Лекция 22. Самообучение чрез стимули

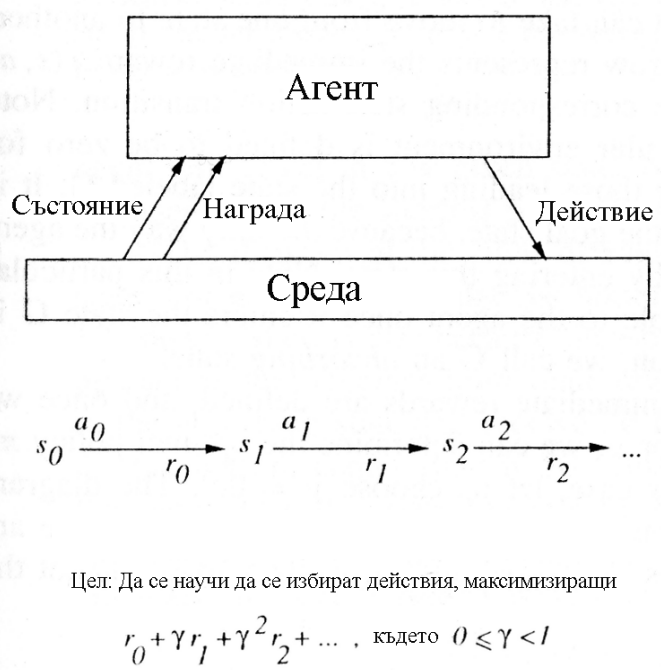
Самообучението чрез стимули (reinforcement learning) адресира въпроса как един автономен агент, който чувства и действа в своята среда, може да научи да избира оптималните действия за постигане на своите цели. Този много общ проблем покрива такива задачи като научаване да управляваш мобилен робот, научаване да оптимизираш операции в фабрики, научаване да играеш различни игри и т.н. Всеки път, когато агентът изпълнява някакво действие в своята среда, учителят може да го награди или да накаже, за да посочи желателност (привлекателност) на полученото състояние. Например, когато учителят (треньорът) учи един агент да играе някаква игра, той може да му даде някаква положителна награда, когато играта е спечелена, някаква отрицателна награда (наказание), когато тя е загубена или никаква награда във всички останали състояния. Задачата на агента е да се научи от тази непряка, закъсняла награда, за да избере последователността от действия, които произвеждат най-голямата кумулативна (с натрупване) награда. В тази лекция ще се фокусираме върху алгоритъм наречен Q-самообучение, който може да научи оптималните стратегии за управление от закъснели награди, дори когато агентът няма никакви предварителни знания за ефекти от своите действия в средата. Алгоритмите за самообучение чрез стимули са свързани с алгоритми на динамично програмиране, които често се използват за решаване на оптимизационни задачи.

22.1. Въведение

Да разгледаме построяване на един самообучаващ се робот. Роботът или агентът разполага с набор от сензори, за да наблюдава *състоянието (state)* на своята среда, както и с един набор от *действия (actions)*, които той може да извършва, за да промени това състояние. Например един мобилен робот може да има такива сензори, като камера и сонари, и да изпълнява такива действия, като „движение напред" и „обръщане назад". Задачата му е да научи някоя стратегия или *политика (policy)* за избор на действия, която води до постигане на неговите цели. Например роботът може да има за цел да отиде към устройство за зареждане на батерии, когато нивото на неговата батерия стане низко.

В тази лекция ще се концентрираме върху това, как подобни агенти могат да научат успешни политики за управление чрез провеждане на експерименти в окръжаващата ги среда. Ще подразбираме, че целите на агента могат да бъдат дефинирани чрез някаква *функция за награда* (reward function), която назначава определена числова стойност – незабавно възнаграждение – за всяко отделно действие, което агентът извърши във всяко отделно състояние. Например, целта „отиване към устройство за зареждане на батерии" може да бъде свързана чрез някоя положителна награда (например + 100) към такъв преход „цел-действие", който веднага води до подсъединяване към зареждащото устройство, и чрез нулева награда - към всеки останал преход от типа „цел-действие". Тази функция на наградата може да бъде вградена в робота или да бъде известна само на някой външен учител, който предоставя конкретна наградата за всяко действие, изпълнено от робота. Задачата на робота е да изпълнява последователности от действия, да наблюдава последствия от тези действия и да научи някоя политика за управление на тези действия. Политиката за управление, която ние искаме да бъзе научена, е тази, която за всяко начално състояние избира действия, максимизиращи наградата,

акумулирана във времето от агента. Тази обща постановка на задачата за самообучение на робота е резюмирана във Фиг. 22-1.



Фиг. 22-1. *Един агент, взаимодействащ със своята среда.* Агентът съществува в една среда, описвана чрез някое множество от възможни състояния S, може да изпълнява всяко едно от множество възможни действия A. Всеки път, когато той изпълнява някое действие a_t в състояние s_t, агентът получава определена награда (реална величина) r_t, която указва незабавната „цена" на този преход „състояние-действие". Този процес произвежда последователност от състояния s_i, действия a_i и незабавни награди r_i, както е показано на фигурата. Задачата на агента е да научи такава управляваща политика $\pi: S \rightarrow A$, която максимизира очакваната сума от тези награди плюс бъдещи награди, намалени експоненциално в съответствие с тяхната отдалеченост във времето.

Както се вижда от Фиг. 22-1, задачата за научаване на някоя политика за управление с цел максимизиране на кумулативната награда е много обща и покрива множество други задачи извън задачата за самообучение на роботи. В общия случай задачата принадлежи към класа задачи за научаване на политики за управление на последователни процеси, които включват, например, производствени оптимизационни задачи, в които една последователност от производствени действия трябва да бъде избрана и наградата, която трябва да бъде максимизирана, е стойността на произведените стоки минус производствените разходи. Към този клас се отнасят и задачи за последователни разписания, като, например, избор кои таксита трябва да бъдат изпратени на пасажери в един голям град В тези задачи наградата, подлежаща на максимизиране, е една функция от времето на изчакване от страна на пасажерите и общата цена на гориво за таксиметрови услуги. В общия случай ние се интересуваме от всякакви агенти, които трябва да се самообучават, за да избират действия, които променят състояние на своята среда, и които използват някаква кумулативна награда, за да дефинират качеството на произволна последователност от действия. Вътре в този клас задачи ще разгледаме различни специфични предположения, включително и такива, при които действията имат детерминистични или недетерминистични последствия, както и такива, при които агентът има или няма предварителни знания за ефекта от своите действия върху средата.

Задачата за научаване на някоя управляваща политика за избиране на действия е подобна, в някои отношения, на задачи за апроксимация на функции, които сме разглеждали в другите лекции. Целевата функция, която трябва да бъде научена в този случай, е една политика за управление $\pi: S \rightarrow A$, такава че извежда едно подходящо действие a от множеството A при зададено текущо състояние s от множеството S. Обаче таза задача за самообучение чрез стимули се отличава от другите задачи по апроксимация на функции по следните важни аспекти:

