UNIVERZITET METROPOLITAN BEOGRAD

CS360 - VEŠTAČKA INTELIGENCIJA

Inteligentni agenti

Lekcija 02

PRIRUČNIK ZA STUDENTE

CS360 - VEŠTAČKA INTELIGENCIJA

Lekcija 02

INTELIGENTNI AGENTI

- Inteligentni agenti
- Poglavlje 1: Agenti i okruženje
- Poglavlje 2: Dobro ponašanje: koncept racionalnosti
- Poglavlje 3: Priroda agentovog okruženja
- Poglavlje 4: Struktura agenata
- Poglavlje 5: Primena inteligentnih agenata
- Poglavlje 6: Pokazne vežbe
- Poglavlje 7: Domaći zadatk
- Zaključak

UVOD

U ovoj sekciji dat je uvod.

U ovoj lekciji upoznaćemo se sa inteligentnim agentima. Proći ćemo najpre kroz definiciju inteligentnog agenta i njegovog okruženja. Nakon toga, biće detaljno objašnjen koncept racionalnosti i šta čini jednog agenta da bismo mogli da kažemo da je on racionalan. Zatim, biće reči o prirodi agentovog okruženja, i detaljno ćemo objasniti razlike između određenih tipova okruženja:

- Potpuno opservabilno u odnosu na delimično opservabilno
- Okruženje sa jednim agentom u odnosu na više agenata
- Determinističko u odnosu na stohastičko okruženje
- Epizodno okruženje zadatka u odnosu na sekvencijalno
- Statičko u odnosu na dinamičko okruženje
- Diskretno okruženje u odnosu na kontinualno
- Poznato okruženje u odnosnu na nepoznato

Takođe, biće objašnjena struktura agenata i četiri osnovne vrste agenata koji su u osnovi svih inteligentnih sistema:

- Jednostavni refleksni agenti
- Refleksni agenti zasnovani na modelu
- Agenti zasnovani na ciljevima
- Agenti zasnovani na korisnosti

Biće objašnjeni i obučavajući agenti, kao posebna kategorija, pošto se sve prethodne strukture agenata mogu pretvoriti u obučavajuće agente.

Na kraju, biće dat prikaz primene inteligentnih agenata u svakodnevnom životu.

Poglavlje 1

Agenti i okruženje

DEFINICIJA INTELIGENTNOG AGENTA

U ovoj sekciji data je definicija inteligentnog agenta.

Agent je sve što se može posmatrati kao percepcija svog okruženja kroz *senzore* i delovanje na to okruženje putem *aktuatora*.

U veštačkoj inteligenciji, inteligentni agent (IA) je bilo šta što percipira svoju okolinu, preduzima akcije autonomno da bi postiglo ciljeve i može poboljšati svoje performanse učenjem ili može koristiti znanje. Oni mogu biti jednostavni ili složeni - termostat se smatra primerom inteligentnog agenta, kao i ljudsko biće, kao i svaki sistem koji ispunjava definiciju, kao što je firma ili država.

Agent ima „*funkciju cilja*“ koja obuhvata sve ciljeve IA. Takav agent je dizajniran da kreira i izvrši bilo koji plan koji će, po završetku, maksimizirati očekivanu vrednost funkcije cilja. Na primer, agent za učenje sa pojačanjem ima „funkciju nagrađivanja“ koja omogućava programerima da oblikuju željeno ponašanje IA, a ponašanje evolucionog algoritma je oblikovano „funkcijom fitnesa“.

Inteligentni agenti u veštačkoj inteligenciji su usko povezani sa agentima u ekonomiji, a verzije paradigme inteligentnog agenta se proučavaju u kognitivnoj nauci, etici, filozofiji praktičnog razuma, kao i u mnogim interdisciplinarnim socio-kognitivnim modeliranjem i kompjuterskim društvenim simulacijama.

Inteligentni agenti se često šematski opisuju kao apstraktni funkcionalni sistem sličan kompjuterskom programu. Apstraktni opisi inteligentnih agenata se nazivaju *apstraktni inteligentni agenti* (AIA) da bi se razlikovali od njihovih implementacija u stvarnom svetu. Autonomni inteligentni agent je dizajniran da funkcioniše u odsustvu ljudske intervencije. Inteligentni agenti su takođe blisko povezani sa *softverskim agentima* (autonomni računarski program koji izvršava zadatke u ime korisnika).

AGENTI I OKRUŽENJE

U ovoj sekciji definisano je okruženje agenta.

Kao što smo već rekli, agent je bilo šta što vrši percepciju svog okruženja kroz *senzore* i deluje na to *okruženje* putem *aktuatora*. Ova jednostavna ideja je ilustrovana na Slici 1. Ljudski agent ima oči, uši i druge organe za senzore i ruke, noge, vokalni trakt, itd. za aktuatore. Robotski agent može imati kamere i infracrvene daljinomere za senzore i razne motore za aktuatore. Softverski agent prima pritiske na tastere, sadržaj datoteka i mrežne

pakete kao senzorne ulaze i deluje na okruženje tako što prikazuje na ekranu, piše datoteke i šalje mrežne pakete. Kada kažemo okruženje, ono može biti bukvalno sve - u teoriji čitav univerzum! U praksi, to je deo univerzuma o čijim stanjima brinemo kada dizajniramo agenta - to je onaj deo koji utiče na ono što agent opaža, i što je "pogođeno" agentovim akcijama.

Koristimo termin *percepcija* ili opažanje što će se odnositi na bilo kakav sadržaj koji agentovi senzori opažaju. Agentova *sekvenca opažanja* je kompletna istorija svega što je agent ikada primetio. Uopšteno, agentov izbor akcije u bilo kom trenutku može zavisiti od njegovog ugrađenog znanja i celokupne sekvence opažanja do tada, ali ne i od nečega što nije primetio.

Specificirajući agentov izbor akcije za svaku moguću sekvencu percepcije, rekli smo manje-više sve što se može reći o agentu. Matematički govoreći, kažemo da je ponašanje agenta opisano *funkcijom* ***agenta*** koja preslikava bilo koju datu sekvencu percepcije u akciju. Interno, funkciju agenta za veštačkog agenta će implementirati *agentski program*. Važno je da ove dve ideje budu različite. Funkcija agenta je apstraktan matematički opis, a agentski program je konkretna implementacija pokrenuta unutar nekog fizičkog sistema.

Slika 1.1 Agent u interakciji sa okruženjem kroz senzore i aktuatore [1]

PRIMER ROBOTSKOG AGENTA U SVETU USISIVAČA

U ovoj sekciji dat je primer robotskog agenta u svetu usisivača kako bismo ilustrovali prethodno uvedene koncepte.

Kako bismo ilustrovali ideje koje smo uveli koristimo jednostavan primer - svet usisivača koji se sastoji od robotskog agenta usisivača u svetu koji čine kvadrati koji mogu biti prljavi ili čisti. Na Slici 2 prikazana je jednostavna konfiguracija sa samo dva kvadrata, A i B. Agent opaža u kom kvadratu se nalazi i da li ima prljavštine u njemu. Agent počinje od kvadrata A. Dostupne akcije su: pomeri se na desno, pomeri se levo, usisaj prljavštinu ili ne radi ništa. Jedna veoma jednostavna funkcija agenta je sledeća: ako je trenutni kvadrat prljav, usisaj ga, u suprotnom, pomeri se u drugi kvadrat. Ovo je sve pojednostavljeno u cilju ilustracije osnovnih koncepata inteligentnih agenata. Delimična tabela funkcije agenta prikazana je na Slici 3.

