max_{a'} Q^{(t)}(s',a')\right) - Q^{(t)}(s,a) \right]$$

α → LR

This term makes the difference how? (α)

at start
everything is 0 → but 1 gets added along
with λ → $0<\lambda<1$



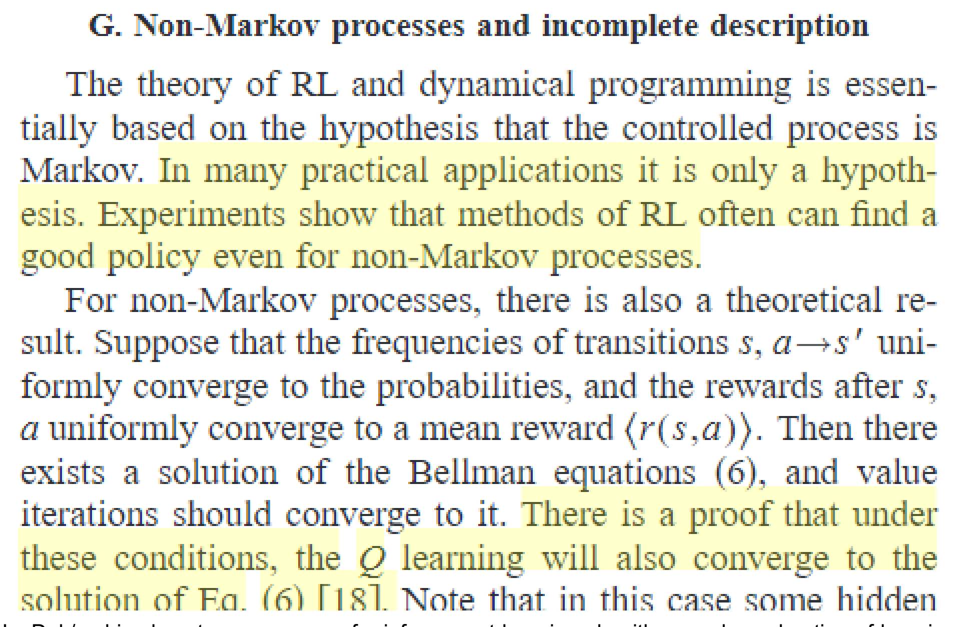
F. Complex problems

The methods described above are efficient when the number of states is of the order of several hundreds or thousands but not too large. If the number of states is too large, there is no possibility of visiting them all and trying them all or at least all the essential actions during a reasonable time. On the other hand, the state may be continuous.

For very complex tasks, special techniques are needed. Usually, it is assumed that the task has some properties or special structures that enable one to reduce the complexity. This complexity reduction sometimes is called "coding" [2]. There are two main approaches.

(a) Aggregate many states s into one global state S. The number of such global states can be made not too large. After aggregation, the methods of RL are applied to the states S. The main shortcomings of this approach are (i) creation of a good aggregation is an art rather than science unless it is obvious that for a group of states the optimal action should

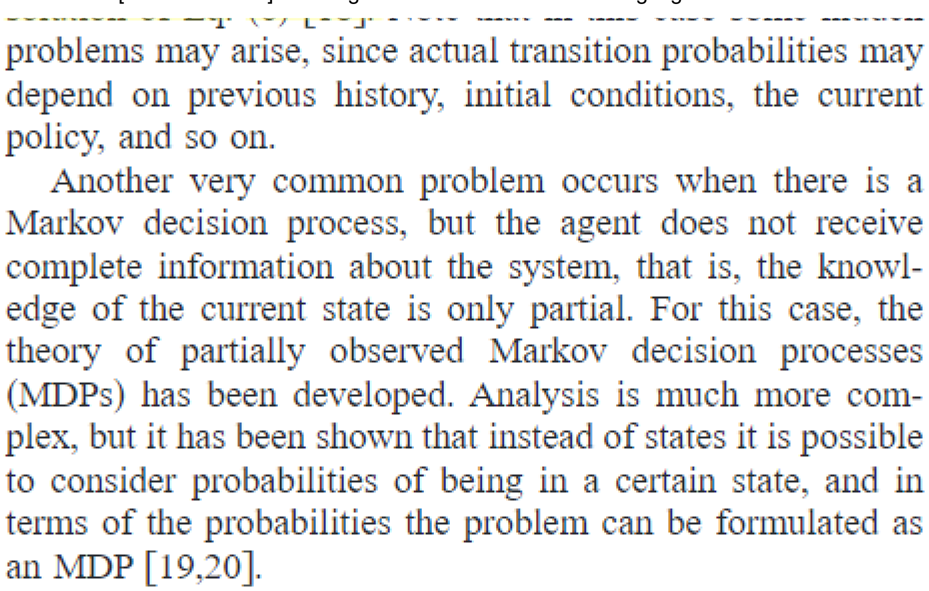
Again → the problems are not solvable when there are many states → so we need to think about this as well. (complexity reduction → coding → such as sparse coding → we can aggregate many states into one state). (aggregation → destroys Markov properties). (Neural network and linear functions are good choices) (TD Gammon → the most successful case → for this paper → direct policy estimation also exist).



