Strojno učenje na temelju logike i vjerojatnosti

Zovak, Trpimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij računarstvo

STROJNO UČENJE NA TEMELJU LOGIKE I VJEROJATNOSTI

Završni rad

Trpimir Zovak

Osijek, 2017.

SADRŽAJ

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 12.07.2017.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Trpimir Zovak
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3720, 23.07.2014.
OIB studenta:	91324108822
Mentor:	Doc.dr.sc. Tomislav Rudec
Sumentor:	Ivana Hartmann-Tolić
Sumentor iz tvrtke:	
Naslov završnog rada:	Strojno učenje na temelju logike i vjerojatnosti
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	12.07.2017.
Datum potvrde ocjene Odbora:	17.07.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 17.07.2017.

Ime i prezime studenta:	Trpimir Zovak
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3720, 23.07.2014.
Ephorus podudaranje [%]:	4 %

Ovom izjavom izjavljujem da je rad pod nazivom**: Strojno učenje na temelju logike i vjerojatnosti**

izrađen pod vodstvom mentora Doc.dr.sc. Tomislav Rudec

i sumentora Ivana Hartmann-Tolić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

I. UVOD

Strojno učenje je područje koje svakim danom raste i napreduje kao i njegova primjena. Ovaj rad daje pojašnjenje što je strojno učenje i razlučuje rješavanje problema pomoću dizajna od rješavanja problema uz pomoć učenja. Također, pokrivena su i područja strojnog učenja kao i izabrani algoritmi koji se koriste u rješavanju problema strojnog učenja. U cilju što lakšeg i boljeg razumijevanja teorije strojnog učenja pri uvođenju novih pojmova korišteni su primjeri iz stvarnog života.

Pravi cilj ovog rada je pokazati što je potrebno kako bi se znanje iz područja strojnog učenja efikasno primIjenilo u modeliranju poker agenta, kao i dosadašnje primjene u svijetu zajedno s načinima i rezultatima tih primjena. Igre kao što su šah i poker su idealne za testiranje umjetne inteligencije jer imaju dobro definirana pravila kao i cilj same igre. Naime, dosad je korištenje strojeva u području igara bilo koncentrirano na šah dok je poker bio u sjeni. Za razliku od šaha, poker je igra skrivenih informacija i iz tog razloga umjetna inteligencija zasad nije uspjela pronaći rješenje same igre, tj. napraviti stroj koji će igrati bolje od najboljih svjetskih igrača. U radu su kratko pojašnjena pravila same igre kako bi razumijevanje rada bilo moguće i onima koji ne znaju pravila pokera. Iako su pravila jednostavna poker je strateški kompleksna igra. Uz pravila, u trećem poglavlju su objašnjeni glavni koncepti pokera koji su ključni za modeliranje poker agenta. Zaključno, četvrto poglavlje pokazuje dosad ostvarenu primjenu umjetne inteligenciju u pokeru, način primjene i uspješnost primjene.

1.1. Zadatak završnog rada

U radu će se izložiti osnovne metode iz područja strojnog učenja te primijeniti na rješavanje problema oblikovanja igrača za *Texas Hold'em* poker.

II. ŠTO JE STROJNO UČENJE?

„Područje proučavanja koje omogućuje računalima učenje bez da budu programirana eksplicitno za to” - Arthur Samuel, 1959.

„Kažemo da program uči na temelju iskustva E uzimajući u obzir skup zadataka T i mjeru performanse P, ako se njegova performansa zadataka T, mjerena s P, poboljšava s iskustvom E.” - Tom Mitchell, Carnegie Mellon University, 1998.

Osnovna razlika između čovjeka i stroja je ta što čovjek uči na temelju iskustva, dok se stroju mora reći što treba raditi, tj. stroj slijedi upute. Može li i stroj učiti na temelju iskustva? Odgovor je da, i točno time se bavi strojno učenje. Iskustvo na temelju kojega strojevi uče predstavljaju nekakve „podatke”.

Ako djetetu od tri godine pokažete sliku i pitate ga postoji li na njoj oblak, vjerojatno ćete dobiti točan odgovor. Ako pitate čovjeka od 33 godine što je oblak, vjerojatno ćete dobiti neuvjerljiv odgovor. Nismo naučili što je oblak tako što smo ga znanstveno definirali, naučili smo što je oblak gledajući ga. Drugim riječima, naučili smo iz „podataka”.