- *Отложена награда*. Задачата на агента е да научи целевата функция π, която отобразява текущото състояние s в оптималното действие $a = \pi(s)$. По-рано ние винаги смятахме, че при научаване на някоя целева функция от типа на π всеки обучаващ пример се задава във вида на двойка $<s, \pi(s)>$. При самообучение чрез стимули обучаващата информация в този вид *не е налична*. Вместо това учителят предоставя само една последователност от стойностите на незабавна награда, когато агентът изпълнява своята последователност от действия. По този начин агентът се изправя срещу задачата за *временно гласуване на доверие (temporal credit assignment),* определяща на кои действия от тази последователност трябва да бъде гласувано доверие чрез произвеждане на евентуалните награди.
- *Проучване.* При самообучение чрез стимули агентът влияе на разпределението на обучаващи примери чрез последователността от действия, избрана от него. Това повдига въпроса, коя стратегия за експериментиране води до най-ефективното самообучение. Системата за самообучение се изправя срещу един компромис в избора дали е изгодно *проучване* на неизвестни състояния и действия (за да събере нова информация) или *използване* на състояния и действия, за които той вече научи, че водят до голяма награда (за да максимизира своята кумулативна награда).
- *Частично наблюдаеми състояния.* Макар да е удобно да предполагаме, че сензорите на агента могат да долавят пълното състояние на средата във всеки момент от времето, в много практически ситуации сензорите предоставят само частична информация. Например един робот с камерата, сочеща напред, не може да види, какво става зад него. В такива случаи за да избере своите действия, заедно с текущите данни от сензори агентът трябва да може да преглежда и своите наблюдения, направени по-рано, а най-добрата политика може да е тази, която избира действия, насочени специално към подобряване на обозримостта на средата.
- *Самообучение през цял живот (life-long learning)*. В отличие от изолираните задачи по апроксимиране на функции, самообучение на роботи често изисква от робота той да научава няколко свързани задачи в една и съща среда, използвайки едни и същи сензори. Например един мобилен робот може да има необходимост да научи, как да се подключи към своето зарядно устройство, как да лавира през тесни коридори и как да взема готови листа хартия от лазерни принтери. Тази постановка на задачата увеличава възможността от използване на придобития по-рано опит или знания, за да намали сложността на научаване на нови задачи.

22.2. Задачата за самообучение

Да разгледаме задачата за научаване на последователните стратегии за управление по-детайлно. Трябва да се отбележи, че тя може да бъде формулирана по няколко различни начина. Например можем да предполагаме, че действията на агента са детерминистични, или че те са недетерминистични. Можем да предполагаме, че агентът може да предвиди следващото състояние, което ще е резултатът от всяко едно действие, или че той не може да направи това. Можем да предполагаме, че агентът се обучава от експерта, който му показва примери на оптимални последователности от действия, или че той трябва да се учи сам чрез изпълнение на действие по негов собствен избор. По-долу ще формулираме една доста обща формулировка на задачата, която се базира върху Марковските процеси за вземане на решения. Тази формулировка следва задачата, илюстрирана на Фиг. 22-1.

В един Марковски процес за вземане на решения (МПВР) агентът може да възприема множество S от различни състояния на своята среда и разполага с множество A от действия, които той може да изпълни. Във всеки дискретен момент от време t агентът „усеща" текущото състояние s_t, избира едно текущо действия a_t и го изпълнява. Средата отговаря чрез предоставянето на агента на определена награда $r_t = r(s_t, a_t)$ и чрез преход в следващото състояние $s_{t+1} = \delta(s_t, a_t)$. Функциите δ и r са част от средата и не е задължително да са известни на агента. В един МПВР функциите $\delta(s_t, a_t)$ и $r(s_t, a_t)$ зависят само от текущото състояние и действие и не зависят от предишните състояния или действия. В тази лекция ще разгледаме само случая, в който множествата S и A са крайни. В общия случай δ и r могат да бъдат недетерминистични функции, но ние ще започнем с разглеждането само на детерминистичния случай.

Задачата на агента е да научи политика $\pi: S \rightarrow A$, за да избере своето следващо действие a_t на базата на текущото наблюдаемо състояние s_t, т.е. $\pi(s_t) = a_t$. Как да укажем, коя политика π бих ме желали да научи агентът? Един очевиден подход е да изискаме такава политика, която произвежда за робота възможно най-голяма кумулативна награда във време. За да формулираме това изискване по-точно, ще дефинираме кумулативната стойност $V^{\pi}(s_t)$, постигана от една произволна политика π, започвайки от едно произволно начално състояние s_t:

$$V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{t=0}^{\infty} \gamma^t r_{t+1} \qquad (22.1)$$

където последователност от награди r_{t+1} се генерира, започвайки от състояние s_t и чрез повтарящо се използване на политиката π, за да избира действия по описания по-горе начин (т.е. $a_t = \pi(s_t)$, $a_{t+1} = \pi(s_{t+1})$ и т.н.). тука $0 \leq \gamma < 1$ е една константа, която определя относителната стойност на забавената награда срещу незабавната. В частност награди, получавани след i времеви стъпки в бъдещето, са намаляват експоненциално чрез фактор γ^i. Обърнете внимание, че ако $\gamma = 0$, то ще се разглежда само незабавната награда. Ако установим стойността на γ близо до 1, то бъдещите награди ще дават по-голям принос по отношение на незабавната награда.

Величината $V^{\pi}(s_t)$, зададена чрез формула (22.1), често се нарича *намалена кумулативна награда* (discounted cumulative reward), постигната от политика π от начално състояние s. Намаляването на бъдещите награди относително незабавната награда е изглежда разумно, тъй като, в много случаи, ние предпочитаме да получим наградата по-рано, от колко по-късно. Обаче са използвани и други дефиниции на общата награда. Например

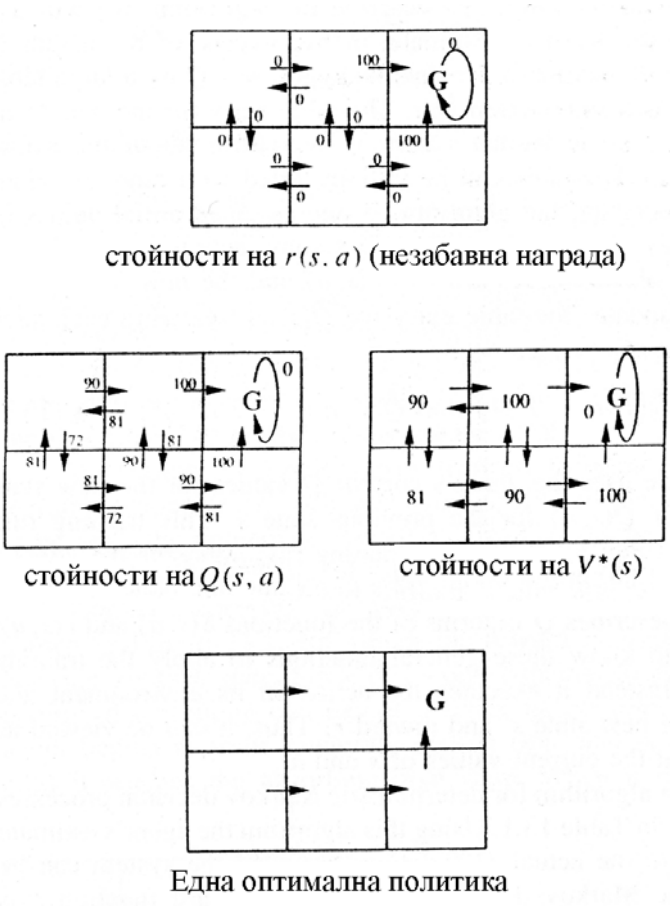
наградата на крайния хоризонт (finite horizon reward): $\sum_{i=0}^{h} r_{i+1}$ разглежда намаляващата сума от награди само в рамките на един краен брой стъпки *h*. Другата възможност е *осреднената награда* (average reward): $\lim_{h\to\infty} \frac{1}{h}\sum_{i=0}^{h} r_{i+1}$, която разглежда осредненатата награда за една стъпка върху целият живот на агента. В тази лекция ни ще се ограничим до разглеждане на намаляващата награда, както тя е дефинирана в формулата (22.1). Mahadevan (1996) разглежда подробно самообучението чрез стимули в случаи, когато критерият за оптимизиране е осредненатата награда.

Да дефинираме задача за самообучение на агента формално: ще изискваме агентът да научава политика π, която максимизира $V^{\pi}(s)$ за всички състояния s. Ще наричаме такава политика оптимална политика и я означим с π*:

$$\pi^* \equiv \arg\max_{\pi} V^{\pi}(s), (\forall s) \quad (22.2)$$

С цел облекчаване на записването, ще означаваме оценъчната функция $V^{\pi^*}(s)$ на такава оптимална политика чрез *V*(S)*. *V*(S)* дава максимум на намалената кумулативна награда, която агентът може да получи, стартирайки от състояние s, следвайки оптималната политика.