G. Non-Markov processes and incomplete description

The theory of RL and dynamical programming is essentially based on the hypothesis that the controlled process is Markov. In many practical applications it is only a hypothesis. Experiments show that methods of RL often can find a good policy even for non-Markov processes.

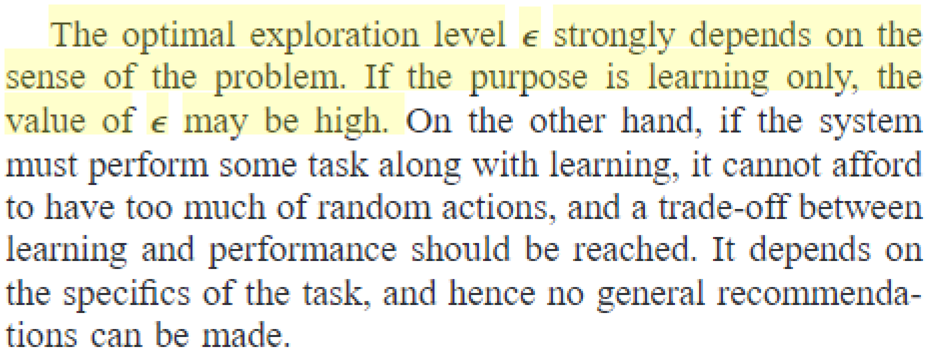
For non-Markov processes, there is also a theoretical result. Suppose that the frequencies of transitions $s, a \rightarrow s'$ uniformly converge to the probabilities, and the rewards after s, a uniformly converge to a mean reward $\langle r(s,a) \rangle$. Then there exists a solution of the Bellman equations (6), and value iterations should converge to it. There is a proof that under these conditions, the Q learning will also converge to the solution of Eq. (6) [18]. Note that in this case some hidden



problems may arise, since actual transition probabilities may depend on previous history, initial conditions, the current policy, and so on.

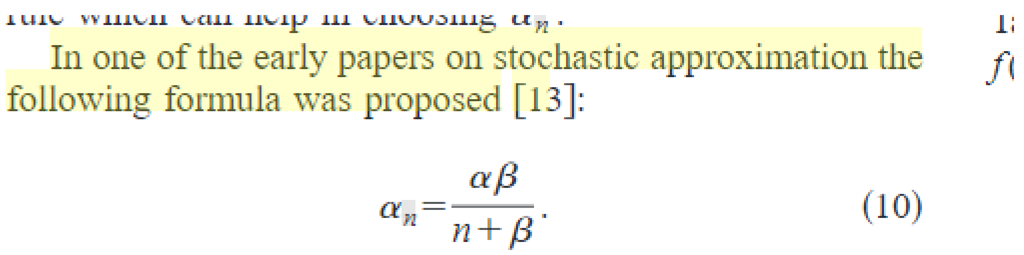
Another very common problem occurs when there is a Markov decision process, but the agent does not receive complete information about the system, that is, the knowledge of the current state is only partial. For this case, the theory of partially observed Markov decision processes (MDPs) has been developed. Analysis is much more complex, but it has been shown that instead of states it is possible to consider probabilities of being in a certain state, and in terms of the probabilities the problem can be formulated as an MDP [19,20].

When the control process is not Markov → This is not a good thing → but the algorithms seem to do converge into a good place. (the Q learning will also converge). (partial observable problem also a good problem).



The optimal exploration level ϵ strongly depends on the sense of the problem. If the purpose is learning only, the value of ϵ may be high. On the other hand, if the system must perform some task along with learning, it cannot afford to have too much of random actions, and a trade-off between learning and performance should be reached. It depends on the specifics of the task, and hence no general recommendations can be made.

The convergence → really does depend on the choice of hyperparameters. (such as discount rate → depending on this → we may not even ever converge). (there are a contradictory recommendation of how to set the hyperparameters).



rule which can help in choosing α_n.