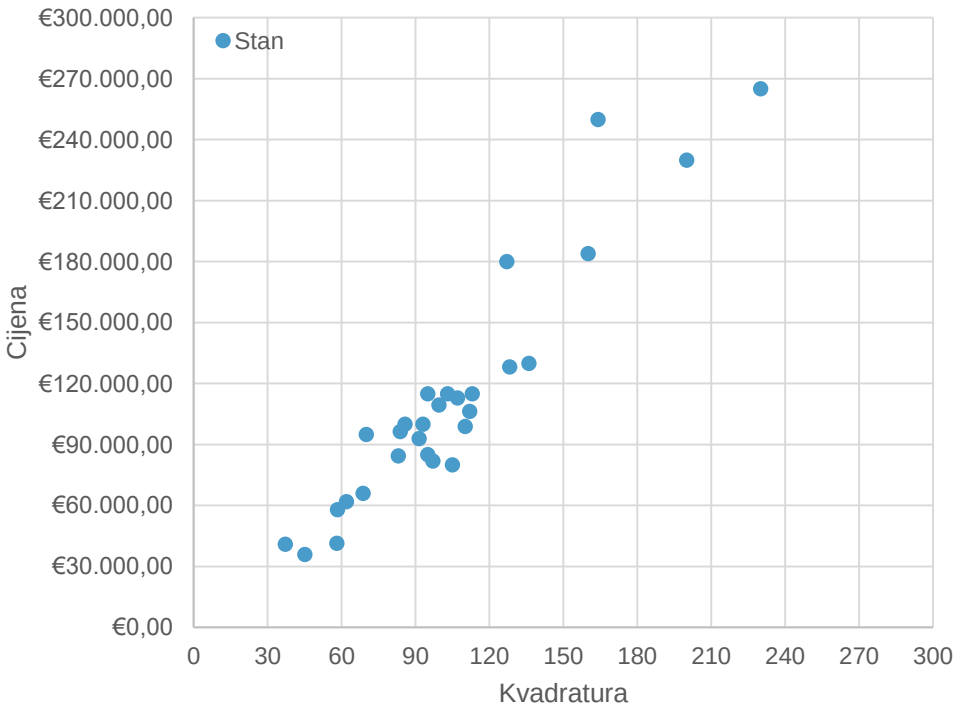
Učenje iz „podataka” koristi se u situacijama gdje nemamo analitičko rješenje, ili je ono prekomplicirano, ali imamo podatke koje možemo koristiti kako bi dobili empirijsko rješenje, tj. rješenje na temelju iskustva. [1]

Primjer 2.1. Procjena cijene stana u Osijeku na temelju kvadrature stana

S hrvatske web stranice za online trgovanje Njuškalo prikupljeni su podaci o cijeni stanova u Osijeku i njihovim kvadraturama. Na temelju tih podataka dobiven je graf 2.1. [2]

Kružići na grafu predstavljaju stanove. Njihova x vrijednost je veličina tog stana (kvadratura), a y vrijednost je cijena tog stana u eurima.