За илюстрация на тези понятия да разгледаме една проста среда във вид на решетка, показана в най-горната част на Фиг. 22-2.



Фиг. 22-2. *Един прост пример на детерминистичния свят, илюстриращ базовите понятия на Q самообучение.* Всяка клетка представлява различно състояние, всяка

стрелка – различно действие. Функцията на незабавната награда *r(s, a)* дава награда от 100 единици за всяко действие, водещо до влизане в целевото състояние *G,* и от 0 единици – за всички останали случаи. Стойностите на *V*(s)* и *Q(s, a)* следват от *r(s, a)*, а намаляващия фактор $\gamma = 0.9$. Показана е и една оптимална политика, съответстваща на действия с максимални стойности на *Q*.

Шестте квадратни клетки на решетката представят шест възможни състояния или локации на агента. Всяка стрелка представлява едно възможно действие, което агентът може да предприеме за да се придвижи от едно състояние в друго. Числото, асоциирано с всяка от стрелките, представлява незабавната награда *r(s, a)*, която агентът получава, ако изпълнява съответния преход състояние-действие. Обърнете внимание, че незабавната награда в тази конкретна среда е определена да бъде равна на нула за всички преходи състояние-действие освен тези, които водят към състояние, маркирано с ***G***. Това състояние е удобно да бъде разглеждано като целево състояние, тъй като единственият начин агентът да получи наградата си в този случай, е той да се премести в това състояние. Освен това, в тази специална среда единственото действие, достъпно на агента, когато той влезне в състояние ***G***, е да остане в това състояние. По тази причина ние наричаме състояние ***G*** - *поглъщащо (absorbing) състояние*.

След като всички състояния, действия и незабавни награди са определени и след като бъде избрана конкретна стойност за намаляващия фактор γ, вече можем да дефинираме оптималната политика π^* и нейната оценъчна функция *V*(s)*. В конкретния случай нека да изберем $\gamma = 0.9$. Диаграмата в долната част на Фиг. 22-2 показва една оптимална политика (съществуват и други) за този случай. Както и всяка друга политика, тази политика задава точно по едно действие, което агентът може да избере във всяко едно състояние. Няма нищо учудващо, че оптималната политика насочва агента по най-краткия път към състояние **G**.

Диаграмата в дясната част на фигурата показва стойностите на *V** за всяко състояние. Например да разгледаме състояние в най-долната дясна част на решетката на тази диаграма. Стойността на *V** за това състояние е равна на 100, тъй като оптималната политика в това състояние избира действие "премести се нагоре", което получава незабавната награда 100. От там нататък агентът ще остане в поглъщащото състояние и няма да получи никакви други награди. По същия начин стойността на *V** на състояние, разположено долу в центъра, е 90, тъй като оптималната политика ще премести агента от това състояние вдясно (генерирайки незабавната награда равна на нула), а след това нагоре (генерирайки незабавната награда равна 100). Следователно намалената бъдеща (кумулативна) награда от състояние долу в центъра е $0 + 0.9*100 + 0.9^2*0 + 0.9^3*0 + \ldots = 90$. Напомням, че *V** е определена да бъде сума от намалени бъдещи награди в неограничено бъдеще. В нашата конкретна среда след като агентът веднъж е достигна поглъщащото състояние **G,** неговото неограничено бъдеще се състои в оставане в това състояние и получаване на нулевата награда.

22.3. Q самообучение

Как един агент може да научи някаква оптимална политика π^* в произволна среда? Директното научаване на функцията $\pi^*: S \rightarrow A$ е сложно, тъй като наличните обучаващи данни не предоставят обучаващи примери във вида $<s, a>$. Вместо това единствената налична обучаваща информация за системата за самообучение е

последователността от незабавни награди $r(s_i, a_i)$, $i = 0, 1, 2, \ldots$ Както ще видим по-нататък, при наличието на този вид обучаващата информация е по-лесно отначало да бъде научена една числова оценъчна функция, дефинирана върху състояния и действия, а след това да бъде имплементирана оптималната политика в термини на тази функция.

Каква оценъчна функция трябва да опита да научи агентът? Един очевиден избор е V^*. Агентът трябва да предпочита състояние s_1 пред s_2, когато $V^*(s_1) > V^*(s_2)$, тъй като кумулативната бъдеща награда ще бъде по-голяма в s_1. Ясно е, че политиката на агента трябва да избира между действия, а не между състояния. Обаче той може да използва V^* при определени условия, за да избира и между действия. Оптималното действие в състояние s е такова действие a, което максимизира сумата от незабавната награда $r(s, a)$ плюс стойността на V^* в състояние, което е непосредствен наследник на s, намалена с фактора γ (напомням, (е чрез $\delta(s, a)$ означаваме състояние, получено след прилагането на действие a към състояние s):

$$\pi^*(s) \equiv \arg\max_a [r(s,a) + \gamma V^{\pi}(\delta(s,a))] \quad (22.3)$$

Следователно агентът може да постигне оптималната политика чрез научаване на V^* *при условие, че той има точни знания за функцията на незабавната награда* r *и функцията на прехода от състояние в състояние* δ. Когато агентът знае функциите r и δ, използвани от средата за отговор на неговите действия, той може да използва формула (22.3), за да изчисли оптималното действие за всяко състояние s.

За съжаление научаването на V^* представлява един плодотворен начин за научаване на оптималната политика *само*, когато агентът има абсолютно точни знания за δ и r. Това изисква той да е в състояние *абсолютно точно* да предвижда непосредствения резултат (т.е. незабавната награда и следващото състояние) за всеки възможен преход действие-състояние. За много практически задачи, като например управление на роботи, е невъзможно агентът или човекът-програмист да предвиди предварително точния резултат от прилагане на едно произволно действие в едно произволно състояние. Представете си, например, трудността на описание на δ за една роботизирана ръка, която насипва пръст, когато резултиращото състояние включва позиции на частиците от пръстта. В случаи, когато или δ, или r са неизвестни, научаването на V^*, за съжаление, не помага за избор на оптималните действия, тъй като агентът не може да оцени уравнение (22.3).

22.3.1. Q функцията

Нека да дефинираме оценъчната функция $Q(s,a)$ така, че нейната стойност представлява максималната намалена кумулативна награда, която може да бъде достигната от състояние s след прилагане на действие a като първото действие. С други думи стойността на Q е наградата, получена *веднага* след изпълнение на действие a от състояние s плюс стойността (намалена с γ) на последващата оптимална политика:

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a)) \quad (22.4)$$

Обърнете внимание, че $Q(s, a)$ е абсолютно същия израз, който се максимизира в формула (22.3), за да бъде избрано оптималното действие a в състояние s. Следователно ние можем да препишем формула (22.3) в термините на $Q(s, a)$:

$$\pi^*(s) \equiv \arg\max_a Q(s,a) \quad (22.5)$$

Защо този начин на преписване е важен? Защото той показва, че ако агентът научава функцията Q вместо функцията V^*, той ще бъде в състояние да избира оптимални действия *дори когато той няма точни знания за функциите* δ *и* r. Както показва формулата (22.5), агентът трябва да разгледа всяко налично действие a в текущо състояние, в което той се намира, и да избере действието, което максимизира $Q(s, a)$.

На пръв поглед изглежда доста учудващо, че някой може да избира една глобално оптимална последователност от действия чрез повтарящ избор на локално оптимални стойности на Q в текущо състояние. Това означава, че агентът може да избира оптималното действие без никога да не извършва едно предсказващо търсене, целящо точното определяне, до какво състояние ще доведе изпълнение на конкретното действие. Част от красотата на Q самообучение се състои в това, че оценъчната функция е определена да има точно това свойство – стойността на Q за текущото състояние и действие обобщава в едно единствено число цялата информация, необходима, за да се определи намалената кумулативна награда, която ще бъде спечелена в бъдещето, ако действие a е избрано в състояние s.