In one of the early papers on stochastic approximation the following formula was proposed [13]:

$$\alpha_n = \frac{\alpha\beta}{n+\beta}. \qquad (10)$$

The above rule was proposed → a special case of a is also driveable. (this is very cool). (now we are going to see how the math works out for the recommendation). (we are going to first think about the deterministic case).

$$Q_{n+1}(s,a) = Q_n(s,a) + \alpha f(\{Q_n\}).$$

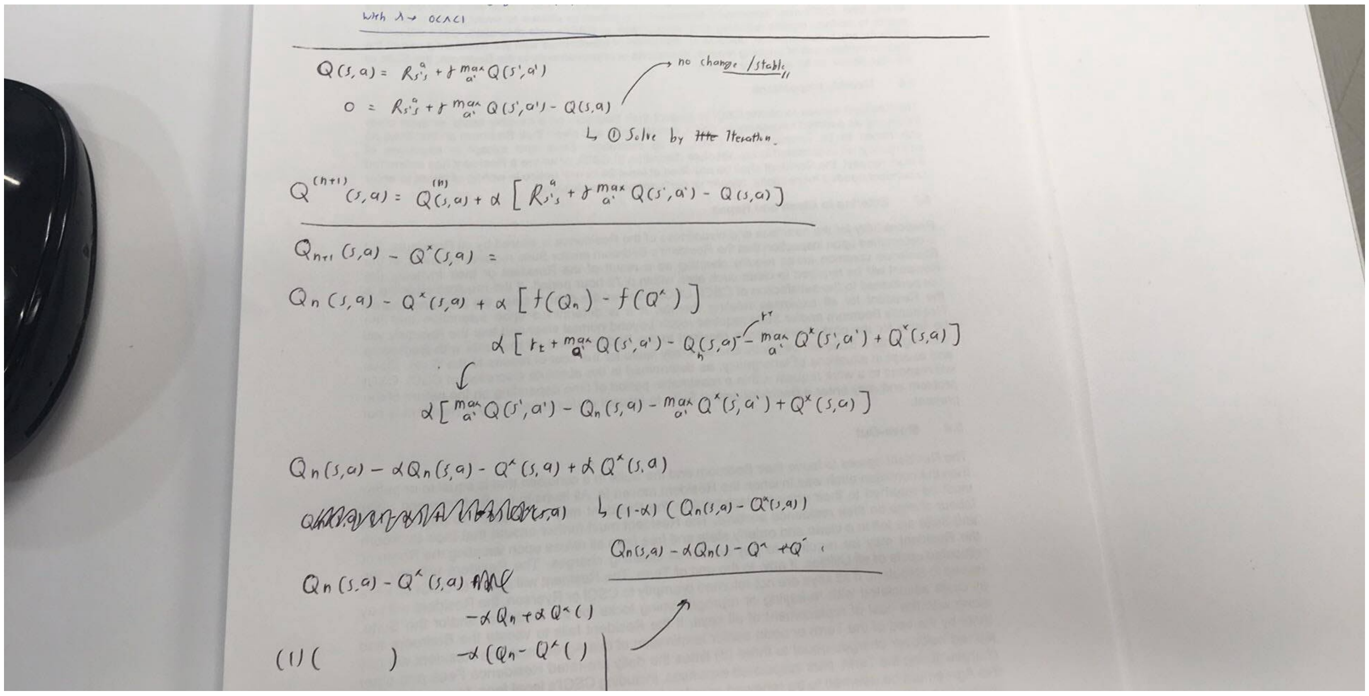
Taking into account that for the optimal policy Q^* $f(\{Q^*\}) = 0$,

$$\begin{aligned} Q_{n+1}(s,a) - Q^*(s,a) &= Q_n(s,a) - Q^*(s,a) + \alpha[f(\{Q_n\}) - f(\{Q^*\})] \\ &= (1-\alpha)[Q_n(s,a) - Q^*(s,a)] \\ &\quad + \alpha\gamma[\max_{a'} Q(s',a') - \max_{a'} Q^*(s',a')] \end{aligned}$$

or

$$\begin{aligned} |Q_{n+1}(s,a) - Q^*(s,a)| &\leqslant |1-\alpha||Q_n(s,a) - Q^*(s,a)| \\ &\quad + |\alpha|\gamma|\max_{a'} Q(s',a') - \max_{a'} Q^*(s',a')|. \end{aligned}$$

And we can put an upper bound of the convergence.



with $\lambda \to 0<\lambda<1$

$Q(s,a) = R^a_{s's} + \gamma \max_{a'} Q(s',a')$ → no change / stable

$0 = R^a_{s's} + \gamma \max_{a'} Q(s',a') - Q(s,a)$

↳ ① Solve by ~~the~~ Iteration.