Graf 2.1. *Ovisnost cijene stana o kvadraturi stana*
Izvor: Autor

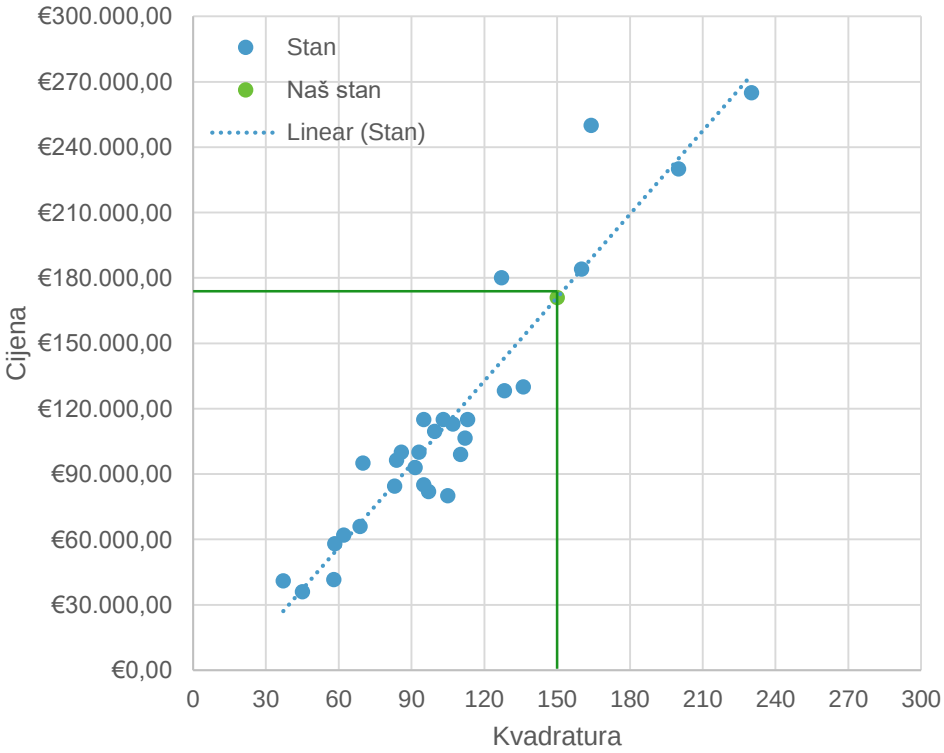


Tab. 2.1*. Cijene i kvadrature stanova u Osijeku*
Izvor: Autor

Kvadratura stana(m^2)	Cijena stana(€)	Kvadratura stana(m^2)	Cijena stana(€)
230	265 000	128,24	128 240
164	250 000	103	115 000
200	230 000	113	115 000
160	184 000	95	1149 90
127	180 000	106,98	113 000
58,37	58 000	99,54	109 494
37,1	41 000	112	106 400
62	62 000	85,7	100 000
68,76	66 000	93	100 000
83	84 500	110,15	99 000
45	36 000	83,79	96 350
58	41 500	70	95 000
136	130 000	91,39	93 000
97	82 000	95	85 000

Recimo da želimo prodati ili kupiti stan u Osijeku i želimo algoritam koji će nam reći koliko vrijedi taj stan. Budući da se iz grafa može vidjeti da postoji linearna ovisnost između kvadrature i cijene stana, može se povući pravac koji najbolje odgovara svim vrijednostima grafa (graf 2.2.). Ako stan ima 150m^2, bit će procijenjen na 171 000€.
Budući da su ovisnosti linearno opisane, ovakav model naziva se linearna regresija.

Graf 2.2. *Ovisnost cijene stana o kvadraturi stana*
Izvor: Autor



2.1. Komponente učenja

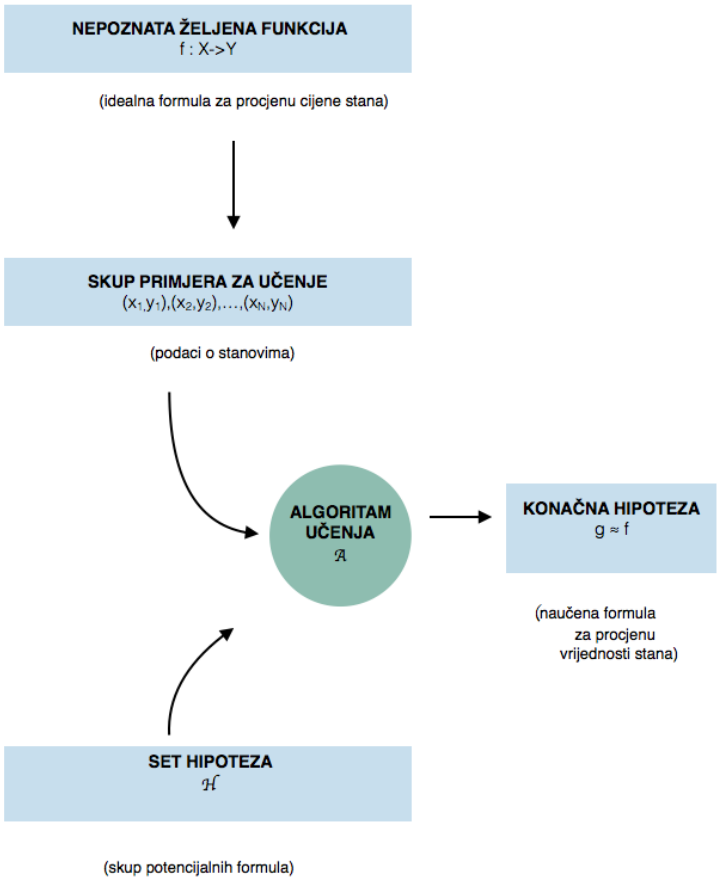
U primjeru procjene cijene stana u Osijeku na temelju kvadrature stana ulazna varijabla je kvadratura stana, a izlazna varijabla je cijena stana. Želimo poboljšati taj algoritam i ne želimo da cijena ovisi isključivo o kvadraturi, nego da se u obzir uzima lokacija, godina izgradnje, opremljenost stana i slično.