Като илюстрация Фиг. 22-2 показва стойностите на Q за всяко състояние и действие в един прост „решетъчен" свят. Обърнете внимание, че стойността на Q за всеки преход състояние-действие равна на стойността на r за този преход плюс стойността на V^* за резултиращото състояние, намалена с γ. Обърнете също внимание, че оптималната политика, показана но фигурата, съответства на избора на действия с максималните стойности на Q.

22.3.2. Алгоритъм за Q самообучение

Научаването на функцията Q съответства на научаване на оптималната политика. Как Q може да бъде научена?

Основният проблем е в намиране на надеждния начин за оценяване на обучаващи стойности на Q, когато в наличност само някоя последователност от незабавни награди r разпределени във времето. Това може да бъде извършено чрез итеративната апроксимация. За да видим, как става това, ще напомня за тясната връзка между Q и V^*:

$$V^*(s) = \max_{a'} Q(s,a')$$

която позволява преписване на уравнение (22.4) във вида:

$$Q(s,a) = r(s,a) + \lambda \max_{a'} Q(\delta(s,a),a') \quad (22.6)$$

Тази рекурсивна дефиниция на Q предоставя основа за алгоритми, които итеративно апроксимират Q (Watkins 1989). За да опишем алгоритъма, ще използваме символ $\hat{Q}$, за да означим оценката, направена от системата за самообучение, или хипотеза за действителния вид на функцията Q. В този алгоритъм системата за самообучение представлява своята хипотеза $\hat{Q}$ чрез една голяма таблица, в която всяка двойка състояние-действие е представена чрез отделен елемент. Елементът на таблица, представляващ двойката $<s, a>$ съхранява стойността на $\hat{Q}(s,a)$ - текущата хипотеза на

системата за действителна, но неизвестна стойност на *Q(s, a)*. Таблицата може в начало да бъде попълнена със случайни стойности (макар че е по-лесно да се разбере алгоритъма, ако се предполага, че началните стойности са нулеви). Агентът циклично преглежда своето текущо състояние *s*, избира някакво действие *a*, изпълнява го и след това преглежда получената като резултат награда *r = r(s,a)* и новото състояние *s' = δ(s, a)*. След това той обновява елементът от таблицата за $\hat{Q}(s,a)$, следвайки всеки такъв преход в съответствие със следното обучаващо правило:

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \quad (22.7)$$

Обърнете внимание, че това обучаващо правило използва текущите стойности на агента $\hat{Q}$ за новото състояние s', за да уточни своята оценка $\hat{Q}(s,a)$ за предишното състояние *s*. Правилото е мотивирано от уравнение (22.6), макар че то засяга приближението, направено от агента $\hat{Q}$, докато уравнението (22.6) се прилага към действителната функция *Q*. Макар че уравнението (22.6) описва *Q* в термини на функциите *δ(s, a)* и *r(s, a)*, агентът не се нуждае от познаването на тези общи функции, за да прилага обучаващото правило (22.7). Вместо това той изпълнява действието в своята среда и след това наблюдава резултиращото ново състояние *s'* и наградата *r*. По този начин той може да бъде разгледан като правещ анализ на тези функции в текущите стойности на *s* и *a*.

Описаният по-горе алгоритъм за Q самообучение, приложен към детерминистичен Марковски процес за вземане на решение, е подробно представен в Таблица 22-1. Използвайки този алгоритъм оценката на агента $\hat{Q}$ се схожда в предела към действителната функция *Q*, при условие, че системата може да бъде моделирана като един Марковски процес за вземане на решения, че функцията на награда *r* е ограничена, и че действията се избират по такъв начин, че всяка двойка състояние-действие се посещава неограничено често.

Алгоритъм за Q самообучение

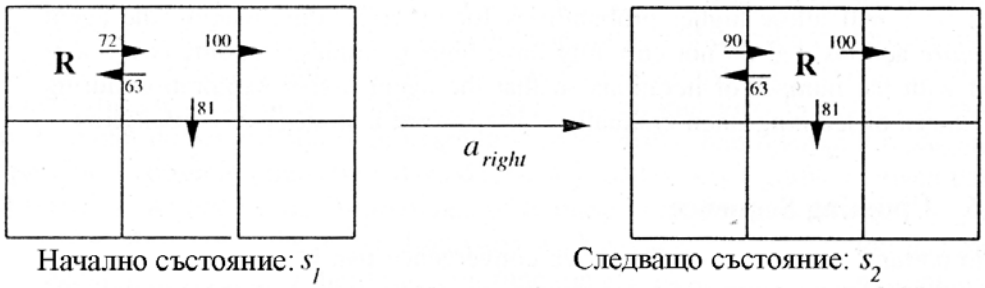
За всяко *s, a* инициализирай елементът от таблицата $\hat{Q}(s,a)$ с нула
Прегледай текущото състояние s
Прави до безкрай:

- Избери някое действие *a* и го изпълни
- Получи незабавната награда *r*
- Прегледай новото състояние s'
- Обнови елементът от таблицата $\hat{Q}(s,a)$ по следния начин:
  $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- s ← s'

Таблица 1. Алгоритъм за Q самообучение, подразбиращ детерминистични награди и действия. Намаляващият фактор γ може да е всякаква константа, такава че $0 \leq \gamma < 1$.

22.3.3. Илюстративен пример

За да илюстрираме работата на алгоритъма за Q самообучение, да разгледаме едно единствено действие, предприето от агента, и съответното уточняване на $\hat{Q}$ (Фиг. 22-3). В този пример агентът се предвижва една клетка надясно в своят „решетъчен" свят и получава незабавната награда, равна на нула, за този преход. След това той прилага обучаващото правило (22.7), за да уточни своята оценка $\hat{Q}$ за извършения преход състояние-действие. В съответствие с обучаващото правило, новата оценка $\hat{Q}$ за този преход е равна на сумата от получената награда (нула) и най-голямата стойност на $\hat{Q}$, асоциирана с резултиращото състояние (100), намалена с фактора γ (0.9).



$$\hat{Q}(s_1, a_{right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a')$$
$$\leftarrow 0 + 0.9 \max\{63, 81, 100\}$$
$$\leftarrow 90$$

Фиг. 22-3. *Обновяване на $\hat{Q}$ след изпълнение на едно единствено действие*. Диаграмата вляво показва началното състояние s_t на робота (**R**) и няколко релевантни стойности на $\hat{Q}$, съответстващи на неговите начални хипотези. Например стойността $\hat{Q}(s_1, a_{right}) = 72$, където a_{right} означава действие, предвижващо робота вдясно. Когато роботът изпълнява действието a_{right}, той получава незабавната награда $r = 0$ и преминава в състояние s_2. След това той обновява оценката $\hat{Q}(s_1, a_{right})$ на базата на своите оценки $\hat{Q}$ за новото състояние s_2. В примера $\gamma = 0.9$.

Всеки път, когато агентът се предвижва от едно старо състояние в някое ново състояние, Q самообучение разпространява оценките за $\hat{Q}$ *обратно* от новото състояние към старото. В същото време незабавната награда, получена от агента за прехода, се използва за увеличаване на тези разпространени стойности на $\hat{Q}$.

Да разгледаме прилагането на този алгоритъм към решетъчния свят и функцията на награда, показани на Фиг. 22-2, за която наградата е навсякъде нула, освен когато се влиза в целевото състояние. Тъй като този свят съдържа едно поглъщащо целево състояние, ще подразбираме, че обучението се състои от серия на *епизоди*. По време на всеки епизод агентът започва от някое случайно избрано състояние и на него се позволява да изпълни действия, докато той не стигне до поглъщащото целево състояние. Когато това се случи, епизодът се завършва и агентът се поставя в някое ново, пак случайно избрано състояние, за да започне нов епизод.