$Q^{(n+1)}(s,a) = Q^{(n)}(s,a) + \alpha [R^a_{s's} + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

$Q_{n+1}(s,a) - Q^*(s,a) =$

$Q_n(s,a) - Q^*(s,a) + \alpha [f(Q_n) - f(Q^*)]$

$\alpha [r_t + \max_{a'} Q(s',a') - Q_n(s,a) - \overset{r_t}{} \max_{a'} Q^*(s',a') + Q^*(s,a)]$

$\alpha [\max_{a'} Q(s',a') - Q_n(s,a) - \max_{a'} Q^*(s',a') + Q^*(s,a)]$

$Q_n(s,a) - \alpha Q_n(s,a) - Q^*(s,a) + \alpha Q^*(s,a)$

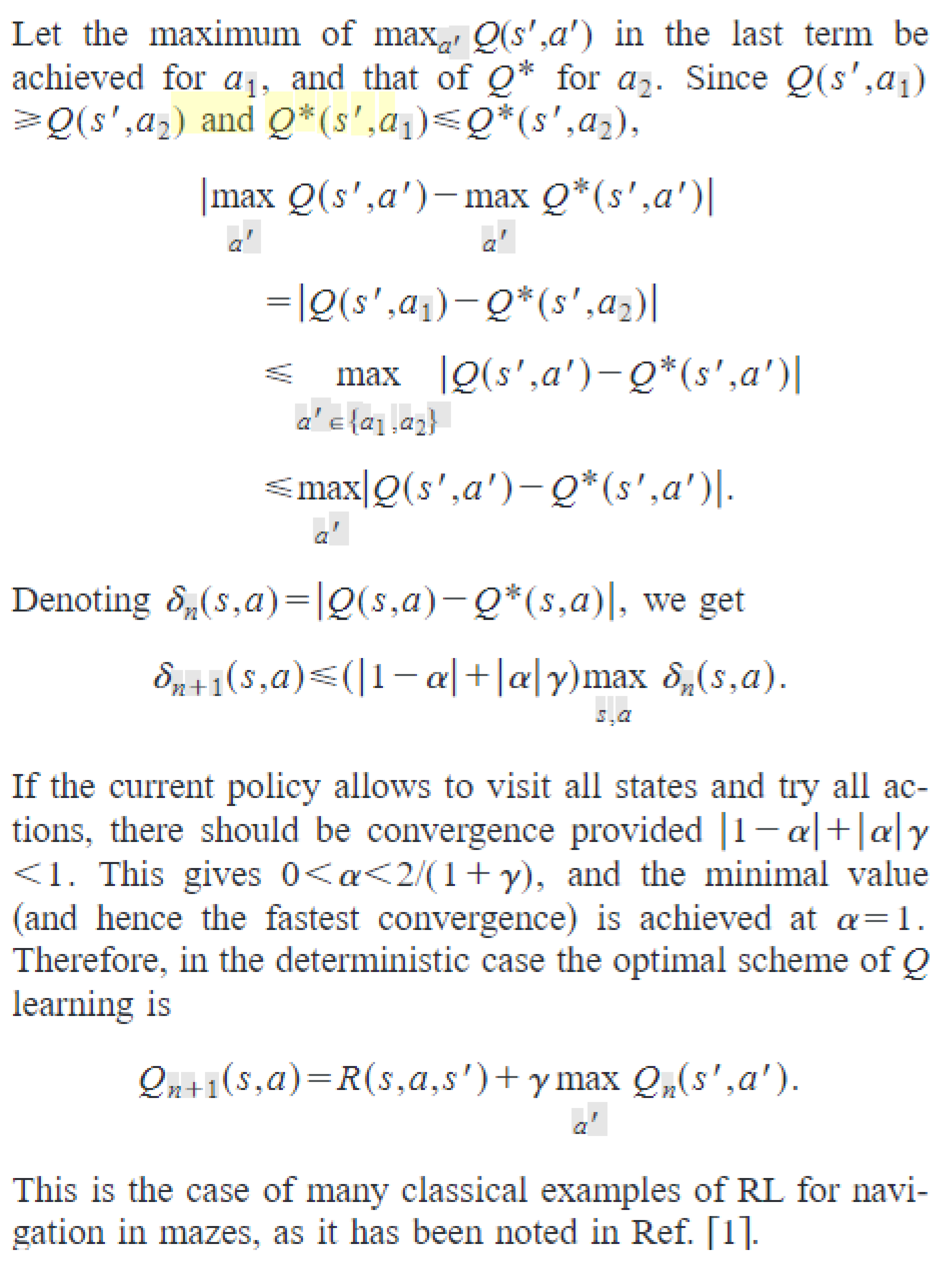
[illegible] ↳ $(1-\alpha)(Q_n(s,a) - Q^*(s,a))$