Pridružimo sada imena i simbole glavnim komponentama ovog problema učenja. Imamo ulaz x (informacije o stanu koje se koriste za procjenu vrijednosti stana), nepoznatu željenu funkciju

$f: X \rightarrow Y$ (idealna formula za procjenu vrijednosti stana), gdje X predstavlja skup svih mogućih ulaza x, a Y predstavlja skup svih mogućih izlaza, tj. cijene stana. D je skup ulazno-izlaznih primjera $(x_1,y_1),\ldots,(x_N,y_N)$ gdje je $y_n=f(x_n)$ za $n=1,\ldots,N$, tj. ulazi odgovaraju prethodnim stanovima i cijeni koja je, retrospektivno gledajući, točno određena. [1]

Na kraju ostaje algoritam učenja koji koristi skup D kako bi našao formulu $g: X \rightarrow Y$ koja aproksimira funkciju *f*. Algoritam bira *g* iz skupa potencijalnih formula, koji se naziva skup hipoteza H. U ovom slučaju H može biti skup svih linearnih formula iz kojih će algoritam izabrati onu koja najbolje odgovara podacima.

Recimo da ovaj algoritam koristi tvrtka koja se bavi prodajom nekretnina. Ona će svoju procjenu cijene nekretnine temeljiti na g (hipotezi koju je proizveo algoritam učenja), a ne na *f* (idealnoj funkciji koja ostaje nepoznata). Procjena će biti dobra samo ako *g* dosljedno aproksimira *f*.
Da bi to postigao, algoritam bira *g* koji najbolje interpolira *f* na temelju primjera prijašnjih stanova.



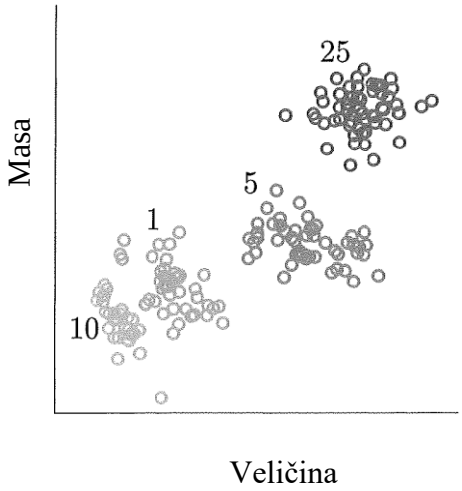
Sl. 2.1. *Osnovni primjer problema učenja*
Izvor: Autor prema [1]

2.2. Dizajn ili učenje?

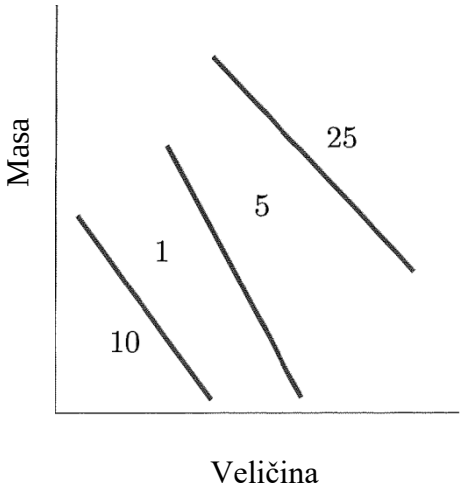
Dok se učenje temelji na podacima, drugi pristupi ih ne koriste. To je pristup „dizajniranja" koji se temelji na specifikacijama (tehničkim podacima).