Как ще се променят стойностите на $\hat{Q}$, когато алгоритмът за Q самообучение се приложи към този случай? Тъй като всички стойности на $\hat{Q}$ са инициализирани с нула, агентът няма да прави никакви промени за никакви елементи от таблицата с $\hat{Q}$, докато не му се случи да достигне целевото състояние и да получи една ненулева награда. Това ще доведе до уточняването на стойността на $\hat{Q}$ за един единствен преход, водещ към целевото състояние. В следващия епизод, ако агентът премине през това съседно с целевото състояние, неговата ненулева стойност на $\hat{Q}$ ще позволи да бъде уточнена стойността на прехода, намиращ се на две стъпки от целта, и т.н. При наличие на достатъчен брой обучаващи епизоди, информацията ще се разпространява от преходи с ненулева награда обратно през цялото пространство от действия-състояния, достъпни за агента, което ще доведе, евентуално, до таблицата $\hat{Q}$, съдържаща стойностите на Q, показани на Фиг. 22-2.

В следващия раздел ще докажем, че при определени предположения алгоритмът за Q самообучение от Таблица 1 ще има сходимост към правилната функция Q. Първо ще разгледаме два общи свойства на този алгоритъм, които са изпълнени за всеки детерминистичен МПВР, в който наградите са неотрицателни, предполагайки че всички стойности на $\hat{Q}$ са инициализирани с нула. Първото свойство е, че при тези условия стойностите на $\hat{Q}$ никога не намаляват при обучение. По-формално, нека $\hat{Q}_n(s,a)$ означава научената стойност на $\hat{Q}(s,a)$ след n-та итерация на обучаващата процедура (т.е. след n-ти преход състояние-действие, изпълнен от агента). Тогава

$$(\forall s,a,n) \quad \hat{Q}_{n+1}(s,a) \geq \hat{Q}_n(s,a)$$

Второто общо свойство, което е в сила при същите предположения, е че по време на процеса по обучение всяка стойност на $\hat{Q}$ ще остане в интервал между нула и нейната истинска стойност Q:

$$(\forall s,a,n) \quad 0 \leq \hat{Q}_n(s,a) \leq Q(s,a)$$

22.3.4. Сходимост

Дали алгоритмът от Таблица 1 ще има сходимост към $\hat{Q}$, което е равно на истинската функция Q? Отговорът е "да", но при определени условия. Първо трябва да предположим, че системата представлява един детерминистичен МПВР. Второ трябва да предположим, че стойностите на незабавната награда са ограничени, т.е. съществува някаква положителна константа c, такава че за всички състояния s и действия a е изпълнено условие $|r(s, a)| < c$. Трето, трябва да предположим, че агентът избира действия по такъв начин, че обхожда всяка възможна двойка състояние-действие неограничено често. Под това се разбира, че ако действие a е едно разрешено действие от състояние s, то с течение на време агентът трябва да изпълнява действие a от състояние s няколкократно с ненулева честота, като дължината на последователността от неговите действия се приближава до безкрайност. Обърнете внимание, че някои от тези предположения са доста общи, а някои – силно рестриктивни. От едната страна те

описват едни по-общи условия от тези, които са илюстрирани от примера в предишния раздел, тъй като позволяват работа със среди, в които имат произволни положителни или отрицателни награди, както и със среди, в които произволен брой преходи състояние-действие могат да произвеждат ненулеви награди. От другата страна условията са също така и рестриктивни, тъй като изискват агентът да посещава всеки различен преход състояние-действие безкрайно често. Това е много силно предположение за големи (или непрекъснати!) проблемни области. Ние ще обсъдим по-силни резултати, свързани със сходимостта, по-късно. Обаче резултатът, описан в този раздел, предоставя базовата интуиция за разбиране, защо Q самообучение работи.

Ключовата идея, лежаща в основа на доказателството за сходимост, е че елементът на таблицата $\hat{Q}(s,a)$, който има най-голямата грешка, трябва да има намаление на тази грешка при своето обновяване с фактора γ. Причината за това е, че новата стойност зависи само частично от предразположени към грешки оценки $\hat{Q}$, като останалата част зависи от свободната от грешки незабавна награда *r*.

Теорема 1. Сходимостта на Q самообучение за недетерминистични Марковски процеси за вземане на решения. Да разгледаме един агентът, изпълняващ Q самообучение в един детерминистичен МПВР с ограничени награди $(\forall s,a)\,|r(s,a)|\le c$. Агентът използва обучаващото правило (22.7), инициализира своята таблица $\hat{Q}(s,a)$ с произволни крайни стойности и използва някой намаляващ фактор γ, такъв че $0 \le \gamma < 1$. Нека $\hat{Q}_n(s,a)$ означава хипотезата $\hat{Q}(s,a)$ на агента, получена след *n*-то обновяване. Ако всяка двойка състояние-действие се посещава неограничено често, то $\hat{Q}_n(s,a)$ ще има сходимост към *Q(s, a)* при $n \to \infty$ за всички *s, a*.

Доказателство. Тъй като всеки преход състояние-действие се изпълнява неограничено често, да разгледаме последователни интервали в течение на които всеки преход е бил осъществен най-малко един път. Доказателството се състои в показване, че максималната грешка върху всички елементи от таблицата $\hat{Q}$ се намалява с не по-малко от фактор γ при всеки такъв интервал. Нека $\hat{Q}_n$ е таблицата на агента с приближения на стойности на *Q* след *n* обновявания. Нека Δ_n е максималната грешка в $\hat{Q}_n$, т.е.: $\Delta_n \equiv \max_{s,a} |\hat{Q}_n(s,a) - Q(s,a)|$. Нека с *s'* да означим *δ(s, a)*. За всеки елемент от таблицата $\hat{Q}_n(s,a)$, който се обновява на итерация *n + 1*, размерът на грешка в тази ревизирана оценка $\hat{Q}_{n+1}(s,a)$ е

$$
\begin{aligned}
|\hat{Q}_{n+1}(s,a) - Q(s,a)| &= |(r + \gamma \max_{a'} \hat{Q}_n(s',a')) - (r + \gamma \max_{a'} Q(s',a'))| = \\
&= \gamma\, |\max_{a'} \hat{Q}_n(s',a') - \max_{a'} Q(s',a')| \le \\
&\le \gamma\, |\max_{a'} \hat{Q}_n(s',a') - Q(s',a')| \le \\
&\le \gamma\, |\max_{s'',a''} \hat{Q}_n(s'',a') - Q(s'',a')|
\end{aligned}
$$

Следователно $|\hat{Q}_{n+1}(s,a) - Q(s,a)| \le \gamma\Delta_n$

Третата линия в доказателството следва от втората, тъй като за всеки две функции f_1 и f_2 в сила е следното неравенство:

$$|\max_a f_1(a) - \max_a f_2(a)| \le \max_a |f_1(a) - f_2(a)|$$

При прехода от третата линия към четвъртата ние въведохме новата променлива *s''*, по която се извършва максимизацията. Това е позволено, тъй като максималната стойност ще бъде най-малко толкова голяма, като в случая, когато позволяваме тази допълнителна променлива да се променя. Обърнете внимание, че чрез въвеждането на тази променлива получаваме израз, който съвпада с дефиницията на Δ_n.

И така обновената стойност $\hat{Q}_{n+1}(s,a)$ за всяко *s, a* е равна най-много на γ умножена по максимална грешка в таблицата $\hat{Q}_n$ - Δ_n. Най-голямата грешка е в началната таблица - Δ_0 е ограничена, тъй като стойности на $\hat{Q}_0(s,a)$ и *Q(s, a)* са ограничени по условие за всички *s, a*. Следователно след първия интервал, в течение на който всяка двойка *s, a* е посетена, най-голямата грешка в таблицата ще бъде най-много $\gamma\Delta_0$. След *k* такива интервали грешката ще бъде най-много $\gamma^k\Delta_0$. Тъй като всяко състояние се посещава неограничено често, броят на такива интервали е безкраен и $\Delta_n \to 0$ при $n \to \infty$. Това доказва теоремата.