Slika 1.2 Svet usisivača sa samo dve lokacije [1].

Slika 1.3 Delimična tabela jednostavne funkcije agenta za svet usisivača [1].

Gledajući Sliku 3 vidimo da različiti agenti iz sveta usisivača mogu biti definisani jednostavnim popunjavanjem desne kolone tabele. Postavlja se pitanje, koji je najbolji način da se tabela popuni? Drugim rečima, šta čini jednog agenta da kažemo da je glup, pametan, dobar ili loš? Na ova pitanja ćemo odgovoriti u narednim sekcijama.

Poglavlje 2

Dobro ponašanje: koncept racionalnosti

KONCEPT RACIONALNOSTI

U ovoj sekciji opisan je koncept racionalnosti.

Kao što smo već rekli u prethodnoj lekciji, *racionalni agent* je onaj agent koji radi pravu stvar. Postavlja se pitanje, šta je to prava stvar? Moralna filozofija je razvila nekoliko različitih notacija "prave stvari", ali VI se drži konsekvecionalizma: evaluacije agentovog ponašanja na osnovu posledica. Kada se agent "pusti" u njegovo okruženje, on generiše sekvencu opservacija na osnovu opažaja koje prima. Ova sekvenca akcija prouzrokuje da okruženje prolazi kroz sekvencu stanja. Ako je sekvenca poželjna, to znači da je agent bio "dobar". Ova notacija poželjnosti se sadrži u meri performansi (meri učinka) koja evaluira bilo koju datu sekvencu stanja okruženja.
Primetite da smo rekli stanja okruženja, a ne stanja agenta. Ako uspeh definišemo u smislu *agentovog mišljenja o sopstvenom učinku*, agent bi mogao postići savršenu racionalnost jednostavno zavaravajući sebe da je njegov učinak savršen. Ljudski agenti su posebno poznati po „kiselom grožđu“ – verujući da nisu zaista želeli nešto (npr. Nobelovu nagradu) nakon što to nisu dobili.
Očigledno, ne postoji jedno fiksno merilo učinka za sve zadatke i agente; tipično, dizajner će osmisliti jedan prikladan okolnostima. Ovo nije tako lako kao što zvuči. Razmotrite, na primer, agenta iz sveta usisivača iz prethodne sekcije. Možemo predložiti merenje učinka količinom prljavštine koja je očišćena u jednoj osmosatnoj smeni. Sa racionalnim agentom, naravno, ono što tražite je ono što dobijate. Racionalni agent može maksimizirati ovu meru učinka tako što će očistiti prljavštinu, zatim je sve baciti na pod, zatim ponovo očistiti i tako dalje. Prikladnija mera učinka bi nagradila agenta za čist pod. Na primer, jedan poen bi se mogao dodeliti za svaki čisti kvadrat u svakom vremenskom koraku (možda sa kaznom za potrošenu električnu energiju i generisanu buku). Kao opšte pravilo, bolje je dizajnirati mere učinka u skladu sa onim što neko zapravo želi u okruženju, a ne prema tome kako misli da agent treba da se ponaša.

Čak i kada se izbegnu očigledne zamke, ostaju neka pitanja koja treba razmotriti. Na primer, pojam „čist pod” u prethodnom odeljku zasniva se na prosečnoj čistoći tokom vremena. Ipak, istu prosečnu čistoću mogu postići dva različita agenta, od kojih jedan obavlja osrednji posao sve vreme, dok drugi čisti energično, ali pravi velike pauze. Što bi bilo bolje, može biti polemika nauke o odmaranju, ali u stvari je to duboko filozofsko pitanje sa dalekosežnim implikacijama. Šta je bolje —nesmotren život pun uspona i padova, ili bezbedna, ali neupadljiva egzistencija? Šta je bolje — ekonomija u kojoj svi žive u umerenom siromaštvu, ili ekonomija u kojoj neki žive u izobilju, dok su drugi veoma siromašni?

RACIONALNI AGENT I PRIMER MODERNOG RACIONALNOG AGENTA

U ovoj sekciji data je definicija racionalnog agenta.

Šta je *racionalno* u bilo kom trenutku zavisi od četiri stvari:

- Mere učinka koja definiše kriterijum uspeha.
- Agentovog prethodnog znanja o okruženju.
- Akcija koje agent može da izvrši.
- Agentove sekvence opažanja do određenog trenutka.

Ovo dovodi do definicije racionalnog agenta:
Za svaku moguću sekvencu opažanja, racionalni agent treba da izabere akciju za koju se očekuje da će maksimizirati njegovu meru učinka, s obzirom na dokaze koje pruža sekvenca opažanja i bilo koje ugrađeno znanje koje agent ima.

Slika 2.1 Primer modernog racionalnog agenta [5].

Uzmite u obzir jednostavnog agenta- usisivača iz prethodnog primera sveta usisivača koji čisti kvadrat ako je prljav i prelazi na drugi kvadrat ako nije; funkcija agenta prikazana na Slici 3. Da li je ovo racionalan agent? To zavisi! Prvo, treba da kažemo šta je mera učinka, šta je poznato o životnoj sredini i koje senzore i aktuatore agent ima. Možemo da pretpostavimo sledeće: 1) mera učinka dodeljuje jedan poen za svaki čisti kvadrat u svakom vremenskom koraku, tokom „životnog veka“ od 1000 vremenskih koraka. 2) "geografija" okruženja je unapred poznata, ali raspodela prljavštine i inicijalna lokacija agenta nisu. 3) Akcije Levo i Desno pomeraju agenta za jedan kvadrat osim u slučaju kada ovo pomera agenta van okruženja, u tom slučaju agent ostaje tamo gde je 4) Jedine dostupne akcije su Desno, Levo i Usisaj. 5) Agent korektno opaža svoju lokaciju i gde lokacija sadrži prljavštinu. Pod ovim okolnostima agent je zaista racionalan. Njegov očekivani učinak je najmanje (barem) dobar kao učinak bilo kojeg drugog agenta. Lako se može zaključiti da bi isti agent bio iracionalan u nekim dugim okolnostima. Na primer, kada se sva prljavština očisti, agent će bespotrebno oscilovati napred-nazad;ako mera učinka uključuje kaznu od jednog poena za svaki pokret, agent će loše proći. Bolji agent za ovaj slučaj ne bi radio ništa jednom kada je siguran da su svi kvadrati čisti. Ako se čisti kvadrati ponovo zaprljaju, agent treba povremeno da proverava i ponovo ih očisti ako je potrebno. Ako je geografija okruženja nepoznata, agent će morati da je istraži.

Pokazni primer modernog racionalnog agenta dat je na Slici 1 i predstavlja autonomno vozilo koje se koristi u USA u komercijalne svrhe.

SVEZNANJE, UČENJE I AUTONOMIJA

U ovoj sekciji opisani su sveznanje, učenje i autonomija.