$Q_n(s,a) - \alpha Q_n() - Q^* + Q^*$

$Q_n(s,a) - Q^*(s,a)$ [illegible]

$-\alpha Q_n + \alpha Q^*()$

$(1)(\quad)$ $-\alpha(Q_n - Q^*())$



Let the maximum of $\max_{a'} Q(s',a')$ in the last term be achieved for a_1, and that of Q^* for a_2. Since $Q(s',a_1) \geqslant Q(s',a_2)$ and $Q^*(s',a_1) \leqslant Q^*(s',a_2)$,

$$\begin{aligned}|\max_{a'} Q(s',a') - \max_{a'} Q^*(s',a')| \\ &= |Q(s',a_1) - Q^*(s',a_2)| \\ &\leqslant \max_{a' \in \{a_1,a_2\}} |Q(s',a') - Q^*(s',a')| \\ &\leqslant \max_{a'} |Q(s',a') - Q^*(s',a')|.\end{aligned}$$

Denoting $\delta_n(s,a) = |Q(s,a) - Q^*(s,a)|$, we get

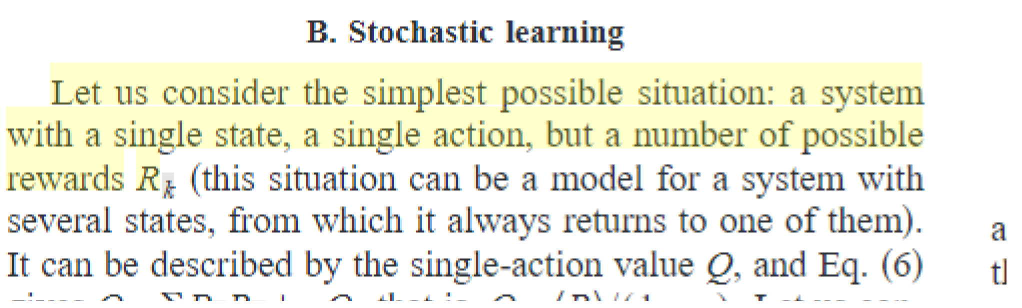
$$\delta_{n+1}(s,a) \leqslant (|1-\alpha| + |\alpha|\gamma) \max_{s,a} \delta_n(s,a).$$

If the current policy allows to visit all states and try all actions, there should be convergence provided $|1-\alpha| + |\alpha|\gamma < 1$. This gives $0 < \alpha < 2/(1+\gamma)$, and the minimal value (and hence the fastest convergence) is achieved at $\alpha = 1$. Therefore, in the deterministic case the optimal scheme of Q learning is

$$Q_{n+1}(s,a) = R(s,a,s') + \gamma \max_{a'} Q_n(s',a').$$

This is the case of many classical examples of RL for navigation in mazes, as it has been noted in Ref. [1].

And it turns out that the alpha value is one → the most optimal → hence the rule can be written like above. (where we started the whole thing).