22.3.5. Стратегии за експериментиране

Обърнете внимание, че алгоритъм от Таблица 1 не указва, как се избират действия от агента. Една очевидна стратегия е агентът в състояние *s* да избира действието *a*, което максимизира $\hat{Q}(s,a)$, като по този начин използва текущата апроксимация $\hat{Q}$. Обаче при тази стратегия съществува риск, че агентът ще набляга върху действия, за които при ранните фази на обучение той намери, че имат високи стойности на $\hat{Q}$ и няма да тества други възможни действия, които имат дори по-високи стойности. Фактически теоремата за сходимост изисква всеки преход състояние-действие да се среща безкрайно често. Ясно, че това няма да стане, ако агентът винаги избира действия, които максимизират неговите текущи стойности на $\hat{Q}(s,a)$. По тази причина общоприето е в Q самообучение да се използва вероятностен подход към избора на действия. На действия с по-високи стойности на $\hat{Q}$ се назначава по-голяма вероятност, но всяко действие има ненулева вероятност. Един начин да назначаваме подобни вероятности е:

$$P(a_i \mid s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}}$$

където $P(a_i|s)$ е вероятността за избор на действие a_i при условие, че агентът се намира в състояние *s* и където $k > 0$ е една константа, определяща колко силно изборът ще фаворизира действия с високите стойности на $\hat{Q}$. По-големите стойности на *k* ще водят до назначаване на по-големи вероятности на действия, които имат стойности на $\hat{Q}$ над средните. Това ще кара агента да *използва* вече наученото и да търси действия, за които той вярва че ще максимизират наградата. Обратно на това, по-малките стойности на *k*

ще позволяват по-високите вероятности да бъдат назначавани на други действия, което ще кара агента да *изследва* действия, които в момента нямат високи стойности на $\hat{Q}$. В някои случаи k се варира заедно с броя на итерации, така че агентът в ранните стадии на обучение предпочита да изследва, а след това постепенно преминава към стратегията на преизползването.

22.3.6. Последователност на обновяване

Едно важно следствие от теоремата за сходимост е, че Q самообучение не се нуждае от обучение върху оптималната последователност от действия, за да има сходимост към оптималната политика. На практика то може да научи функцията Q (и, следователно, оптималната политика), докато се учи от действия, избрани напълно случайно на всяка стъпка при условие, че резултиращата обучаваща последователност обхожда всеки преход състояние-действие неограничено често. Този факт предполага промяна на последователността от обучаващите примери-преходи, за да подобри ефективността на самообучение без да застрашава крайната сходимост. За илюстрация да разгледаме отново самообучение в един МПВР с едно единствено поглъщащо целево състояние (както показания на Фиг. 22-1). Както и по-рано ще предполагаме, че обучаваме агента чрез някоя последователност от епизоди. За всеки епизод агентът се помества в някое случайно начално състояние и му се разрешава да изпълнява действия и да обновява таблицата $\hat{Q}$ докато той не достигне поглъщащото целево състояние. Всеки нов епизод започва с премахване на агента от целевото състояние и поставянето му в някое ново случайно избрано начално състояние. Както вече сме отбелязали по-рано, ако започваме със стойностите на $\hat{Q}$, инициализирани с нули, то след първия пълен епизод само един елемент от таблицата ще бъде променен – елементът, съответстващ на крайния преход в целевото състояние. Обърнете внимание, че ако по случайност във втория епизод агентът ще следва абсолютно същата последователност от действия от същото начално състояние, то и втория елемент от таблица ще стане ненулев и т.н. Ако по този начин изпълняваме повторно едни и същи епизоди, границата на ненулеви стойности на $\hat{Q}$ ще пълзи в посока от целевото състояние обратно към началното със скоростта един нов преход състояние-действие на епизод. А сега да разгледаме обучението на същите преходи състояние-действие, но в обратен хронологичен ред за всеки епизод. Т.е. ние прилагаме същото правило за обновление (22.7) за всеки разгледан преход, но изпълняваме тези обновявания в обратен ред. В този случай след първия пълен епизод агентът ще обнови своите оценка за $\hat{Q}$ за всеки преход от пътя, водещ от началното състояние към целевото. Ясно е, че този процес на обучение ще има сходимост за по-малък брой итерации, макар че той изисква агентът да използва повече памет, за да съхранява пълния епизод, преди да започне обучение за този епизод.

Втората стратегия за подобряване на скоростта на сходимост е да се пазят миналите преходи състояние-действие заедно с незабавната награда, която е била получена, а след това периодично да се прави върху тях повторното обучение. Макар че на пръв поглед изглежда като загубата на усилия да правиш повторно обучение върху същия преход, напомням, че обновената стойност $\hat{Q}(s,a)$ се определя от стойностите $\hat{Q}(s',a)$ на състояние-наследник $s' = \delta(s, a)$. По този начин, ако последвалото обучение променя една от стойностите $\hat{Q}(s',a)$, то повторното обучение върху прехода $<s, a>$

може да доведе до промяна на стойността $\hat{Q}(s,a)$. В общия случай степента, в която искаме да повтаряме старите преходи в сравнение с получаване на нови такива от средата, зависи от относителната цена на тези две операции във всяка конкретна проблемна област. Например в областта на роботи с навигационни действия, които могат да изискват няколко секунди за изпълнение, забавянето в събиране на един нов преход състояние-действие от външния свят може да бъде на няколко порядъка по-скъпо от вътрешното повтаряне на един вече наблюдаван преход. Тази разлика може да бъде много значителна при условие, че Q самообучение може често да изисква хиляди обучаващи итерации, за да се получи сходимостта.

Трябва да се отбележи, че в тази дискусия ние сме се придържали към предположението, че агентът не знае функцията на прехода състояние-действие *δ(s, a)*, използвана от средата или функция *r(s, a)*, използвана за генериране на награди. Ако той действително знае тези две функции, то са възможни много по-ефективни методи за обучение. Например, ако изпълнение на някое външно действие е скъпо, агентът може просто да игнорира средата и вместо това да я симулира вътрешно, ефективно генерирайки симулационни действия и назначавайки съответните симулирани награди.

22.4. Недетерминистични награди и действия

До сега сме разглеждали Q самообучение в детерминистични среди. В този раздел ще разгледаме недетерминистичен случай, в който функцията на награди *r(s, a)* и функцията на преходи *δ(s, a)* могат да имат вероятностен резултат. Например в програмата за игра на табла (Tesauro 1995) резултати от действия са напълно вероятностни, тъй като всяко движение включва хвърлянето на зарове. По същия начин в задачите с роботи с зашумени сензори и датчици (effectors) често е подходящо да се моделират действия и награди като недетерминистични. В такива случаи функциите *r(s, a)* и *δ(s, a)* могат да се разглеждат като, първо, произвеждащи някакво вероятностно разпределение на своите изходи, базиращо се на *s* и *a,* а след това избиращи някой резултат по случаен начин в съответствие с това разпределение. Когато тези вероятностни разпределения зависят само от *s* и *a* (т.е. те не зависят от предишните състояния или действия), наричаме системата недетерминистичен Марковски процес за вземане на решения.

За да разширим алгоритъма за Q самообучение на недетерминистичния случай, ще разгледаме внимателно аргументи, водещи към създаване на алгоритъма в детерминистичния случай, ревизирайки него, когато възникне необходимост за това.

В недетерминистичния случай ние трябва първо да преформулираме целта на системата за самообучение по такъв начин, че да отчита факта, че резултатите от действия вече не са детерминистични. Едно очевидно обобщение се състои в предефиниране на стойността V^{π} на политика π така, че тя да бъде *очакваната стойност* (върху тези недетерминистични изходи) на намалената кумулативна награда, получена чрез прилагане на тази политика:

$$V^{\pi}(s_t) = E\left[\sum_{i=0}^{\infty} \gamma^i r_{i+1}\right]$$

където, както и по-рано, последователността от награди r_{i+1} се генерира чрез следване на политиката π, започвайки от състояние s. Обърнете внимание, че това представлява обобщението на формула (22.1), което покрива детерминистичния случай.