Moramo voditi računa da razlikujemo racionalnost i sveznanje. Sveznajući agent zna stvarni ishod svojih akcija i može da deluje u skladu sa tim; ali je sveznanje nemoguće u stvarnosti. Razmotrite sledeći konceptualni primer: Šetam jednog dana Jelisejskim poljima i preko puta vidim starog prijatelja. U blizini nema saobraćaja i ništa mi ne odvlači pažnju, pa, racionalno, počinjem da prelazim ulicu. U međuvremenu, u 33.000 stopa, tovarna vrata padaju sa aviona koji prolazi, i pre nego što stignem na drugu stranu ulice, spljoštim se. Da li sam bio iracionalan da pređem ulicu? Malo je verovatno da bi moja čitulja glasila „Idiot pokušava da pređe ulicu“.

Ovaj primer pokazuje da racionalnost nije isto što i savršenstvo. Racionalnost maksimizira očekivane performanse, dok savršenstvo maksimizira stvarne performanse. Odustajanje od zahteva za savršenstvom nije samo pitanje poštenja prema agentima. Poenta je u tome da ako očekujemo da agent uradi ono što se ispostavi da je najbolja akcija posle činjenice, biće nemoguće dizajnirati agenta da ispuni ovu specifikaciju – osim ako se vremenom ne izmisle vremenske mašine koje gledaju u budućnost.
U meri u kojoj se agent oslanja na prethodno znanje svog dizajnera, a ne na sopstvene percepcije i proces učenja, kažemo da u toj meri agentu nedostaje *autonomija*. Racionalni agent treba da bude autonoman—treba da nauči šta može da nadoknadi zbog delimičnog ili pogrešnog prethodnog znanja. Na primer, agent za usisavanje koji može da nauči da predvidi gde i kada će se pojaviti dodatna prljavština će biti bolji od onog koji to ne čini. Praktično, retko je potrebna potpuna autonomija od samog početka: kada agent ima malo ili nimalo iskustva, morao bi da deluje nasumično osim ako mu dizajner ne pruži neku pomoć. Dakle, baš kao što evolucija daje životinjama dovoljno ugrađenih refleksa da prežive dovoljno dugo da same uče, bilo bi razumno pružiti veštačkom inteligentnom agentu neko početno znanje, kao i sposobnost učenja. Nakon dovoljnog iskustva iz svog okruženja, ponašanje racionalnog agenta može postati efektivno nezavisno od njegvog prethodnog znanja. Dakle, inkorporacija učenja omogućava da se dizajnira jedan racionalni agent koji će uspeti u velikom broju okruženja.

Poglavlje 3

Priroda agentovog okruženja

ODREĐIVANJE OKRUŽENJA ZADATKA

U ovoj sekciji opisano je određivanje okruženja zadatka.

U našoj raspravi o racionalnosti jednostavnog agenta usisivača, morali smo da navedemo meru performansi, okruženje, aktuatore i senzore agenta. Sve ovo grupišemo pod naslovom okruženje zadatka. Ovaj opis okruženja zadatka grupišemo pomoću akronima PEAS - Performanse, Okruženje, Aktuatori, Senzori (engl. Performance, Environment, Actuators, Sensors) opis. Tokom dizajniranja agenta, prvi korak uvek mora biti da se specificira okruženje zadatka što je moguće potpunije. Agent usisivač je bio jako jednostavan primer, tako da ćemo sada razmotriti kompleksniji primer - automatizovani taksi vozač. Na Slici 1 nalazi se PEAS opis okruženja zadatka za automatizovanog taksi vozača.

Slika 3.1 PEAS opis okruženja zadatka za automatizovani taksi [1].

Prvo pitanje koje postavljamo jeste koja je **mera učinka** koju želimo da naš automatizovani vozač postigne? Poželjno je da stigne do tačnog odredišta uz minimalnu potrošnju goriva, da minimizuje vreme putovanja, da maksimizira udobnost i bezbednost, da maksimizuje profit. Očevidno neki od ovih zahteva su konfliktni, tako da će morati da se prave kompromisi.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER: AUTOMATIZOVANI TAKSI VOZAČ

U ovoj sekciji diskutuje se o okručenju zadatka za automatizovanog taksi vozača.

Drugo pitanje koje postavljamo jeste - **kakvo je okruženje** za vožnju sa kojim će se taksi susresti? Bilo koji taksi vozač mora da se susretne sa mnoštvom puteva - od seoskih puteva, do kompleksnih auto puteva sa po 12 traka. Putevi sadrže komplikovane saobraćajnice, pešake, životinje pored puta, radove na putu, policijska kola, bare, rupe, itd. Taksi takođe mora imati interakciju sa potencijalnim i trenutnim putnicima. Taksi može da vozi u zemljama gde je sneg retko problem (Južna Kalifornija) ili na Aljasci gde je sneg veoma često problem. Može voziti desnom stranom ili biti prilagođen da vozi i levom stranom u zemljama gde se vozi levom stranom (Britanija ili Japan). Očigledno, što više ograničenja imamo u okruženju, lakše je dizajnirati problem.

Aktuatori za automatizovani taksi uključuju one aktuatore koji su dostupni i ljudskim vozačima: kontrolu motora preko gasa i kontrolu nad upravljanjem volanom i kočenjem. Pored toga biće potreban izlaz na displej ekrana ili sintisajzer glasa kako bi se razgovaralo sa putnicima, i možda neki način da se komunicira sa drugim vozilima.

Osnovni **senzori** za taksi vozača uključuju jednu ili više video kamera kako bi mogli da "vide", kao i lidar kao ultrazvučni senzor pomoću kojeg može da detektuje distancu do drugih vozača i prepreka. Da bi izbegao kazne za prebrzu vožnju, taksi mora imati merač brzine, posebno na krivinama mora imati akcelerometar. Da bi odredio mehaničko stanje vozila, mora imati standardne senzore za stanje motora, nivo goriva i električne senzore sistema. Takođe, trebaće mu GPS signal da se ne izgubi, kao i touch screen ili neki ulaz za glas putnika kako bi putnici zatražili željenu destinaciju.

Slika 3.2 Automatizovano vozilo/taksi [1].

VRSTE OKRUŽENJA ZADATKA

U ovoj sekciji dat je pregled različitih vrsta okruženja zadatka.

Na osnovu različitih tipova problema koje bi agent potencijalno mogao da reši i na osnovu različitih realnih ograničenja koja mogu biti nametnuta možemo izvršiti osnovnu podelu različitih tipova okruženja zadatka:

- *Potpuno opservabilno* (vs. *parcijalno opservabilno*): Senzori agenta omogućavaju uvid u kompletno stanje okruženja u svakom trenutku vremena.
- *Determinističko* (vs. *stohastičko*): Naredno stanje okruženja je u potpunosti određeno tekućim stanjem i akcijama koje je agent obavio. Ukoliko je okruženje determinističko, sa izuzetkom akcija drugih agenata, tada se ono naziva strategijskim.
- *Epizodno* (vs. *sekvencijalno*): Iskustvo agenta je podeljeno na epizode (svaka epizoda se sastoji od opažaja i odgovarajuće akcije), i izbor akcije u svakoj epizodi zavisi samo od tekuće epizode, a ne zavisi od prethodnih epizoda.
- *Statičko* (vs. *dinamičko*): Okruženje se ne menja u toku izbora akcije. Okruženje je semidinamičko ukoliko se okruženje ne menja u ovom intervalu vremena, ali se menja vrednost mere kvaliteta.
- *Diskretno* (vs. *kontinualno*): Konačan broj distinktnih, jasno definisanih opažaja i akcija.
- *Jedan agent* (vs. *više agenata*): Jedan agent deluje sam u datom okruženju

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

POTPUNO OPSERVABILNO OKRUŽENJE U ODNOSU NA DELIMIČNO OPSERVABILNO

U ovoj sekciji opisano je potpuno opservabilno okruženje u odnosu na delimično opservabilno okruženje.