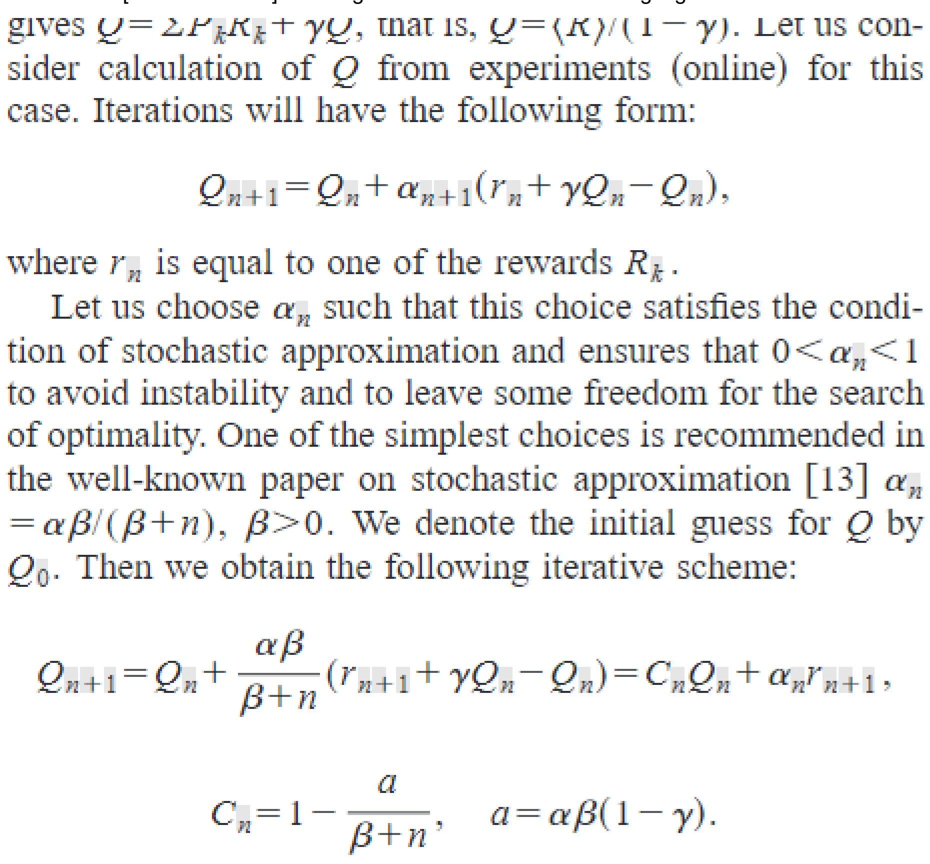


B. Stochastic learning

Let us consider the simplest possible situation: a system with a single state, a single action, but a number of possible rewards R_k (this situation can be a model for a system with several states, from which it always returns to one of them). It can be described by the single-action value Q, and Eq. (6)

a

tl



gives $Q=\sum P_k R_k+\gamma Q$, that is, $Q=\langle R\rangle/(1-\gamma)$. Let us consider calculation of Q from experiments (online) for this case. Iterations will have the following form:

$$Q_{n+1}=Q_n+\alpha_{n+1}(r_n+\gamma Q_n-Q_n),$$

where r_n is equal to one of the rewards R_k.

Let us choose α_n such that this choice satisfies the condition of stochastic approximation and ensures that $0<\alpha_n<1$ to avoid instability and to leave some freedom for the search of optimality. One of the simplest choices is recommended in the well-known paper on stochastic approximation [13] $\alpha_n=\alpha\beta/(\beta+n)$, $\beta>0$. We denote the initial guess for Q by Q_0. Then we obtain the following iterative scheme:

$$Q_{n+1}=Q_n+\frac{\alpha\beta}{\beta+n}(r_{n+1}+\gamma Q_n-Q_n)=C_nQ_n+\alpha_n r_{n+1},$$

$$C_n=1-\frac{a}{\beta+n},\quad a=\alpha\beta(1-\gamma).$$

Next, the authors prove → for the stochastic case as well.

CONVERGENCE OF REINFORCEMENT LEARNING . . . PHYSICAL REVIEW E **67**, 026706 (2003)

