UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 1. stopnja

Tim Kalan

Spodbujevalno učenje pri igranju namiznih iger

Delo diplomskega seminarja

Mentorica: izr. prof. dr. Marjetka Knez

Ljubljana, 2021

Kazalo

Spodbujevalno učenje pri igranju namiznih iger

Povzetek

V povzetku na kratko opišite vsebinske rezultate dela. Sem ne sodi razlaga organizacije dela – v katerem poglavju/razdelku je kaj, pač pa le opis vsebine.

Reinforcement learning in board games

Abstract

Prevod zgornjega povzetka v angleščino.

Math. Subj. Class. (2010): navedite vsaj eno klasifikacijsko oznako – dostopne so na `www.ams.org/mathscinet/msc/msc2010.html`

Ključne besede: Spodbujevalno učenje, Markovski proces odločanja

Keywords: Reinforcement learning, Markov decision process

1. Uvod

Namizne igre ljudje igramo že od prazgodovine. Na Kitajskem je bila igra Go znana kot ena izmed štirih umetnosti Kitajskega učenjaka poleg igranja inštrumenta s strunami, kaligrafije in slikanja. Spremljajo nas že zelo dolgo časa, zato je naravno, da jih želimo ljudje čim bolje igrati.

Z adventom računalnika in računalništva je bil ta problem postavljen v novi luči. Vprašanje ni bilo več samo, kako dobro lahko človek igra igro sam, temveč tudi do kakšnega nivoja lahko spravi računalnik. Izkazalo se je, da nam pri tem problemu (in mnogih drugih) zelo dobro koristi »umetna inteligenca« oz. metode strojnega učenja (SU). Eno izmed vej SU bomo predstavili v tem delu in pogledali, kako nam lahko pomaga pri igranju namiznih iger.

Ideja, da bi nek stroj igral igre ni nova, in kompleksnosti takega stroja so se zavedali ljudje že pred obstojem računalnika. Za konec uvodnega dela morda zabeležimo še citat iz eseja ameriškega pisatelja in pesnika Edgarja Allana Poea, ki govori o mehaničnem igralcu šaha:

> »Če prej omenjenemu [igralcu šaha] rečemo čisti stroj, moramo biti pripravljeni priznati, da je zunaj vseh primerjav, najbolj čudovit izum človeštva.

1.1. **Motivacija.** Spodbujevalno učenje ima zelo lepo motivacijo, in sicer izhaja iz psihologije. Znana psihologa Thorndike in Skinner, sta na živalih izvajala eksperimente; postavila sta jih v neko novo situacijo, kjer je lahko žival naredila akcijo, ki je rezultirala v neki nagradi. Ko je bila žival ponovno postavljena v to situacijo, je hitreje ugotovila, katero akcijo mora storiti, da pride do nagrade.

Koncept, ki je opisan v zgornjem odstavku, se imenuje instrumentalno pogojevanje. Z njim se srečamo tudi ljudje; tako se namreč učijo otroci, odrasli ljudje pa se bolj zanesejo na logično razmišljanje. Vseeno pa je to motiviralo utemeljitelje spodbujevanega učenja

1.2. **Strojno učenje.** To relativno novo raziskovalno področje se deli na tri glavne veje:

- **Nadzorovano učenje** se ukvarja s tem, kako iz nekih označenih podatkov naučimo računalnik, da prepoznava razne signale (slike, govor, tekst, ...) in to znanje uporabi za razpoznavo novih, neoznačenih podatkov.
- **Nenadzorovano učenje** odstrani označevanje iz podatkov in v njih probava odkriti skrite vzorce.
- **Spodbujevalno učenje** se ukvarja z »učenjem iz izkušenj.«

1.3. **Struktura naloge.** Naloga je razdeljena na štiri glavne dele. Na začetku so predstavljeni osnovni koncepti spodbujevalnega učenja in nekateri glavni algoritmi s tega področja. Potem se fokus obrne na namizne igre in ob nekaj malega teorije iger povzame osnovne koncepte, na katere naletimo tam. V naslednjem odseku potem združimo znanje iz prejšnjih dveh in predstavimo, kako nam teorija iger pripomore pri spodbujevalnem učenju v tem konteksu. Na koncu pa so predstavljeni nekateri empirični rezultati, ki so posledica zgoraj navedene teorije.