Ako senzori agenta daju pristup kompletnomstanju njegovog okruženja u svakom trenutku, onda kažemo da je okruženje zadatka *u potpunosti opservabilno*. Okruženje zadatka je efektivno u potpunosti vidljivo ako senzori detektuju sve aspekte koji su relevantni za izbor akcije; relevantnost, zauzvrat, zavisi od mere učinka. Potpuno opservabilna okruženja su zgodna jer agent ne mora da održava nikakvo unutrašnje stanje da bi pratio svet. Okruženje može biti *delimično oservabilno* zbog bučnih i netačnih senzora ili zato što delovi stanja jednostavno nedostaju u podacima senzora — na primer, usisivač sa samo lokalnim senzorom prljavštine ne može da kaže da li ima prljavštine u drugim kvadratima, a automatizovani taksi ne vidi šta drugi vozači razmišljaju. Ako agent uopšte nema senzore, onda kažemo da je okolina *neopservabilna*.

Slika 3.3 Primer opservabilnog okruženja u odnosnu na delimično opservabilno [3].

OKRUŽENJE SA JEDNIM AGENTOM U ODNOSNU NA VIŠE AGENATA

U ovoj sekciji opisana je razlika ukoliko imamo jednog ili više agenata u okruženju.

Razlika između okruženja sa jednim i više agenata može izgledati dovoljno jednostavno. Na primer, agent koji sam rešava ukrštene reči je očigledno u okruženju jednog agenta, dok je agent koji igra šah u okruženju sa dva agenta. Međutim, postoje neki suptilni problemi. Prvo, opisali smo kako se entitet može posmatrati kao agent, ali nismo objasnili koji entiteti se moraju posmatrati kao agenti. Da li agent A (taksista na primer) mora da tretira objekat B (drugo vozilo) kao agenta, ili se može tretirati samo kao objekat koji se ponaša u skladu sa zakonima fizike, analogno talasima na plaži ili lišću koje duva na vetru? Ključna razlika je u tome da li se ponašanje agenta Bnajbolje opisuje kao maksimiziranje mere učinka čija vrednost zavisi od ponašanja agenta A. Na primer, u šahu, protivnički entitet B pokušava da maksimizira svoju meru učinka, što, po pravilima šaha, minimizira meru učinka agenta A. Dakle, šah je takmičarsko multiagentsko okruženje. U okruženju za vožnju taksija, s druge strane, izbegavanje sudara maksimizira meru performansi svih agenata, tako da je to *delimično kooperativno okruženje agenta*. Delimično je i *konkurentno* jer, na primer, samo jedan automobil može da zauzme parking mesto. Problemi dizajna agenata u okruženjima sa više agenata su često prilično različiti od onih u okruženjima sa jednim agentom; na primer, komunikacija se često pojavljuje kao racionalno ponašanje u multiagentskim okruženjima; u nekim konkurentskim okruženjima, ***randomizovano ponašanje*** je racionalno jer izbegava zamke predvidljivosti.

Slika 3.4 Jedan agent u odnosu na više agenata [3].

DETERMINISTIČKO U ODNOSU NA STOHASTIČKO OKRUŽENJE

U ovoj sekciji objašnjena je razlika između determinističkog i stohastičkog agenta.

Ako je sledeće stanje okruženja u potpunosti određeno trenutnim stanjem i radnjom koju je izvršio agent, onda kažemo da je okruženje *determinističko;* inače je *stohastičko*. U principu, agent ne mora da brine o **neizvesnosti** u potpuno vidljivom, determinističkom okruženju. U našoj definiciji, ignorišemo neizvesnost koja proizilazi isključivo iz akcija drugih agenata u okruženju sa više agenata: na primer, igra može biti deterministička iako svaki agent možda nije u stanju da predvidi akcije drugih.

Ako je okruženje delimično opservabilno , može se učiniti da je stohastičko. Većina stvarnih situacija je toliko složena da je nemoguće pratiti sve neopažene aspekte; u praktične svrhe, moraju se tretirati kao stohastičke. Taksi vožnja je u tom smislu jasno stohastična, jer se nikada ne može tačno predvideti ponašanje saobraćaja. Na primer, nečije gume eksplodiraju i motor se blokira bez upozorenja. Svet usisivača kako smo ga opisali je deterministički, ali varijacije mogu uključivati stohastičke elemente kao što su nasumično pojavljujuća prljavština ili nepouzdan mehanizam usisavanja. Kažemo da je *okruženje neizvesno* ako nije u potpunosti opservabilno ili nije determinističko. Jedna poslednja napomena: naša upotreba reči „stohastički“ generalno implicira da je neizvesnost u pogledu ishoda kvantifikovana u smislu verovatnoće ("šansa da će sutra padati kiša je 25%"); nedeterminističko okruženje je ono u kojem se akcije karakterišu njihovim mogućim ishodima, ali im se ne vezuju nikakve verovatnoće ("postoji šansa da će sutra padati kiša") . Nedeterministički opisi okruženja obično su povezani sa merama performansi koje zahtevaju od agenta da uspe za sve moguće ishode svojih akcija.

Slika 3.5 Deterministički u odnosu na stohastičkog agenta [3].

EPIZODNO OKRUŽENJE ZADATKA U ODNOSU NA SEKVENCIJALNO

U ovoj sekciji objašnjena je razlika između epizodnog zadatka okruženja u odnosu na sekvencijalni.

U epizodnom okruženju zadatka, iskustvo agenta je podeljeno na *atomske epizode*. U svakoj epizodi agent prima percepciju i zatim izvodi jednu akciju. Ono što je najvažnije, sledeća epizoda ne zavisi od radnji preduzetih u prethodnim epizodama. Mnogi klasifikacioni zadaci su epizodni. Na primer, agent koji mora da uoči neispravne delove na montažnoj traci zasniva svaku odluku na trenutnom delu, bez obzira na prethodne odluke; štaviše, sadašnja odluka ne utiče na to da li je sledeći deo neispravan. U sekvencijalnim okruženjima, s druge strane, trenutna odluka bi mogla da utiče na sve buduće odluke. Igra šaha i vožnja taksija su sekvencijalni zadaci: u oba slučaja, kratkoročne akcije mogu imati dugoročne posledice.
Epizodna okruženja su mnogo jednostavnija od *sekvencijalnih okruženja* jer agent ne mora da razmišlja unapred.

Slika 3.6 Epizodno u odnosu na sekvencijalno [3].

STATIČKO U ODNOSU NA DINAMIČKO OKRUŽENJE

U ovoj sekciji objašnjena je razlika između statičkog i dinamičkog okruženja.