Както и по-рано ще дефинираме оптималната политика π^* като политиката π, която максимизира V^{π} за всички състояния s. Сега ще обобщим въведената по-рано дефиниция на Q от формулата (22.4) отново чрез използване на очакваната стойност:

$$Q(s,a) = E[r(s,a) + \gamma V^*(\delta(s,a))] = E[r(s,a)] + \gamma E[V^*(\delta(s,a))] = \\ = E[r(s,a)] + \gamma \sum_{s'} P(s' \mid s,s) V^*(s') \qquad (22.8)$$

където $P(s' \mid s, a)$ е вероятността, че избор на действие a в състояние s ще произведе следващото състояние s'. Обърнете внимание, че тук сме използвали $P(s' \mid s, a)$ за да препишем очакваната стойност на $V^*(\delta(s, a))$ в термини на вероятностите, асоциирани с възможните изходи на вероятностната функция δ.

Както и по-рано можем да изразим Q рекурсивно:

$$Q(s,a) = E[r(s,a)] + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q(s',a') \qquad (22.9)$$

Което представлява обобщението на въведеното по-рано уравнение (22.6). И така, ние просто предефинирахме $Q(s, a)$ в недетерминистичния случай да бъде очакваната стойност на въведената по-рано величина за детерминистичния случай.

След като обобщихме дефиницията на Q за случая с недетерминистичните функции на средата r и δ, трябва да изведем и новото правило за обучение. Въведеното по-рано правило, изведеното за детерминистичния случай (формула (22.7)) не осигурява сходимостта при тази недетерминистична постановка на задачата. Да разгледаме, например, една недетерминистична функция на награди $r(s, a)$, която произвежда различни награди при всяко повторение на прехода $\langle s, a \rangle$. В този случай обучаващото правило ще променя всеки път стойностите на $Q(s, a)$, дори ако ние инициализираме стойностите на таблицата $\hat{Q}$ с коректните стойности на функцията Q. С други думи това обучаващото правило няма да има сходимост. Тази трудност може да бъде преодоляна чрез модифициране на обучаващото правило по такъв начин, че то да взима намаленото претеглено средно от текущата стойност на $\hat{Q}$ и на ревизираната оценка. Означавайки с $\hat{Q}_n$ оценката на агента на n-та итерация на алгоритъма, можем да изведем следното ревизирано обучаващо правило, което гарантира сходимостта на $\hat{Q}$ към Q:

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \max_{a'} \hat{Q}_{n-1}(s',a')] \quad (22.10)$$

където:

$$\alpha_n = \frac{1}{1 + visits_n(s,a)} \qquad (22.11)$$

В тази формула s и a са състояние и действие, обновени по време на n-та итерация, а $visits_n(s, a)$ е общия брой пъти, когато тази двойка състояние-действие беше посетена и включена в n-та итерация.

Основната идея на това ревизирано правило се състои в това, че ревизиите на $\hat{Q}$ се правят с по-малки стъпки в сравнение с детерминистичния случай. Обърнете внимание, че ако установим α_n в 1 в уравнение (22.10), то получим същото обучаващо правило както и в детерминистичния случай. С по-малки стойности на α този израз ще се осреднява с текущото $\hat{Q}(s,a)$, за да произведе новата обновена стойност. Стойността α_n в формула (22.11) намалява при нарастване на n, така че обновяванията стават по-малки с напредване на обучението. Чрез намаляване на α с подходяща скорост по време на обучението, можем да постигнем сходимостта към коректните стойности на функцията Q. Изборът на α_n е едно от условията, които трябва да бъдат удовлетворени, за да се получи сходимостта:

Теорема 2. Сходимост на Q самообучение за недетерминистични Марковски процеси за вземане на решения (Watkins and Dayan 1992). Да разгледаме някой Q самообучаващ се агент в недетерминистичен МПВР с ограничени награди $(\forall s,a \mid r(s,a) \mid \leq c$. Агентът използва обучаващото правило (22.10), инициализира своята таблица $\hat{Q}(s,a)$ с произволни крайни стойности и използва намаляващия фактор γ, такъв че $0 \leq \gamma < 1$. Нека *n(i, s, a)* е итерацията, съответстваща на *i*-я път, когато действието *a* се прилага в състояние *s*. Ако всяка двойка състояние-действие се посещава неограничено често, $0 \leq \alpha_n < 1$ и

$$\sum_{i=1}^{\infty} \alpha_{n(u,s,a)} = \infty, \quad \sum_{i=1}^{\infty} [\alpha_{n(u,s,a)}]^2 < \infty,$$

то за всички *s* и *a* следва, че $\hat{Q}_n(s,a) \to Q(s,a)$ при $n \to \infty$ с вероятност 1.

Когато може да се докаже, че Q самообучение и съответния алгоритъм за самообучение чрез стимули имат сходимост при определени условия, практическите системи, използващи Q самообучение, често изискват много хиляди обучаващи итерации за получаване на сходимостта. Например цитираният по-рано алгоритъм на Tesauro за игра на табла TD-GAMMON изисква обучение върху 1.5 милиона игри на табла, като всяка от тях съдържа десетки преходи състояние-действие.

22.5. Самообучение на времеви разлики

Алгоритмът за Q самообучение работи чрез итеративното намаляване на несъответствия между оценките на стойностите на Q за съседни състояния. В този смисъл Q самообучение представлява специалния случай на един общ клас *алгоритми за времеви разлики*, които научават чрез намаляване на несъответствия между оценките, направени от агента в различни моменти от време. Докато обучаващото правило (22.10) намалява разликите между оценъчните стойности $\hat{Q}$ на някое състояние и негов непосредствен наследник, ние можем да конструираме алгоритъм, който намалява несъответствия между това състояние и неговите по-отдалечени наследници.

Искам да припомня, че правилото за Q самообучение изчислява една обучаваща стойност $\hat{Q}(s_t,a_t)$ в термини на стойности за $\hat{Q}(s_{t+1},a_{t+1})$, където s_{t+1} е резултатът от прилагане на действие a_t към състояние s_t. Нека $Q^{(1)}(s_t, a_t)$ означава обучаващата стойност, изчислена при това едно-стъпково предвиждане:

$$Q^{(1)}(s_t, a_t) \equiv r_i + \gamma \max_a \hat{Q}(s_{t+1}, a)$$

Един алтернативен начин за изчисляване на обучаващата стойност за $Q(st, a_t)$ е на базата на наблюдаваната награда за две стъпки:

$$Q^{(2)}(s_t, a_t) \equiv r_i + \lambda r_{i+1} + \gamma^2 \max_a \hat{Q}(s_{t+2}, a)$$

или в общия случай – за n стъпки:

$$Q^{(2)}(s_t, a_t) \equiv r_i + \lambda r_{i+1} + ... + \gamma^{n-1} r_{n-1} + \gamma^n \max_a \hat{Q}(s_{t+n}, a)$$

Sutton (1988) въведе един общ метод за комбиниране на тези алтернативни обучаващи оценки, наречен TD(λ). Идеята е да се използва някаква константа $0 \le \lambda \le 1$, за да комбинира оценките, получени от различните последващи състояния по следния начин:

$$Q^{\lambda}(s_t, a_t) \equiv (1-\lambda)[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + ...]$$

или еквивалентната рекурсивната дефиниция:

$$Q^{\lambda}(s_t, a_t) = r_t + \gamma[(1-\lambda) \max_a \hat{Q}(s_t, a_t) + \lambda Q^{\lambda}(s_{t+1}, a_{t+1})]$$

Обърнете внимание, че ако изберем $\lambda = 0$, ще получим оригиналната обучаваща оценка $Q^{(1)}$, която разглежда само едно-стъпкови несъответствия в оценките $\hat{Q}$. При нарастване на λ алгоритмът придава по-голямо внимание на несъответствия, базирани на все по-отдалечени предвиждания. При екстремната стойност $\lambda = 1$ наблюдаваните стойности r_{t+i} ще се разглеждат без никакъв принос към текущата оценка $\hat{Q}$. Когато $\hat{Q} = Q$ обучаващите стойности, задавани от Q^{λ}, ще бъдат еднакви за всички стойности на λ, такива че $0 \le \lambda \le 1$.