Uzmimo za primjer problem prepoznavanja kovanica različitih vrijednosti koji je presudan, primjerice, u slučaju korištenja automata na kovanice. Budući da provođenje vlastitog eksperimenta nije bilo moguće zbog manjka informacija, primjer je preuzet (prema [1]). Cilj je da automat prepozna kovanice od 1, 5, 10 i 25 centa. Usporedimo pristup učenja iz podataka s pristupom dizajniranja iz specifikacija. Svaki novčić (svaka kovanica) bit će predstavljen veličinom i masom, odnosno uređenim parom ulaznih jedinica.

Graf 2.3.1. *Pristup učenjem klasifikaciji kovanica: Podaci o kovanicama*
Izvor:[1]



Graf 2.3.2. *Pristup učenjem klasifikaciji kovanica: Naučena klasifikacija*
Izvor:[1]



Graf 2.3.1. predstavlja podatke za učenje (1, 5, 10 i 25 centa su prikazani u prostoru veličina-masa u kojem se grupiraju). Graf 2.3.2. predstavlja klasifikacijsko pravilo je naučeno iz podataka grupiranjem u četiri grupe. Nova kovanica će biti klasificirana na temelju regije, u veličina-masa ravnini, kojoj pripadne.

U pristupu učenjem pred sobom imamo uzorak kovanica iz svake od navedene četiri vrijednosti i te kovanice se koriste kao set podataka. Veličina i masa smatraju se ulaznim jedinicama, a vrijednost je izlazna jedinica. Graf 2.3.1. prikazuje kako set podataka može izgledati u području ulazne vrijednosti. Postoje određene varijacije veličine i mase u svakoj klasi, ali ugrubo gledajući,

kovanice istih vrijednosti grupiraju se zajedno. Algoritam učenja traži hipotezu koja dobro klasificira set podataka. Ako želimo klasificirati novu kovanicu, stroj mjeri njezinu veličinu i masu, a onda ju klasificira prema naučenoj hipotezi iz grafa 2.3.2.

U pristupu dizajniranjem, kontaktirana je američka tvrtka koja se bavi izradom kovanica i prikupljene se specifikacije svake kovanice kao i informacija o broju kovanica svih vrijednosti kako bi se mogla procijeniti relativna učestalost svake kovanice. Na kraju je napravljen fizički model varijacija veličine i mase uzrokovanih izlaganjem određenim elementima i pogreškama napravljenim prilikom mjerenja. Sve informacije su spojene i izračunata je cjelokupna zajednička razdioba vjerojatnosti veličine, mase i vrijednosti kovanica (prema grafu 2.4.1.). Nakon što je dobivena razdioba vjerojatnosti, stvoreno je pravilo optimalne odluke kako bi bila moguća klasifikacija kovanice na temelju veličine i mase (prema graf 2.4.2.). Pravilo bira vrijednost kovanice koja je najvjerojatnija za određenu veličinu i masu kovanice i tako uspijeva ostvariti najmanju moguću vjerojatnost pogreške.

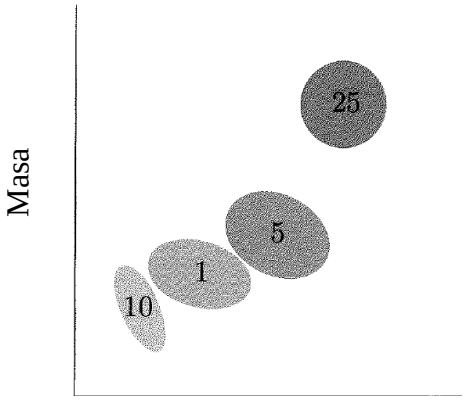
Glavna razlika između pristupa učenjem i dizajniranjem uloga je koju pritom igraju podaci. U pristupu dizajniranjem, problem je dobro određen i *f* se može analitički izvesti bez potrebe da se proučavaju podaci. U pristupu učenjem, problem je puno manje određen i potrebni su podaci kako bi se odredilo što je *f*.