2. Spodbujevalno učenje

Spodbujevalno učenje se ukvarja s ti. učenjem iz interakcije oz. izkušenj. Čeprav se to na prvi pogled ne zdi kot računska metoda, pač pa stvar psihologije, bomo kmalu dognali, kako prevesti to idejo v računalniku razumljiv jezik.

2.1. Osnovni koncepti.

V osnovi nas zanima precej preprosta stvar: kako preslikati neko opazovano situacijo v akcijo na tak način, da učenec, ki mu v spodbujevalnem učenju pravimo **agent**, maksimirizra neko numerično nagrado. Pri tem ne obstaja opazovalec, ki bi agentu povedal ali pa namignil, katere akcije so dobre, to mora ugotoviti sam, s poskušanjem in napakami. V tem dejstvu se skriva bistvena razlika med spodbujevalnim učenjem in ostalimi vejami strojnega učenja.

Pomembna razlika tiči tudi v pomembnosti časa pri spodbujevalnem učenju. Pri drugih oblikah strojnega učenja se ponavadi ukvarjamo s tabelaričnimi podatki, tu pa ponavadi modeliramo nek dinamičen proces, zato je naravno, da je pomemben čas. Čeprav se ga da v tem kontekstu modelirati zvezno, je za naše namene dovolj, da ga jemljemo kot diskretne točke $t \in 1, \ldots, T$, kjer T označuje nek končni čas (v splošnem je lahko seveda $T = \infty$).

2.1.1. *Nagrada.*

Prvi pomemben koncept pri spodbujevanem učenju je nagrada. Kot smo že zgoraj omenili, to za nas pomeni neko numerično vrednost, pozitivno število indicira pozitivno nagrado, negativno pa »kazen«. S pomočjo tega koncepta formaliziramo *cilj* učenja. Edini cilj agenta je maksimizacija te nagrade, pri čemer je vredno omeniti, da na nagrado agent lahko vpliva samo s svojimi akcijami (ne more recimo spremeniti načina, na katerega dobi nagrado).

Posebaj pomembno je na tem mestu poudariti, da akcije nimajo nujno neposredne nagrade. Le-te lahko pridejo v poljubnem kasnejšem časovnem obdobju. To je smiselno, če pomislimo z vidika namiznih iger: pri šahu ne razmišljamo samo o neposrednih akcijah, temveč razvijamo neko dolgoročno strategijo ki nas na koncu nagradi z zmago.

Zgled 2.1 (Križci in krožci)**.** *Pri tej znani otroški igri (in pri mnogo drugih namiznih igrah) modeliramo nagrado na preprost način: če zmagamo, prejmemo nagrado* 1*, če izgubimo pa* -1*. V vseh ostalih situacijah, torej za izenačenje in po vsaki potezi, prejmemo nagrado* 0*.*

Zavedati se moramo tudi potencialnih omejitev oz. pomanjkljivosti takega modela. Razmislimo malo o:

Definicija 2.2 (Hipoteza o nagradi)**.** Vse cilje je mogoče opisati kot maksimizacijo neke kumulativne numerične nagrade.

Zgled 2.3 (Protiprimera hipotezi o nagradi)**.** *Problem je, da hipoteza dovoljuje samo enodimezionalnost:*

- *Ko kupujemo hamburger, nam je pomemben okus in cena; kaj nam več pomeni?*
- *Država želi med epidemijo ohraniti življenja in gospodarstvo; v kakšni meri naj prioritizira ti dve katergoriji?*

Na tem mestu poudarimo še, da se da tudi v takih situacijah modelirati nagrado na zgoraj opisani način in da je ta koncept vseeno dovolj splošen, da zajame zelo velik razred problemov.

2.1.2. *Okolje.* Okolje predstavlja del našega sistema, na katerega agent nima nobenega vpliva. Funkcija okolja je, da agentu pokaže **stanje** (angl. *state*) in mu da nagrado glede na **akcijo**, ki jo prejme od njega. Če se ponovno osredotočimo na namizne igre, bi lahko rekli, da je okolje igralna plošča pri šahu *in* nasprotnik - tudi nanj namreč nimamo vpliva. Okolje nam služi tudi kot sodnik akcij oz. stanj. V kontekstu programa za igranje iger torej okolje izbira nasprotnikove akcije, odloča katero stanje pomeni zmago in dodeljuje nagrade.