Мотивацията за метода TD(λ) е, че при определени условия обучението ще бъде по-ефективно, ако повече по-отдалечени предвиждания се взимат под внимание. Например, когато агентът следва някоя оптимална политика за избор на действия, то Q^{λ} с $\lambda = 1$ ще предостави една перфектна оценка за истинската стойност на Q не зависимо от съществуващи несъответствия в $\hat{Q}$. От другата страна, ако последователността от действия се избира субоптимално, то стойностите на r_{t+i}, наблюдавани в далечно бъдеще, ще бъдат подвеждащи.

22.6. Обобщение от примери

Най-ограничаващото предположение в разгледаното до сега описание на Q самообучение се състои в това, че целевата функция се представя чрез една експлицитна таблица на търсене (lookup table), в която всеки един различен елемент от таблицата отговаря на всяка една различна входна величина (т.е. двойка състояние-действие). По този начин алгоритмите, които сме дискутирали до сега, представляват един вид алгоритми за обучение чрез механично повтаряне (rote learning) и не правят никакви опити да преценят стойности на Q за неизследвани двойки състояние-действие чрез обобщение от тези, който вече са били изследвани. Това предположение за

обучение чрез механично повтаряне е отразено в доказателството за сходимост, което доказва сходимостта само, ако всяка двойка състояние-действие е посетено (при това неограничено често!). Очевидно е, че това е едно нереалистично предположение в големи или безкрайни пространства, или когато цената за изпълнение на действие е висока. Като резултат повечето от практически системи често комбинират методи за апроксимация на функции, които сме дискутирали в други глави, с обучаващи правила за Q самообучение, разгледани в тази глава.

Един алгоритъм за апроксимиране на функции от типа на BACKPOPAGATION може лесно да се обедини с алгоритъма за Q самообучение чрез замяна на таблицата на търсене с невронна мрежа и използване на всяко обновяване $\hat{Q}(s,a)$ като обучаващ пример. Например можем да кодираме състояние s и действие a като входове на мрежата и да обучим мрежата да извежда целевите стойности на $\hat{Q}$ чрез правилата за обучение, зададени с формули (22.7) и (22.10). Една алтернатива на този подход, която понякога е по-успешна в някои практически случаи, е да обучаваме по една отделна мрежа за всяко действие, използвайки състоянието като вход, а $\hat{Q}$ - като изход. Една друга общоприета алтернатива се състои в обучение на една мрежа със състоянието като вход, но с по една $\hat{Q}$ като изход за всяко действие.

Програмата за игра на табла TD-GAMMON (1992) добре показва потенциала на техники за самообучение с използване на стимули. В своята ранна работа (Tesauro and Sejnowski 1989) авторите опитвали да научат представянето на *Q(s, a)* във вид на невронна мрежа директно от примери на ходове, маркирани с относителни стойности от човека-експерт. Този подход се оказа изключително досаден за експерта. Подходът е бил реализиран във вид на програма NEUROGAMMON, която е била силна по компютърни стандарти, но не е могла да се конкурира с хората-експерти. Проектът TD-GAMMON е бил един опит да се самообучава само от игри на програмата сама със себе си. Единственият сигнал за награда се е давал само в края на всяка игра. Оценъчната функция е била представяна от една напълно свързана невронна мрежа с един скрит слой, състоящ от 40 неврона. Чрез съчетание на алгоритъма BACKPROPAGATION с TD(λ)-обучаващото правило, TD-GAMMON се научи да играе значително по-добре от NEUROGAMMON, макар че входното представяне е съдържало само ситуациите на дъската без никакви други изчислени допълнителни признаци. Този процес на самообучение изискал около 200 000 обучаващи игри и две седмици компютърно време. Макар че това изглежда като доста игри, това представлява само една малка част от пространството на състояния. Когато допълнителни предварително изчислени признаци са били добавени към входното представяне, една мрежа с 80 скрити възли е била в състояние, след 300 000 обучаващи игри, да постигне нивото на играта, сравнимо с този на три най-добрите играчи – хора.

Трябва да се отбележи, че освен системи, успешно обединяващи Q самообучение с обобщение от примери, съществуват и други задачи, в които алгоритмите за самообучение чрез стимули не могат да получат сходимостта, когато те са обединени с някой алгоритъм за апроксимиране на функции. Обърнете внимание, че теоремата за сходимост се прилага само, когато $\hat{Q}$ е представена чрез експлицитна таблица. За да видим проблема, да разгледаме вместо такава таблица, използвана за представяне на $\hat{Q}$, една невронна мрежа. Когато алгоритъмът за самообучение обновява мрежата, за да получи по-добро приближение към обучаващите стойности на Q за някой конкретен

преход $<s_i, a_i>$, променените тегла на мрежата могат също така да променят оценките на $\hat{Q}$ и за други преходи. Тъй като тези промени на теглата могат да увеличат грешката в оценките на $\hat{Q}$ за тези други преходи, аргументът, доказващ оригиналната теорема, вече няма да е в сила.

22.7. Връзка с динамичното програмиране

Такива методи на самообучение чрез стимули като Q самообучение са тясно свързани с изследванията върху подходи за динамичното програмиране, предназначени за решаване на Марковски процеси за вземане на решения. В по-ранните работи обикновено се подразбира, че агентът притежава перфектни знания за функциите *δ(s, a)* и *r(s, a)*, които определят средата на агента. Следователно те основно адресират въпросът, как да бъде изчислена оптималната политика, използвайки най-малки изчислителни ресурси, при условие, че средата може да бъде симулирана абсолютно точно и че не се изискват преки взаимодействия със средата. Новият аспект на Q самообучение се състои в това, че то предполага, че агентът *не притежава* знания за *δ(s, a)* и *r(s, a)* и че вместо да се предвижва съгласно вътрешния мисловен модел на пространството от състояния, той трябва да се предвижва в реалния свят и да наблюдава последствия от своите действия. В този случай ние сме основно заинтересувани не в броя на изчислителни цикли, които агентът трябва да изразходва, а в броя на действия в реалния свят, които агентът трябва да извършва, за да осигури сходимостта към някоя приемлива политика. Причината е, че в повече от реални проблемни области, такива като, например, производствени задачи, цената за извършване на едно действие във външния свят, изразена във време и в пари, доминира над стойността на изчисления. Системите, които се самообучават чрез предвижване в реалната среда и чрез наблюдаване на резултатите от своите действия, обикновено се наричат он-лайн системи, докато тези, които се самообучават само чрез симулиране на действия в един вътрешен модел, се наричан оф-лайн системи.

Близкото съответствие между тези по-ранни подходи и задачите за самообучение чрез стимули, дискутирани в тази лекция, се вижда от разглеждане на уравнението на Белман, което лежи в основата на множество подходи за динамичното програмиране за решаване на МПВР. Уравнението на Белман е следното:

$$(\forall s \in A)\, V^*(s) = E[r(s, \pi(s) + \gamma V^*(\delta(s, \pi(s))]$$

Обърнете внимание на много близка връзка между това уравнение и въведената по-рано дефиниция на оптималната политика (22.2). Белман показа (1957), че оптималната политика π^* удовлетворява написаното по-горе уравнение, и че всяка политика π, удовлетворяваща това уравнение, представлява някоя оптимална политика. Ранните работи върху динамичното програмиране включват алгоритъм на Белман-Форд за намиране на най-краткия път, който научава пътища в графа чрез повтарящо се обновяване на оценъчното разстояние до целта от всеки възел на графа, базирайки се на разстояния до съседните възли. В този алгоритъм предположението, че дъгите на графа и целевия възел са известни, е еквивалентно на нашето предположение, че *δ(s, a)* и *r(s, a)* са известни.