Ako okruženje može da se promeni dok agent razmatra, onda kažemo da je okruženje *dinamičko* za tog agenta; inače je *statičko*. Sa statičnim okruženjima je lako izaći na kraj jer agent ne mora stalno da gleda u svet dok odlučuje o akciji, niti treba da brine o protoku vremena. S druge strane, dinamična okruženja stalno pitaju agenta šta želi da uradi; ako agent još nije odlučio, to se računa kao odluka da ništa ne uradi. Ako se sama okolina ne menja tokom vremena, ali se rezultat učinka agenta menja, onda kažemo da je okruženje *poludinamično*. Vožnja taksijem je jasno dinamična: ostali automobili i sam taksi nastavljaju da se kreću dok se algoritam vožnje muči o tome šta dalje. Šah, kada se igra sa satom, je poludinamičan. Ukrštene reči su statične.

Slika 3.7 Statičko okruženje u odnosu na dinamičko [3].

DISKRETNO OKRUŽENJE U ODNOSU NA KONTINUALNO

U ovoj sekciji objašnjena je razlika između diskretnog i kontinualnog okruženja.

Razlika *diskretno*/ *kontinuirano* se primenjuje na stanje okoline, na način na koji se upravlja vremenom i na percepcije i akcije agenta. Na primer, šahovsko okruženje ima konačan broj različitih stanja (isključujući sat), šah takođe ima diskretan skup opažaja i akcija. Vožnja taksija je problem kontinualnog stanja i kontinualnog vremena: brzina i lokacija taksija i drugih vozila prelaze niz kontinualnih vrednosti i to rade glatko tokom vremena. Akcije taksi-vožnje su takođe kontinuirane (uglovi upravljanja itd.). Unos digitalnih kamera je diskretan, strogo govoreći, ali se obično tretira kao da predstavlja kontinuirano promenljiv intenzitet i lokacije.

Slika 3.8 Diskretno u odnosu na kontinualno [3].

POZNATO OKRUŽENJE U ODNOSU NA NEPOZNATO

U ovoj sekciji objašnjena je razlika između poznatog i nepoznatog okruženja.

Strogo govoreći, ova razlika se ne odnosi na samo okruženje, već na agentovo ili dizajnerovo stanje znanja o „zakonima fizike“ okoline. U poznatom okruženju, dati su ishodi (ili verovatnoće ishoda ako je okruženje stohastičko) za sve akcije. Očigledno, ako je okruženje nepoznato, agent će morati da nauči kako ono funkcioniše da bi *doneo dobre odluke*. Imajte na umu da razlika između poznatih i nepoznatih okruženja nije ista kao razlika između

potpuno i delimično opservabilnih okruženja. Sasvim je moguće da poznato okruženje bude delimično opservabilno — na primer, u kartaškim igrama pasijansa, znamo pravila, ali igrač ne može da vidi protivničke karte koje još nisu okrenute. Suprotno tome, nepoznato okruženje može biti u potpunosti opservabilno—u novoj video igri ekran može da prikaže celo stanje igre, ali još uvek ne znamo šta dugmad rade dok ih ne isprobamo.

Slika 3.9 Poznato okruženje u odnosu na nepoznato [3].

Najteži slučaj jeste kada je okruženje parcijalno opservabilno, multiagentsko, nederminističko, sekvencijalno, dinamičko, kontinualno i nepoznato. Vožnja taksija je teška u svakom pogledu i u svim ovim slučajevima, osim što je vozačevo okruženje uglavnom poznato.

Poglavlje 4

Struktura agenata

UVOD U STRUKTURU AGENATA I VRSTE AGENATA

U ovoj sekciji dat je uvod u strukturu agenata i pregled vrsta agenata.

Do sada smo govorili o agentima opisujući ponašanje — akciju koja se izvodi nakon bilo koje date sekvence opažanja. Sada ćemo objasniti kako unutrašnjost agenta funkcioniše. Zadatak VI je da dizajnira *program agenta* koji implementira funkciju agenta –preslikavajući percepcije u akcije. Pretpostavljamo da će ovaj program raditi na nekoj vrsti računarskog uređaja sa fizičkim senzorima i aktuatorima - to nazivamo *arhitekturom agenta*:

agent = arhitektura + program
Očigledno, program koji izaberemo mora biti onaj koji je prikladan za arhitekturu. Ako će program preporučiti akciju kao što je *Šetati*, bolje bi bilo da arhitektura ima noge. Arhitektura može biti samo običan računar, ili može biti robotski automobil sa nekoliko kompjutera, kamera i drugih senzora. Uopšteno govoreći, arhitektura čini da percepcije sa senzora budu dostupne programu, pokreće program i prenosi izbore akcija programa aktuatorima dok se generišu.
Postoje četiri osnovne vrste agentskih programa koji sadrže principe koji leže u osnovi skoro svih inteligentnih sistema:

- *Jednostavni refleksni agenti*
- *Refleksni agenti zasnovani na modelu*
- *Agenti zasnovani na ciljevima*
- *Agenti zasnovani na korisnosti*

Uopšteno govoreći, svi ovi agenti se mogu pretvoriti u *obučavajuće agente* kako bi poboljšali performanse svojih komponenti, i samim tim generisali bolje akcije.

JEDNOSTAVNI REFLEKSNI AGENTI

U ovoj sekciji opisani su jednostavni refleksni agenti.

Najjednostavnija vrsta agenata su jednostavni refleksni agenti. Ovi agenti biraju akcije na osnovu trenutnog opažanja, ignorišući ostatak istorije opažanja. Na primer, agent usisivač čija je funkcija opisana na Slici 3 u drugom objektu učenja je jednostavan refleksni agent zato što je njegova odluka zasnovana samo na osnovu trenutne lokacije i na osnovu toga da li na toj lokaciji ima prljavštine. Istorija opažanja je istorija svega što je agent uočio do danas. Funkcija agenta preslikava bilo koju sekvencu opažanja u akciju. Funkcija agenta kod jednostavnog refleksnog agenta je zasnovana na pravilu *uslov-delovanje*. Na primer, kod taksi vozača

ukoliko vozilo ispred koči (uslov), inicira se kočenje taksija (akcija). Ovakva funkcija agenta uspeva samo kada je okruženje u potpunosti opservabilno. Za jednostavne refleksne agente koji rade u delimično opservabilnim okruženjima, beskonačne petlje su često neizbežne. Pretpostavimo da je jednostavan refleksni agent usisivač lišen svog senzora lokacije i da ima samo senzor prljavštine. Takav agent ima samo dva moguća opažanja: [Prljavo] i [Čisto]. Može Usisati kao odgovor na [Prljavo], ali da li to treba da uradi kao odgovor na [Čisto]? Pomeranje Levo ne uspeva (zauvek) ako se desi da počne u kvadratu A, a kretanje Desno ne uspeva (zauvek) ako se desi da počne u kvadratu B. Možda je moguće pobeći iz beskonačnih petlji ako agent može **nasumično** da podesi svoje akcije. Na primer, ako usisivač primeti [Čisto], mogao bi baciti novčić da izabere između Desno i Levo. Lako je pokazati da će agent doći do drugog kvadrata u proseku za dva koraka.

Zatim, ako je taj kvadrat prljav, agent će ga očistiti i zadatak će biti završen. Dakle, randomizovani jednostavni refleksni agent može nadmašiti determinističkog jednostavnog refleksnog agenta.Jednostavni refleksni agenti su jednostavni, ali su zato vrlo ograničene inteligencije. Uglavnom rade dobro ako se korektna odluka može doneti na osnovu trenutnog opažanja, odnosno ako je okruženje potpuno opservabilno.

Slika 4.1 Jednostavni refleksni agenti [1].