Zgled 2.4 (Križci in krožci)**.** *Okolje za nas pomeni* 3×3 *igralno polje in našega nasprotnika.*

2.1.3. *Agent.* Kot smo že omenili zgoraj, je agent naš učenec. Njegov cilj je torej maksimizacija numerične nagrade, to težnjo pa dosega s pomočjo **strategije** (angl. *policy*), ki mu pove, katero akcijo naj izbere v določenem stanju. Za ocenjevanje stanja, si pomaga z **vrednostno funkcijo** (angl. *value function*). Kot ime implicira, je to funkcija, ki določa vrednosti stanjem (in akcijam).

Nagrada nam pove takojšnjo vrednost stanja, vrednostna funkcija pa to vrednost gleda na dolgi rok. Je izpeljanka nagrade, a veliko bolj primerna za maksimizacijo kot nagrada, saj upošteva, da so tudi stanja, ki ne prinesejo takojšnje nagrade, lahko veliko vredna.

Poleg tega je v splošnem lažje učenje prek vrednostnih funkcij kot prek strategij neposredno, saj je ponavadi stanj mnogokrat manj kot možnih strategij agenta.

Formalno gledano:

Definicija 2.5. Naj R_t, S_t, A_t zaporedoma označujejo nagrado, stanje in akcijo ob času t. Definiramo naslednje pojme:

- Agentova **strategija** je takšna preslikava $\pi : S \to A$ da velja
$$a = \pi(s) \text{ oz.}$$
$$\pi(a|s) = P(A_t = a \mid S_t = s).$$
Pri čemer prva formula definira *deterministično* strategijo, druga pa *stohastično*. a in s sta realizaciji akcije in stanja v času t (to sta načeloma slučajni spremenljivki.)
- Naj bodo $R_{t+1}, ..., R_T$ nagrade, ji jih bomo prejeli od trenutka t do končnega časa. **Povračilo** (angl. *return*) G_t v splošnem definiramo za $T = \infty$
$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$
kjer je $\gamma \in [0, 1]$ *diskontni faktor*. Predstavlja dejstvo, da imamo raje nagrade, ki bodo prišle prej. Formalno gledano, je cilj učenja maksimizacija pričakovanega povračila
- Naj bo π dana strategija agenta. **Vrednostna funkcija stanja** (angl. *state value function*) glede na strategijo π je
$$v_\pi(s) = \mathrm{E}[G_t \mid S_t = s].$$
Predstavlja torej pričakovani izplen, če se vedemo skladno s strategijo π.
- Naj bo π še vedno dana strategija agenta. **Vrednostna funkcija akcije** (tudi stanja-akcije) (angl. *action value function*) glede na strategijo π definiramo
$$q_\pi(s, a) = \mathrm{E}[G_t \mid S_t = s, A_t = a].$$

Pove nam pričakovani izplen, če ob času t naredimo akcijo a, nato pa se vedemo skladno s strategijo π.

Zgled 2.6 (Križci in krožci)**.** *Agent je v tem primeru računalniški program, ki prejme igralno ploščo, nasprotnikove poteze in nagrade, vrne pa optimalno strategijo (to si želimo).*

2.1.4. *Model.* Model je nenujen del našega sistema. Predstavlja znanje, ki ga ima agent o svojem okolju. Če imamo model, da lahko uporabimo, da napovemo, kako se bo vedlo okolje in s tem premaknemo agentovo učenje iz čistih poskusov in napak na *načrtovanje* (angl. *planning*). Model je torej poleg strategije in vrednostne funkcije še tretja komponenta agenta. Na podlagi modela lahko agent »preračuna« smiselnost svojih akcij, brez da bi dejansko karkoli storil.

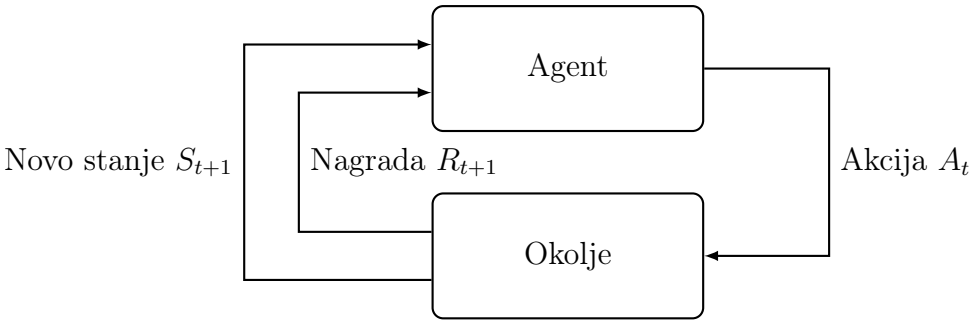
Prisotnost modela je glavna ločnica med dvema velikima, a zelo različnima vejama spodbujevalnega učenja.