Oba pristupa mogu se primjenjivati, ali samo je pristup učenjem moguć u

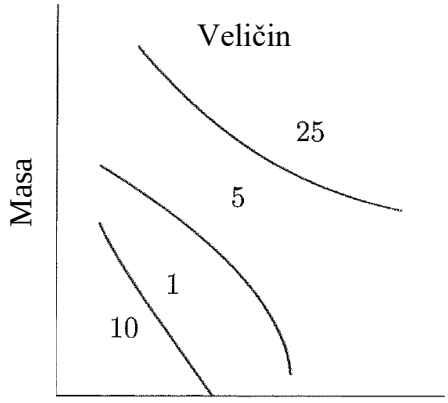
Zahtjevima gdje tražena funkcija nije poznata. Ne pokušavamo usporediti korisnost ili performanse ova dva pristupa, nego tek utvrditi kako je pristup dizajniranjem različit od učenja.

Vjerojatan model veličine, mase i vrijednosti kovanica izvodi se iz poznatih specifikacija. Graf 2.4.1. prikazuje veliko područje vjerojatnosti za svaku vrijednost (1, 5, 10 i 25 centa) prema modelu.

Graf 2.4.1. *Pristup klasifikaciji kovanica dizajniranjem: podaci o kovanicama*
Izvor:[1]



Graf 2.4.2. *Pristup klasifikaciji kovanica dizajniranjem: izvedeni klasifikator*
Izvor:[1]



Pravilo klasificiranja izvodi se analitički kako bi se smanjila vjerojatnost pogreške u klasificiranju kovanica na temelju mase i veličine. Na grafu 2.4.2. prikazana su dobivena područja za svaku vrijednost.

2.3. Tipovi učenja

Nadzirano učenje

Pomoću nadziranog učenja mogu se rješavati dvije vrste problema: klasifikacija i regresija.

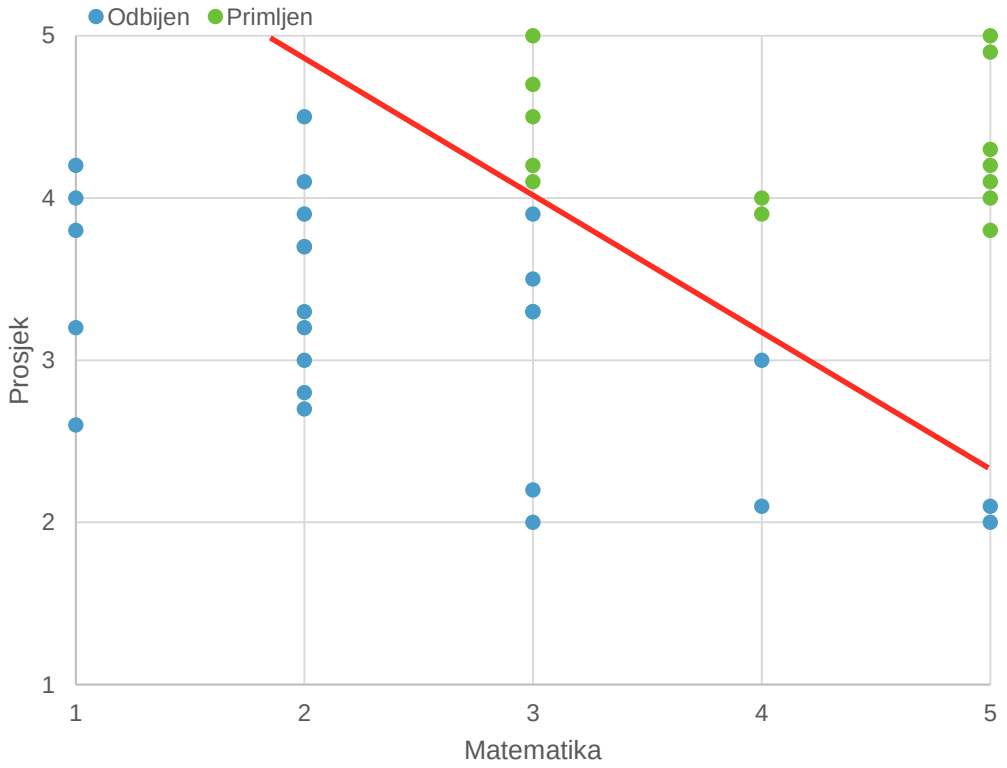
Kada pokusni podaci sadrže točne primjere željenih izlaza za dane ulaze, tada je riječ o nadziranom učenju. Dobar primjer rješavanja problema regresije pomoću nadziranog učenja je primjer 1. U tom primjeru algoritmu je dan skup podataka X koji predstavljaju kvadraturu stana i odgovarajući Y koji predstavlja cijenu stana. Ovakav pristup problemu naziva se nadziranim učenjem jer je algoritam nadziran, tj. algoritmu se daju točni odgovori odnosno točni izlazi za neki broj stanova te se traži od algoritma da nauči odnos između ulaza i izlaza kako bi mogao sam davati točne odgovore, tj. odgovarajuće izlaze za dane ulaze. Ovo je ujedno problem regresije jer su varijable koje se pokušavaju predvidjeti kontinuirane varijable (cijene stanova).