REFLEKSNI AGENTI ZASNOVANI NA MODELU. PRIMER REFLEKSNOG AGENTA.

U ovoj sekciji opisani su refleksni agenti zasnovani na modelu.

Najefikasniji agent da se agent snađe u delimično opservabilnom okruženju jeste da se osposobi da prati deo okruženja koji ne može da vidi. Prema tome, agent bi trebalo da održi neku vrstu internog stanja koje zavisi od istorije opažanja i na taj način odražava neke od neopserviranih aspekata trenutnog stanja. Što se tiče problema kočenja, unutrašnje stanje nije previše opsežno – samo prethodni kadar sa kamere, omogućavajući agentu da otkrije kada se dva crvena svetla na
ivici vozila pale ili gase istovremeno. Za druge zadatke vožnje kao što je promena trake, agent treba da prati gde su ostali automobili ako ne može da ih vidi sve odjednom. Ažuriranje informacije o unutrašnjem stanju tokom vremena zahteva dve vrste znanja koje moraju biti enkodirane u agentovom programu u nekoj formi. Prvo, treba nam informacija o tome kako se sve menja tokom vremena što se može grubo podeliti na dva dela: na efekte agentovih akcija i na to kako svet evoluira nezavisno od agenta. Ovo znanje o tome kako svet funkcioniše se naziva *model prelaza sveta*. Drugo, potrebne su nam informacije o tome kako se stanje sveta odražava na agentovu percepciju. Ova vrsta znanja se naziva *model senzora*. Zajedno, model tranzicije i model senzora omogućavaju agentu da prati stanje sveta – u meri u kojoj je to moguće s obzirom na ograničenja agentovih senzora. Agenti koji koriste takve modele nazivaju se agenti zasnovani na modelu.

Pokazni primer refleksnog agenta dat je u youtube snimku.

Slika 4.2 Refleksni agenti zasnovani na modelu [1].

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

AGENTI ZASNOVANI NA CILJEVIMA

U ovoj sekciji opisani su agenti zasnovani na ciljevima.

Agentovo znanje o trenutnom stanju okruženja nekada nije dovoljno da bi odlučio šta da radi. Na primer, na raskrsnici, taksi može skrenuti levo, desno, ili može ići pravo. Korektna odluka zavisi od toga gde taksi pokušava da stigne. Drugim rečima, osim što agentu treba opis trenutnog stanja, takođe mu i treba neka vrsta informacije o cilju koja opisuje situacije koje su poželjne - na primer da bude na određenoj destinaciji. Program agenta može da kombinuje ovo sa modelom (ista informacija koja je korišćena kod refleksnog agenta zasnovanog na modelu) da izabere akcije kako bi stigao do cilja.

Agenti zasnovani na ciljevima donose odluke na osnovu toga koliko su trenutno daleko od svog cilja (opisa poželjnih situacija). Svaka njihova akcija ima za cilj smanjenje njene udaljenosti od cilja. Ovo omogućava agentu način da bira između više mogućnosti, birajući onu koja dostiže ciljno stanje. Znanje koje podržava njegove odluke je eksplicitno predstavljeno i može se modifikovati, što čini ove agente fleksibilnijim.

Ukoliko zadovoljenje cilja direktno sledi iz samo jedne akcije, onda je izbor akcije zasnovan na cilju prilično jednostavan. Međutim, ako se zadovoljenje cilja postiže iz više koraka, onda se razmatraju skupovi akcija kako bi se našao način da se postavljeni cilj zadovolji. Ovo se svodi na probleme pretraživanja i planiranja, o kojima će biti reči u narednim lekcijama.

Ponašanje agenta zasnovanog na ciljevima može se lako promeniti - na primer, agentovo ponašanje lako se može promeniti da ode na drugu destinaciju, jednostavnim navođenjem tog odredišta kao cilja.

Slika 4.3 Agenti zasnovani na ciljevima [1].

AGENTI ZASNOVANI NA KORISNOSTI

U ovoj sekciji opisani su agenti zasnovani na korisnosti.

Ciljevi sami po sebi nisu dovoljni sa obezbede visoko kvalitetno ponašanje u većini okruženja. Ciljevi nam obezbeđuju grubu binarnu razliku između dva stanja: "srećno" (postignut cilj) i "nesrećno" (cilj nije postignut). Na primer, mnoge sekvence akcija će dovesti taksi do cilja, ali neke vožnje će biti brže, neke bezbednije, neke jeftinije, itd. Već smo pričali o tome da mera učinka dodeljuje poen svakoj sekvenci stanja okruženja, tako da se jasno može napraviti razlika između manje ili više poželjnih načina da se taksi dovede na odredište. Pošto "srećno" stanje nije naučni izraz, uveden je pojam *korisnosti*. Agentova *funkcija korisnosti* je zapravo internalizacija mere učinka. Pod uslovom da se interna funkcija korisnosti i eksterna mera učinka slažu, agent koji bira akcije da maksimizira svoju korisnost će biti *racionalan* u skladu sa eksternom merom učinka. U dve vrste slučajeva, ukoliko su ciljevi neadekvatni, agent zasnovan na korisnosti i dalje može donositi racionalne odluke. Prvo, kada postoje

suprotstavljeni ciljevi, od kojih se samo neki mogu postići (na primer, brzina vožnje i bezbednost), funkcija korisnosti određuje odgovarajući kompromis. Drugo, kada postoji nekoliko ciljeva kojima agent može težiti, od kojih se nijedan ne može postići sa sigurnošću, korisnost obezbeđuje način na koji verovatnoći uspeha mogu dodeliti težinski faktori u zavisnosti od važnosti ciljeva.

Slika 4.4 Agenti zasnovani na korisnosti [1].

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

OBUČAVAJUĆI AGENTI I PRIMER MODERNOG OBUČAVAJUĆEG AGENTA

U ovoj sekciji opisani su obučavajući agenti.

Obučavajući agent u VI je tip agenta koji može da uči iz svojih prošlih iskustava ili ima sposobnosti učenja. Počinje da deluje sa osnovnim znanjem, a zatim je u stanju da deluje i prilagođava se automatski kroz učenje. Obučavajući agent ima uglavnom četiri konceptualne komponente, a to su:

- *Element obučavanja*: Odgovoran je za napredovanje u učenju.
- *Kritičar*: Element obučavanja uzima povratne informacije od kritičara koje opisuju koliko dobro agent radi u odnosu na fiksni standard učinka.
- *Element performansi*: On je odgovoran za odabir spoljašnje akcije. Ranije smo za njega smatrali da je ceo agent (prihvata opažaje i deluje na spoljno okruženje).
- *Generator problema*: Ova komponenta je odgovorna za predlaganje radnji koje će dovesti do novih i informativnih iskustava.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Slika 4.5 Obučavajući agenti [1].

Slika 4.6 Primer obučavajućeg agenta da igra šah [7].

Poglavlje 5

Primena inteligentnih agenata

PRIMERI PRIMENE INTELIGENTIH AGENATA

U ovoj sekciji dati su primeri primene inteligentih agenata.