2.2. **Korak spodbujevalnega učenja.** Spodbujevalno učenje se pogosto ukvarja s procesi, ki naravno razpadejo v ti .**epizode**. Tak proces so recimo namizne igre, kjer so epizode precej naravno posamezne igre. Ni pa nujno, da je delitev tako naravna (ali pa sploh možna oz. smiselna). Za namene te diplomske naloge lahko privzamemo, da taka delitev obstaja.

Ideja učenja je, da agenta spustimo v okolje in mu dovolimo, da doživi (igra) mnogo epizod. Nato na nek način (bo razjasnjeno kasneje) ob nekih določenih časih (npr. po koncu epizode) posodobi svojo strategijo (in/ali vrednostno funkcijo).

Dejanski korak (npr. poteza v namizni igri) v epizodi pa formalno gledano opredelimo:

- Agent naredi akcijo A_t ob prejetem stanju S_t in prejme nagrado R_t.
- Okolje prejme akcijo A_t, posreduje agentu stanje S_{t+1} in nagrado R_{t+1}



2.2.1. *Raziskovanje in izkoriščanje.* Eden izmed glavnih problemov, s katerim se srečamo pri spodbujevalnem učenju je problem raziskovanja in izkoriščanja. Ko se agent uči, začne dojemati katere akcije ali pa kombinacije akcij mu pripeljejo nagrado. Ko to ugotovi, seveda lahko začne te akcije *izkoriščati* in prejemativso nagrado, jim pripada. Pri tem pa naletimo na problem. Agent lahko izkorišča te akcije in nikoli ne ugotovi, da neka druga akcija prinese še višjo nagrado; tega ne izve, ker ne *raziskuje*. Če pa samo raziskuje pa nikoli ne izkoristi potencialnih nagrad, ki jih sreča to je, ničesar se ne nauči.

Uravnoteženje raziskovanja in izkoriščanja je pomemben problem, a se izkaže, da ima dokaj enostavno rešitev (ki deluje dovolj dobro). Spoznali jo bomo v kratkem.

Morda se nekaterim bralcem zdi, da smo zaenkrat preceč »mahali z rokami«, to je zato, ker želimo, da se do te točke razvije intuicija o predstavljenih pojmih. V nadaljevanju bomo do sedaj opisane stvari bolj formalizirali.

2.3. **Markovski proces odločanja.** Spomnimo se najprej procesa spodbujevalnega učenja in ga poskusimo opisati bolj formalno: imamo zaporedje časovnih korakov $t = 0, 1, 2, \ldots$, ob katerih med sabo interaktirata agent in okolje. Ob koraku t agent prejme od okolja stanje (oz. reprezentacijo stanja) $S_t \in \mathcal{S}$, kjer $\mathcal{S}$ označuje množico vseh stanj. Na podalagi stanja in strategije, ki jo ima, izbere akcijo $A_t \in \mathcal{A}(S_t)$, kjer $\mathcal{A}(S_t)$ predstavlja množico akcij, ki jih ima agent na voljo v stanju S_t. Rezultat te akcije je nagrada $R_{t+1} \in \mathcal{R}$ in novo stanje S_{t+1}.

Čeprav se da vse opisane koncepte posplošiti na števne in celo neštevne množice stanj in akcij, se bomo mi omejili na končne množice. To je glede na problem, s katerim se ukvarajmo, dovolj.

2.3.1. *Markovska veriga.* Dogajanje pri spodbujevalnem učenju lahko v grobem opišemo s slučajim procesom stanj $(S_t)_{t=0}^T$. Zato je pomembno, da si natančno pogledamo nekaj lastnosti, ki jih lahko pričakujemo.

Definicija 2.7 (Markovska veriga)**.** Slučajni proces $(S_t)_{t=0}^T$ na končnem verjetnostnem prostoru (Ω, P) je **Markovska veriga oz. Markovski proces** (angl. *Markov chain*), če zanj velja Markovska lastnost

$$P(S_{t+1} = s_{t+1} \mid S_t = s_t, ..., S_0 = s_0) = P(S_{t+1} = s_{t+1} \mid S_t = s_t).$$

Na kratko to *prehodno verjetnost* označimo $p_{ss'} := P(S_{t+1} = s' \mid S_t = s)$. Opazimo, da lahko te verjetnoti zložimo v matriko $\mathcal{P} := [p_{ss'}]_{s,s' \in \mathcal{S}}$, ki ji pravimo *prehodna matrika*.