Drugi tip problema je klasifikacijski problem. U klasifikacijskom problemu varijabla koja se pokušava predvidjeti je diskretna. Recimo da fakultet odlučuje hoće li studenta primiti ili ne na temelju prosjeka ostvarenog tijekom srednje škole i ocjene iz matematike na državnoj maturi.

Graf 2.5. prikazuje dosadašnju selekciju studenata fakulteta gdje plavi kružići predstavljaju studente koji nisu primljeni, a zeleni kružići studente koji su primljeni. Na temelju tih podataka može se procijeniti hoće li idući student biti primljen ili ne.

Kod klasifikacije, primjeru pridružujemo klasu (razred) kojoj taj primjer pripada, dok kod regresije primjeru pridružujemo neku kontinuiranu vrijednost. Razlika je u tome je li ciljna varijabla diskretna ili je kontinuirana. [5]

Graf 2.5. *Povijest fakultetove slekecije studenata*
Izvor: Autor



Podržano učenje

Kada podaci za učenje ne sadrže eksplicitno točan izlaz za svaki ulaz, tada više riječ nije o nadziranom učenju. Primjer podržanog učenja je djetetovo shvaćanje da ne treba dirati šalicu vrele kave. Svaki put kada je dijete suočeno s vrelom šalicom kave može birati između diranja i ne diranja šalice. Svaki put kada bi dijete dodirnulo šalicu, iskusilo bi veliku razinu nelagode, a svaki put kada ne bi dodirnulo šalicu, iskusilo bi samo nelagodu neispunjene znatiželje. S vremenom dijete nauči kako je bolje ne dirati šalicu.

Skup primjera za učenje nije točno rekao što dijete treba činiti, nego je umjesto toga ocjenjivao različite postupke. Ovakav način učenja predstavlja podržano učenje gdje skup primjera za učenje ne sadrži ciljani izlaz, već sadrži moguće izlaze zajedno s mjerom koliko je taj izlaz dobar. U usporedbi s nadziranim učenjem gdje je forma bila (ulaz, točan izlaz), primjeri u podržanom učenju imaju formu (ulaz, neki izlaz, ocjena tog izlaza).

Podržano učenje može biti korisno pri učenju neke igre. Zamislimo situaciju u pokeru gdje moramo izabrati jedan od par mogućih izbora, najčešće ti izbori su odustati, pratiti ili dići. Naš je cilj identificirati najbolju moguću odluku. Određivanje najbolje odluke u određenom stadiju igre nije trivijalan zadatak, stoga se ne mogu tako lako stvoriti primjeri nadziranog. Ako se koristi podržano učenje, sve što moramo učiniti jeste poduzeti neku akciju i izvijestiti koliko je ona bila dobra. Algoritam podržanog učenja će zatim na temelju informacija iz raznih primjera donijeti najbolju odluku. [6]

Nenadzirano učenje

U nenadziranom učenju, podaci za učenje ne sadrže nikakve izlazne informacije. Recimo da u primjeru s kovanicama nisu poznate denominacije kovanica. I dalje ćemo dobiti slično grupiranje, ali će sada te grupe biti neoznačene.

Za nenadzirano učenje može se reći da je to spontano pronalaženje uzoraka i struktura u ulaznim podacima. U primjeru kategorizacije skupine knjiga po temama za koje se koriste opća svojstva različitih grupa, mogu se identificirati knjige koje imaju slična svojstva i grupirati pod zajedničku kategoriju bez imenovanja kategorije.