Inteligentni agenti u VI se primenjuju u mnogim situacijama iz stvarnog života i u realnim problemima. Daćemo pregled nekih najvažnijih primena inteligentnih agenata:

- **Pretraga, pronalaženje i navigacija informacija**: Inteligentni agenti poboljšavaju pristup i navigaciju informacijama. Ovo se postiže pretragom informacija pomoću pretraživača. Internet se sastoji od mnogih objekata podataka koji korisnicima mogu oduzeti mnogo vremena da traže određeni objekat podataka. Inteligentni agenti obavljaju ovaj zadatak u ime korisnika za kratko vreme.
- **Ponavljajuće kancelarijske aktivnosti**: Neke kompanije su automatizovale određene administrativne zadatke kako bi smanjile operativne troškove. Neke od funkcionalnih oblasti koje su automatizovane uključuju korisničku podršku i prodaju. Inteligentni agenti su takođe korišćeni za poboljšanje kancelarijske produktivnosti.
- Medicinska dijagnoza: Inteligentni agenti su takođe primenjeni u zdravstvenim uslugama za poboljšanje zdravlja pacijenata. U ovom slučaju, pacijent se smatra okruženjem. Kompjuterska tastatura se koristi kao senzor koji prima podatke o simptomima pacijenta. Inteligentni agent koristi ove informacije da odluči o najboljem postupku. Medicinska nega se pruža kroz aktuatore kao što su testovi i tretmani.

- **Usisavanje**: VI agenti se takođe koriste za poboljšanje efikasnosti i čistoće usisavanja. U ovom slučaju, okruženje može biti soba, sto ili tepih. Neki od senzora koji se koriste za usisavanje uključuju kamere, senzore za udarce i senzore za detekciju prljavštine. Akciju pokreću aktuatori kao što su četke, točkovi i usisivači.
  Autonomna vožnja: Inteligentni agenti poboljšavaju rad autonomnih automobila. U autonomnoj vožnji, različiti senzori se koriste za prikupljanje informacija iz okoline. To uključuje kamere, GPS i radar. U ovoj aplikaciji, okruženje mogu biti pešaci, druga vozila, putevi ili putokazi. Za pokretanje akcija koriste se različiti aktuatori. Na primer, kočnice se koriste da se automobil zaustavi.

Slika 5.1 Primer modernog inteligentnog agenta [6].

Poglavlje 6

Pokazne vežbe

UPOZNAVANJE SA OPENAI GYM OKRUŽENJEM

U ovoj vežbi upoznajemo se sa OpenAI Gym okruženjem.

Zadatak 1 (45 min.)

U ovom zadatku upozaćemo se sa OpenAI Gym okruženjem. Biće pokazano uvoženje i testiranje osnovnih paketa, istražićemo osnovne karakteristike OpenAI Gym-a. Dat je osnovni kod učenja sa pojačanjem OpenAI Gym, kao i istraživanje dostupnog okruženja OpenAI Gym. Source code korišćen u ovom primeru dat je u referenci [4]: https://github.com/PacktPublishing/Hands-On-Intelligent-Agents-with-OpenAI-Gym.

```
#!/usr/bin/env python
# Script to test if torch installation is successful | Praveen Palanisamy
# Chapter 3, Hands-on Intelligent Agents with OpenAI Gym, 2018

import torch
t = torch.Tensor(3,3)
print(t)

#!/usr/bin/env python
# Simple script to test a box2d environment | Praveen Palanisamy
# Chapter 3, Hands-on Intelligent Agents with OpenAI Gym, 2018

import gym
env = gym.make('BipedalWalker-v2')
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample())

#!/usr/bin/env python
# Handy script for exploring Gym environment's spaces | Praveen Palanisamy
# Chapter 4, Hands-on Intelligent Agents with OpenAI Gym, 2018

import sys
import gym
from gym.spaces import *

def print_spaces(space):
    print(space)
    if isinstance(space, Box): # Print lower and upper bound if it's a Box space
        print("\n space.low: ", space.low)
```

```
        print("\n space.high: ", space.high)


if __name__ == "__main__":
    env = gym.make(sys.argv[1])
    print("Observation Space:")
    print_spaces(env.observation_space)
    print("Action Space:")
    print_spaces(env.action_space)
    try:
        print("Action description/meaning:",env.unwrapped.get_action_meanings())
    except AttributeError:
        pass

#!/usr/bin/env python
# Handy script for listing all available Gym environments | Praveen Palanisamy
# Chapter 4, Hands-on Intelligent Agents with OpenAI Gym, 2018

from gym import envs
env_names = [spec.id for spec in envs.registry.all()]
for name in sorted(env_names):
    print(name)


# Boilerplate code for Reinforcement Learning with Gym | Praveen Palanisamy
# Chapter 4, Hands-on Intelligent Agents with OpenAI Gym, 2018

import gym
env = gym.make("Qbert-v0")
MAX_NUM_EPISODES = 10
MAX_STEPS_PER_EPISODE = 500
for episode in range(MAX_NUM_EPISODES):
    obs = env.reset()
    for step in range(MAX_STEPS_PER_EPISODE):
        env.render()
        action = env.action_space.sample()# Sample random action. This will be
replaced by our agent's action when we start developing the agent algorithms
        next_state, reward, done, info = env.step(action) # Send the action to the
environment and receive the next_state, reward and whether done or not
        obs = next_state

        if done is True:
            print("\n Episode #{} ended in {} steps.".format(episode, step+1))
            break

#!/usr/bin/env python
# Handy script for exploring the available Gym environments | Praveen Palanisamy
# Chapter 4, Hands-on Intelligent Agents with OpenAI Gym, 2018

import gym
import sys

def run_gym_env(argv):
```

```
    env = gym.make(argv[1]) # Name of the environment supplied as 1st argument
    env.reset()
    for _ in range(int(argv[2])): # Number of steps to be run supplied as 2nd
argument
        env.render()
        env.step(env.action_space.sample())
    env.close()

if __name__ == "__main__":
    run_gym_env(sys.argv)
```

PROBLEM PLANINSKOG AUTOMOBILA

U ovoj vežbi dat je primer obučavajućeg agenta na problemu planinskog automobila.

Zadatak 2 (45 min.)