Zdaj Markovsko verigo predstavimo še na alternativni način: kot dvojico $(\mathcal{S}, \mathcal{P})$, kjer je $\mathcal{P}$ zgoraj definirana prehodna matrika.

Markovska lastnost pomeni, da je prihodnost neodvisna od preteklosti, če poznamo sedanjost. Spodbujevalno učenje se ukvarja predvsem s problemi, kjer to dejstvo drži. Tudi pri našem ciljnem problemu to načeloma velja: če pogledamo igralno ploščo na katerikoli točki pogosto izvemo enako o trenutnem stanju, kot če bi opazovali igro od začetka.

2.3.2. *Markovski proces nagrajevanja.* Podoben koncept, ki se malo bolj približa dejanski situaciji v spodbujevalnem učenju je *Markovski proces nagrajevanja*. Kot že ime morda namigne, je precej podoben Markovski verigi, le da v njem nastopajo *nagrade*.

Definicija 2.8 (Markovski proces nagrajevanja)**.** Pričakovani nagradi glede na stanje s in akcijo a pravimo **nagradna funkcija** (angl. *reward function*) in označimo

$$\mathcal{R}_s^a = E[R_{t+1} \mid S_t = s, A_t = a].$$

Naboru $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$, za katerega velja

- $\mathcal{S}$ je (končna) množica stanj,
- $\mathcal{P}$ je prehodna matrika $\mathcal{P}_{ss'} = P(S_{t+1} = s' \mid S_t = s)$,
- $\mathcal{R}$ je *nagradna funkcija* $\mathcal{R}_s = \mathrm{E}[R_{t+1} \mid S_t = s]$,
- $\gamma \in [0, 1]$ je *diskontni faktor*,

pravimo **Markovski proces nagrajevanja** (angl. *Markov reward process*).

Od navadne Markovske verige se torej razlikuje samo v prisotnosti nagrad ob vsakem koraku. Če te nagrade postavimo na $R_t = 0 \forall t$, dobimo navadno Markovsko verigo. Pomembna razlika je še prisotnost diskontnega faktorja γ. Če velja $\gamma < 1$, potem procesu pravimo *diskontirani Markovski proces nagrajevanja.*

Diskontiranje je motivirano iz različnih vidikov: po eni strani nam pomaga, da se v primeru cikličnih procesov izognemo neomejenim povračilom. Poleg tega pa je diskontiranje v mnogo pogledih naraven način za opis situacije: pogosto imamo raje nagrade, ki pridejo prej. Primer tega poznamo recimo iz ekonomije; denar, ki ga dobimo kasneje nam pomeni manj, kot tisti, ki ga dobimo takoj.

Še vedno pa tovrsten proces ne opiše situacije v kateri se znajdemo pri spodbujevalnem učenju, saj ne vsebuje koncepta akcij. Je pa že dovolj »globok«, da v njem lahko definiramo vrednostno funkcijo $v(s) = \mathrm{E}[G_t \mid S_t = s]$, ki je v tem primeru neodvisna od strategije π, saj strategija v tem modelu nima pomena.

Za vrednostno funkcijo lahko izpeljemo rekurzivno enačbo, s pomočjo katere lahko »rešimo« Markovski proces nagrajevanja. S tem mislimo, da vsakemu stanju dodelimo pravo vrednost.

$$\begin{aligned}
v(s) &= \mathrm{E}[G_t \mid S_t = s] \\
&= \mathrm{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s] \\
&= \mathrm{E}[R_{t+1} + \sum_{k=1}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s] \\
&= \mathrm{E}[R_{t+1} + \gamma(\sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k+1}) \mid S_t = s] \\
&= \mathrm{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \mathrm{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]
\end{aligned}$$

Dobili smo torej

$$v(s) = \mathrm{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \text{ oz.} \tag{1}$$

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s'), \tag{2}$$

kjer smo v drugi vrstici upoštevali aditivnost in idempotentnost matematičnega upanja. To je **Bellmanova enačba za Markovkse procese nagrajevanja**.