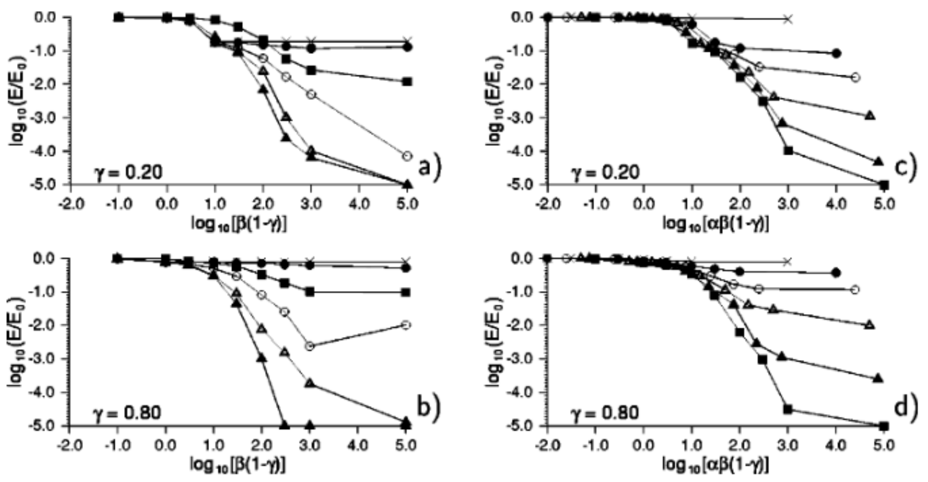


FIG. 1. Convergence of Q learning during 1000 iterations for deterministic process with $n_s=6$ states and $n_a=3$ actions. Panels (a),(b): $\alpha=1.0$, $\gamma=0.2$ and 0.8, exploration $\epsilon=0$ (×), 0.01 (●), 0.1 (○), 0.2 (△), 0.5 (filled triangle), 0.9 (■); panels (c),(d): $\epsilon=0.2$, $\gamma=0.2$, and 0.8, $\alpha=0.01$ (×), 0.1 (●), 0.25 (○), 0.5 (△), 0.75 (filled triangle), 1.0 (■). Panels (a),(b) show that the best convergence is achieved for $\epsilon=0.2$ and 0.5. Panels (c),(d) show that the convergence improves as α goes to 1 and β goes to infinity, which agrees with theoretical analysis of deterministic learning. Navigation in mazes fits into this category.

variance decreases as $\ln n/n$.

Now there are three different cases.

(3) $2a-2>-1$, $a>1/2$, $\beta>1/[2\alpha(1-\gamma)]$,

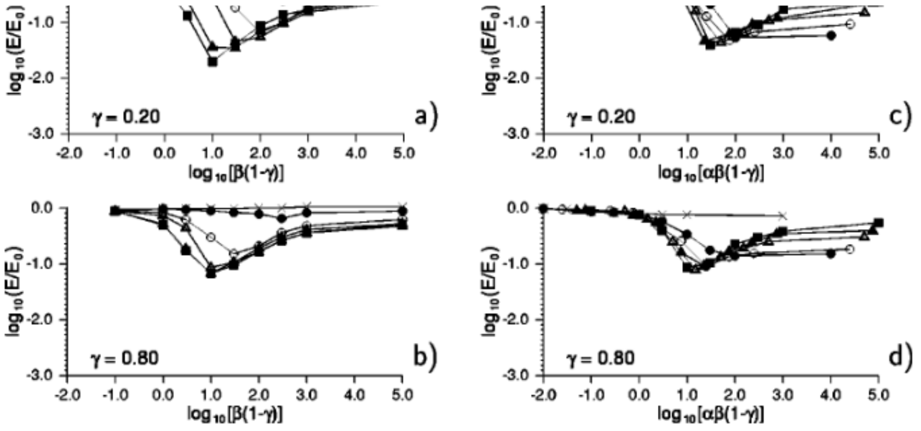


FIG. 2. Like Fig. 1, filled random process with $n_s = 4$ states and $n_a = 2$ actions. Panels (a),(b) show that learning is faster for higher exploration level. Panels (c),(d) show that the optimal convergence corresponds to $\alpha = 1$ and $\alpha\beta(1-\gamma)$ close to 10, that is, $n_0 \approx 10$. Increase of γ [compare panels (a),(b) and (c),(d)] slows the convergence.

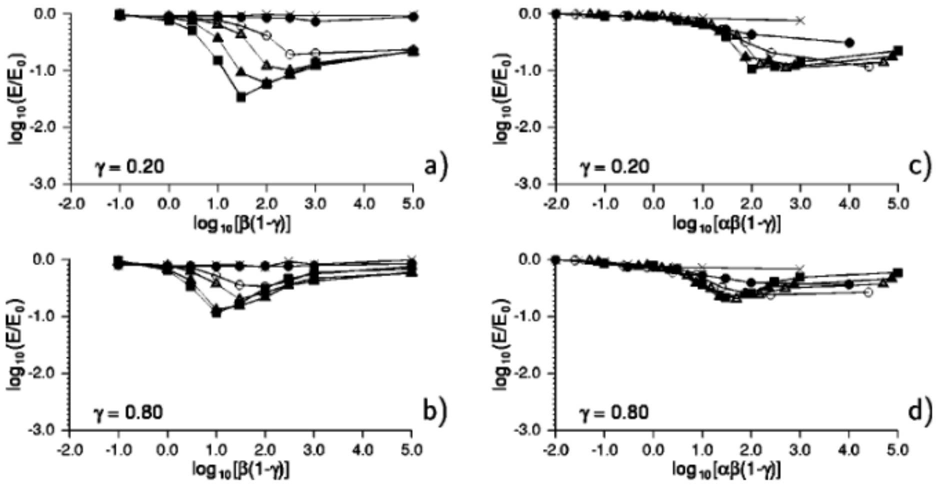
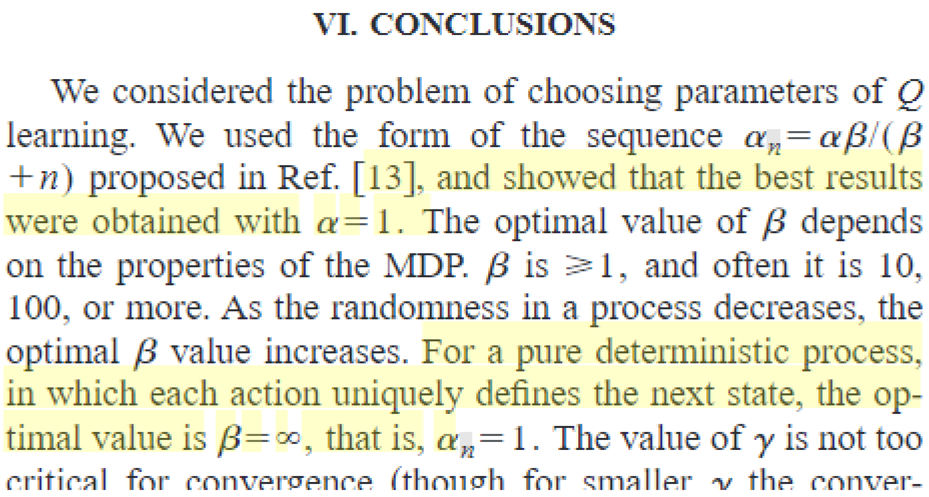