U ovom zadatku biće pokazano rešenje čuvenog problema planinskog automobila pomoću obučavajućeg agenta. Problem je sledeći: automobil se nalazi u podnožju, između dva brda. Cilj je da automobil stigne do vrha brda sa desne strane. Automobil nema dovoljno jak motor da se ubrza i popne na vrh brda samo ubrzavajući i idući napred ka desnom brdu. Kako bi dostigao cilj, auto mora da ide napred - nazad (da se "ljulja" levo desno) i da postigne momentum tako da se u jednom trenutku dovoljno ubrza i popne na vrh desnog brda. Okruženje je prilično jednostavno i potrebno je da znamo samo dve stvari o automobilu u svakom stanju, a to su njegov položaj i njegova brzina. Problem je dvodimenzionalan, kao i njegov prostor stanja (prostor sa svim mogućim pozicijama i brzinama). Auto ima samo 3 moguće akcije u svakom stanju, može ili da ubrza napred, ubrza unazad, ili da ne uradi ništa. Svaki put kada agent preduzme akciju, okruženje će vratiti novo stanje (poziciju i brzinu). Dakle, uzmimo primer gde automobil počinje u sredini, njegov položaj može biti 0, a njegova brzina 0. To je njegovo početno stanje. Agent bi tada mogao odlučiti da ubrza napred i daće tu akciju okruženju. Okruženje će vratiti novo stanje, recimo poziciju 1, brzinu 1. Cilj našeg modela je da nauči agenta koje radnje da preduzme u svakom stanju da mu pomogne da stigne do cilja (vrha brda sa desne strane). Soruce code korišćen u ovom primeru dat je u referenci [4]: https://github.com/PacktPublishing/Hands-On-Intelligent-Agents-with-OpenAI-Gym.

```
#!/usr/bin/env/ python
"""
q_learner.py
An easy-to-follow script to train, test and evaluate a Q-learning agent on the
Mountain Car
problem using the OpenAI Gym. |Praveen Palanisamy
# Chapter 5, Hands-on Intelligent Agents with OpenAI Gym, 2018
"""
import gym
import numpy as np
```

```
MAX_NUM_EPISODES = 50000
STEPS_PER_EPISODE = 200 #  This is specific to MountainCar. May change with env
EPSILON_MIN = 0.005
max_num_steps = MAX_NUM_EPISODES * STEPS_PER_EPISODE
EPSILON_DECAY = 500 * EPSILON_MIN / max_num_steps
ALPHA = 0.05  # Learning rate
GAMMA = 0.98  # Discount factor
NUM_DISCRETE_BINS = 30  # Number of bins to Discretize each observation dim


class Q_Learner(object):
    def __init__(self, env):
        self.obs_shape = env.observation_space.shape
        self.obs_high = env.observation_space.high
        self.obs_low = env.observation_space.low
        self.obs_bins = NUM_DISCRETE_BINS  # Number of bins to Discretize each
observation dim
        self.bin_width = (self.obs_high - self.obs_low) / self.obs_bins
        self.action_shape = env.action_space.n
        # Create a multi-dimensional array (aka. Table) to represent the
        # Q-values
        self.Q = np.zeros((self.obs_bins + 1, self.obs_bins + 1,
                           self.action_shape))  # (51 x 51 x 3)
        self.alpha = ALPHA  # Learning rate
        self.gamma = GAMMA  # Discount factor
        self.epsilon = 1.0

    def discretize(self, obs):
        return tuple(((obs - self.obs_low) / self.bin_width).astype(int))

    def get_action(self, obs):
        discretized_obs = self.discretize(obs)
        # Epsilon-Greedy action selection
        if self.epsilon > EPSILON_MIN:
            self.epsilon -= EPSILON_DECAY
        if np.random.random() > self.epsilon:
            return np.argmax(self.Q[discretized_obs])
        else:  # Choose a random action
            return np.random.choice([a for a in range(self.action_shape)])

    def learn(self, obs, action, reward, next_obs):
        discretized_obs = self.discretize(obs)
        discretized_next_obs = self.discretize(next_obs)
        td_target = reward + self.gamma * np.max(self.Q[discretized_next_obs])
        td_error = td_target - self.Q[discretized_obs][action]
        self.Q[discretized_obs][action] += self.alpha * td_error

def train(agent, env):
    best_reward = -float('inf')
    for episode in range(MAX_NUM_EPISODES):
        done = False
        obs = env.reset()
        total_reward = 0.0
```

```
        while not done:
            action = agent.get_action(obs)
            next_obs, reward, done, info = env.step(action)
            agent.learn(obs, action, reward, next_obs)
            obs = next_obs
            total_reward += reward
        if total_reward > best_reward:
            best_reward = total_reward
        print("Episode#:{} reward:{} best_reward:{} eps:{}".format(episode,
                                    total_reward, best_reward, agent.epsilon))
    # Return the trained policy
    return np.argmax(agent.Q, axis=2)


def test(agent, env, policy):
    done = False
    obs = env.reset()
    total_reward = 0.0
    while not done:
        action = policy[agent.discretize(obs)]
        next_obs, reward, done, info = env.step(action)
        obs = next_obs
        total_reward += reward
    return total_reward


if __name__ == "__main__":
    env = gym.make('MountainCar-v0')
    agent = Q_Learner(env)
    learned_policy = train(agent, env)
    # Use the Gym Monitor wrapper to evalaute the agent and record video
    gym_monitor_path = "./gym_monitor_output"
    env = gym.wrappers.Monitor(env, gym_monitor_path, force=True)
    for _ in range(1000):
        test(agent, env, learned_policy)
    env.close()
```

Poglavlje 7

Domaći zadatk

DOMAĆI ZADATAK #2

U ovoj sekciji dat je Domaći zadatak #2.

Domaći poslati u poruci sa nazivom CS360-DZ02. Arhiva projekta (programa) koji se šalje treba da ima naziv CS360-DZ02-Indeks-ImePrezime. Pored rešenja treba poslati i dokument sa kratkim opisom (algoritma), kao i sa skrinšotovima pokrenutog programa.
Rešenje domaćeg poslati na adresu: lazar.mrkela@metropolitan.ac.rs
Studenti biraju jedan zadatak Zadatak se bira na osnovu formule: broj_indeksa % 4 + 1.
Rešiti jedan od problema pomoću Q-learning algoritma i Gym biblioteke.

1. Acrobot: https://www.gymlibrary.dev/environments/classic_control/acrobot/ **(120 min.)**
2. Cart pole: https://www.gymlibrary.dev/environments/classic_control/cart_pole/ **(120 min.)**
3. Pendulum: https://www.gymlibrary.dev/environments/classic_control/pendulum/ **(120 min.)**
4. Lunar lender: https://www.gymlibrary.dev/environments/box2d/lunar_lander/ **(120 min.)**

U slučaju da student ne zna da primeni Q-learning na zadati problem, može da pokuša rešavanje slučajnim izborom akcija i definisanjem pravila (na primer, ako je ugao veći od neke vrednosti pomeri agenta levo, inače desno i slično u zavisnosti od problema) za manji broj poena.

Poglavlje 8

Zaključak

ZAKLJUČAK

U ovoj sekciji dat je rezime lekcije.

- U ovoj lekciji detaljno smo prošli kroz osnovne koncepte inteligentnih agenata.
- Upoznali smo se sa pojmom racionalnosti i definicijom racionalnog agenta
- Detaljno su objašnjene različite vrste okruženja zadatka
- Takođe, upoznali smo se strukturom agenata i svim vrstama agenata
- Na kraju, pokazana je primena inteligentnih agenata u problemima iz realnog sveta.

Nakon ove lekcije studenti će biti osposobljeni da razumeju osnovne koncepte inteligentnih agenata, pojam racionalnosti, vrste okruženja zadatka, kao i različite strukture agenata. Takođe, moći će da reše jednostavnije probleme sa inteligentnim agentima u programskom jeziku Pajton.

REFERENCE

U ovoj sekciji date su korišćene reference.

1. Artificial Intelligence: A Modern Approach. 4th Edition, S. Russell and P. Norvig. Prentice Hall, 2021.
2. Artificial Intelligence: A New Synthesis, Nils Nilsson, Morgan Kaufmann, 1998

3. https://slidetodoc.com/rational-agents-chapter-2-agents-an-agent-is/

4. Hands-on Intelligent Agents with OpenAI Gym (HOIAWOG), https://github.com/PacktPublishing/Hands-On-Intelligent-Agents-with-OpenAI-Gym

5. https://www.dezeen.com/2016/02/12/google-self-driving-car-artficial-intelligence-system-recognised-as-driver-usa/

6. https://www.rmigo.com/build-a-smart-vacuum-cleaning-robot-with-arduino/

7. https://towardsdatascience.com/what-can-agents-learn-through-self-play-37adb3f3581b