Ker se ukvarjamo s primerom, ko je stanj končno mnogo, recimo n, jih lahko brez škode za splošnost označimo kar $1, \ldots, n$. Potem lahko Bellmanovo enačbo prepišemo v matrični obliki

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} \ldots \mathcal{P}_{1n} \\ \vdots \\ \mathcal{P}_{n1} \ldots \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} \text{ ali krajše}$$

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

Opazimo, da je to *linearna* enačba in da v splošnem eksplicitno rešitev:

$$
\begin{aligned}
v &= \mathcal{R} + \gamma \mathcal{P} v \\
(I - \gamma \mathcal{P}) v &= \mathcal{R} \\
v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}
\end{aligned}
$$

Ta rešitev predpostavlja obrnljivost matrike $(I - \gamma\mathcal{P})$ in računanje njenega inverza, kar zahteva $O(n^3)$ operacij, zato je smiselna samo za majhne procese. Za večje obstajajo iterativni algoritmi in metode, nekater izmed njih bomo spoznali v naslenjem odseku.

Ta del lahko še rahlo matematiziraš - enoličnost rešitev

2.3.3. *Markovski proces odločanja.* Kot ime že nakazuje, *Markovski proces odločanja (MDP)* razširja koncept procesa nagrajevanja z dodatkom odločanja - akcij. S tem dodatkom imamo v popolnosti opisan problem spodbujevalnega učenja: opisujemo torej nek proces, kjer ima agent možnost odločanja, izid pa je vsaj delno slučajen in odvisen od okolja.

Definicija 2.9 (Markovski proces odločanja)**.** Naboru $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, za katerega velja

- $\mathcal{S}$ je (končna) množica stanj,
- $\mathcal{A}$ je (končna) množica akcij,
- $\mathcal{P}$ je prehodna matrika $\mathcal{P}^a_{ss'} = P(S_{t+1} = s' \mid S_t = s, A_t = a)$,
- $\mathcal{R}$ je *nagradna funkcija* $\mathcal{R}^a_s = \mathrm{E}[R_{t+1} \mid S_t = s, A_t = a]$,
- $\gamma \in [0, 1]$ je *diskontni faktor*,

pravimo **Markovski proces odločanja** (angl. *Markov decision process*).

Prehodna matrika in nagradna funkcija sta sedaj seveda odvisna tudi od akcije. V kontekstu MDP-jev lahko sedaj formalno definiramo strategijo agenta π, ki je zahvaljujoč Markovski lastnosti odvisna od enega stanja in ne celotne zgodovine procesa. Definiramo tudi vrednostno funkcijo stanja $v_\pi(s)$ in vrednostno funkcijo akcije $q_\pi(s, a)$. Njihove definicije se ne spremenijo, so pa sedaj formalno umeščene v model.

Opazimo tudi, da je v MDP-ju pri fiksni strategiji π zaporedje oz. proces stanj $S_1, S_2, \ldots$ Markovska veriga $(\mathcal{S}, \mathcal{P}^\pi)$. Če v zaporedje stanj pomešamo še nagrade, torej $S_1, R_2, S_2, R_3, \ldots$, dobimo Markovski proces nagrajevanja $(\mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma)$, kjer smo označili

$$
\begin{aligned}
\mathcal{P}^\pi &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}^a_{ss'} \\
\mathcal{R}^\pi &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}^a_s.
\end{aligned}
$$

Opomba 2.10. Morda je nekatere bralce zbodlo, da v zaporedju stanj in nagrad S_1 ne sledi R_1, temveč R_2. Za tako notacijo smo se odločili, da poudarimo, da okolje agentu sočasno poda S_2 in R_2, glede na stanje S_1 pa se agent odloča o akciji A_1.

MDP-ji so splošna orodja za obravnavo stohastičnih procesov, ki vključujejo odločitve v diskretnem času. Njihov utemeljitelj je Richard Bellman, ki je znan predvsem po izumu dinamičnega programiranja, zato morda ni presenetljivo, da nam prav

dinamično programiranje poda osnovo za njihovo reševanje. Kot pri procesih nagrajevanja, lahko tudi tu izpeljemo Bellmanovo enačbo. Najprej zapišemo rekurzivni zvezi za vrednostni funkciji stanjja in akcije:

$$v_\pi(s) = \mathrm{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$
$$q_\pi(s,a) = \mathrm{E}[R_{t+1} + \gamma q_\pi(S_{t+1}) \mid S_t = s, A_t = a].$$