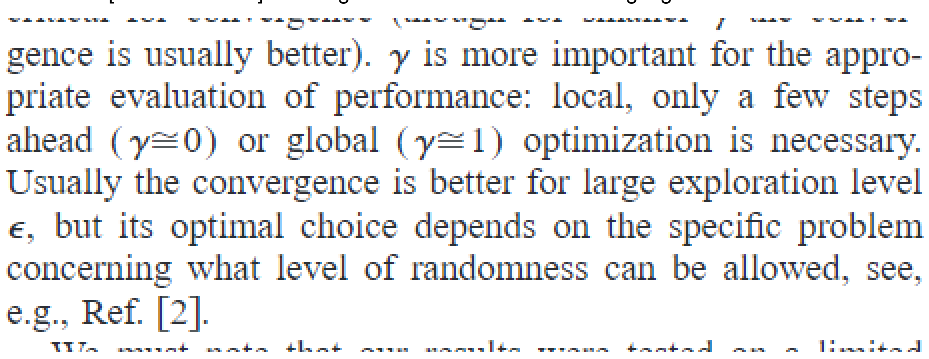
FIG. 3. Like Fig. 1, filled random process with $n_s = 4$ states and $n_a = 4$ actions. Panels (a),(b) show that learning is faster for higher exploration level. Panels (c),(d) show that the optimal convergence corresponds to $\alpha = 1$ and $\alpha\beta(1-\gamma) = n_0$ close to 10^2. For greater γ convergence is slower.

Finally, the authors performed numerical experiments → that align with the theoretical findings. (this is very important → that does FOLLOW THE theoretical works). (greater discount rate → convergence might be slower). (for deterministic cases → convergence is much faster). (RL method → very good method for solution finding → not exactly known why this happens).



VI. CONCLUSIONS

We considered the problem of choosing parameters of Q learning. We used the form of the sequence $\alpha_n = \alpha\beta/(\beta + n)$ proposed in Ref. [13], and showed that the best results were obtained with $\alpha = 1$. The optimal value of β depends on the properties of the MDP. β is $\geqslant 1$, and often it is 10, 100, or more. As the randomness in a process decreases, the optimal β value increases. For a pure deterministic process, in which each action uniquely defines the next state, the optimal value is $\beta = \infty$, that is, $\alpha_n = 1$. The value of γ is not too critical for convergence (though for smaller γ the conver-


gence is usually better). γ is more important for the appropriate evaluation of performance: local, only a few steps ahead ($\gamma \cong 0$) or global ($\gamma \cong 1$) optimization is necessary. Usually the convergence is better for large exploration level ϵ, but its optimal choice depends on the specific problem concerning what level of randomness can be allowed, see, e.g., Ref. [2].

The authors → analyzed the best method for choosing hyperparameters → drove proofs for both cases. (but there are still unknown reasons).

· · ·

Reference

1. https://pdfs.semanticscholar.org/d14a/920ca3aa81d52d323d0dbb9dcd9491c2e980.pdf

Machine Learning