Opazimo, da v_π in q_π povezujta zvezi:

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) q_\pi(s,a)$$
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s').$$

Če zvezi združimo, dobimo:

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left[\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')\right]$$
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \left[\sum_{a'\in\mathcal{A}} \pi(a'|s') q_\pi(s',a')\right].$$

Prvo enačbo (dej to un reference) imenujemo **Bellmanova enačba pričakovanja** (angl. *Bellman expectation equation*), ki ima ponovno matrično obliko in eksplicitno (ENOLIČNO - ZAKAJ) rešitev:

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$
$$v_\pi = (I - \gamma\mathcal{P}^\pi)^{-1}\mathcal{R}^\pi$$

Ta enačba je sicer pomembna, a nas najbolj zanima »rešitev« MDP-ja. Želimo najti torej optimalno strategijo. Pri tem nam pomaga naslednje:

Definicija 2.11.

- **Optimalna vrednostna funkcija stanja** (angl. *optimal state-value function*) $v_*(s)$ je največja med vsemi vrednostnimi funkcijami stanj
$$v_*(s) = \max_\pi v_\pi(s).$$
- **Optimalna vrednostna funkcija akcije** (angl. *optimal action-value function*) $q_*(s,a)$ je največja med vsemi vrednostnimi funkcijami akcij
$$q_*(s,a) = \max_\pi q_\pi(s,a).$$
- Na množici vseh možnih strategij Π definiramo delno urejenost na naslednji način:
$$\pi \geq \pi', \text{če } v_\pi(s) \geq v_{\pi'}(s)\ \forall s.$$

Do konca nas potem pripelje naslednji izrek.

Izrek 2.12. *Za vsak Markovski proces odločanja velja:*

3. Algoritmi pri spodbujevalnem učenju

Algoritmov za reševanje MDP-jev je precej. Mi se bomo omejili predvsem na algoritme, ki se učijo prek vrednostne funkcije in izhajajo iz dinamičnega programiranja, a naj na tem mestu omenimo, da obstajajo tudi algoritmi, ki posodabljajo strategijo neposredno (mogoče kakšen reference al pa kej).

3.1. **Dinamično programiranje.**

Je optimizacijska metoda deluje na principu deljenja velikega problema na manjše prekrivajoče podprobleme. V jedru reševanja je **Bellmanova enačba**, ki opisuje odnos med vrednostmi podproblemov in glavnega problema. Ker je idejo dinamičnega programiranja (DP) in MDP-jev dobil Bellman ob istem času, je naravno, da je DP metoda, ki je prilagojena prav situaciji v MDP-ju.

Morda formalno bolj o bellmanovih enačbah?

3.1.1. *MRP.*

Spomnimo se Markovskega procesa nagrajevanja, torej $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$. Na tej enostavnejši verziji MDP-ja, lahko nazorno pokažemo uporabo dinamičnega programiranja za reševanje. Spomnimo se še vrednostne funkcije in jo poskušajmo razdeliti na dva dela; neposredno nagrado R_{t+1} in *diskontirano* vrednost naslednjega stanja S_{t+1}.

3.1.2. *MDP.*

3.2. **Monte Carlo.**

3.3. **TD(0).**

3.4. **TD(λ).**

3.5. **Izboljšave.**

3.5.1. *Nevronske mreže.*

4. Namizne igre

4.1. **Pregled konceptov teorije iger.**

4.1.1. *Nashevo ravnotežje.*

4.1.2. *Igre z vsoto nič.*

4.1.3. *Ekstenzivne igre.*

4.2. **Kompleksnost iger.**

4.2.1. *Game tree ...*

4.3. **Morda kaj o optimal board representationu?**

4.4. **Pride še kaj v poštev tu?**

5. Spodbujevalno učenje pri namiznih igrah

5.1. **Parcialni model - »po-stanja«.**

5.2. **Učenje.**

5.2.1. *Samoigra.*

5.2.2. *Igre iz podatkovnih baz.*

Literatura

[1] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.

[2] Richard E. Bellman. A markov decision process. *Journal of Mathematical Mechanics*, (6), 1957.

[3] Imran Ghory. Reinforcement learning in board games. 2004.

[4] David Silver. Introduction to reinforcement learning. `https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver`, 2015.

[5] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An introduction*. The MIT Press, Cambridge, Massachusetts, 2 edition, 2015.

[6] Csaba Szepesvari. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, Alberta, Canada, 2009.