Na nenadzirano učenje se može gledati kao na način da se stvori bolja razina prikaza podataka. Recimo da ne znate ni riječ njemačkog, ali sljedeći mjesec se selite u Njemačku zbog posla. Vaša

tvrtka je dogovorila tečaj njemačkog, ali vi se želite bolje pripremiti prije nego što odete. Jedino čemu imate pristup su njemački TV programi. Punih mjesec dana ste u stalnom doticaju s njemačkim jezikom; ovo je iskustvo nenadziranog učenja budući da ne znate značenje riječi. Međutim, postepeno razvijate bolji prikaz jezika u mozgu tako što postajete naviknuti na jednostavne zvukove i jezične konstrukcije. Kada dođete u Njemačku, bit ćete u boljoj početnoj poziciji s lekcijama njemačkog. Dakle, nenadzirano učenje može biti prethodnik nadziranom učenju. [1]

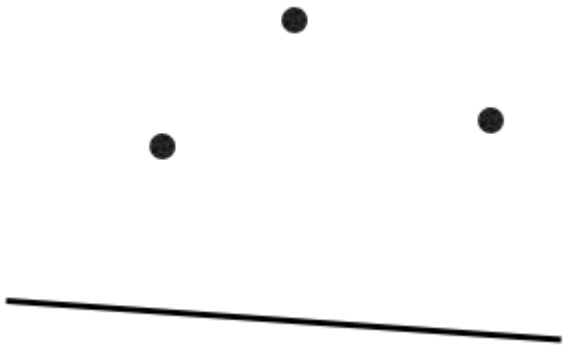
2.4. Algoritmi strojnog učenja

Linearni model

Linearni model se koristio u prvom primjeru gdje se predviđala cijena stana na temelju njegove kvadrature, gdje se na temelju podataka prikupljenih s Njuškala dobio graf s brojnim točkama koji predstavljaju te stanove. Na kraju je povučen pravac na temelju tih točaka koji nam je pomogao da procijenimo cijenu željenog stana. Metoda kojom se taj pravac dobio naziva se linearna regresija.

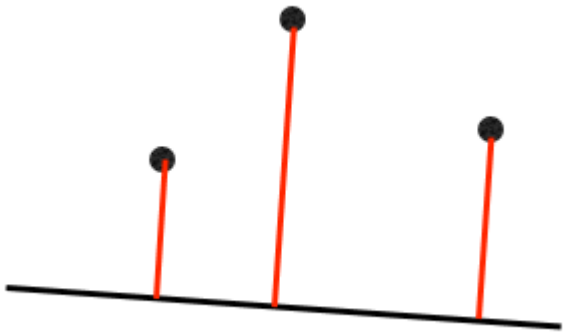
Sl. 2.2. *Tri nasumične točke*

Ako želimo povući pravac koji najbolje odgovara ovim točkama prvi korak je da povuče se pravac bilo gdje.



Sl. 2.3*. Tri nasumične točke i nasumični pravac*

Nakon povučenog nasumičnog pravca potrebno je provjeriti koliko je on dobar, ili loš, stoga se računa pogreška. Kako bi se izračunala pogreška, gledaju se udaljenosti točaka od povučenog pravca.



Sl. 2.4. Udaljenosti između točaka i nasumičnog pravca

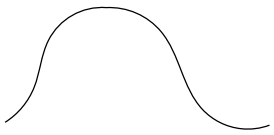
Pogrešku predstavlja zbroj tri crvene linije koje spajaju točke s nasumičnim pravcem.

Sl. 2.5. *Računanje ukupne pogreške*

Nakon računanja pogreške, pomakne se pravac, a zatim se ponovno računa pogreška. Ako se pogreška povećala nakon pomicanja pravca, zaključak je kako će se daljnjim pomicanjem u istom smjeru pogreška jedino povećavati, stoga se pravac pomiče u suprotnom smjeru. Ovaj postupak se ponavlja sve dok se ne nađe najbolje odgovarajući pravac. Ovaj postupak se naziva gradijentni spust. Budući da u stvarnom životu ne postoje negativne udaljenosti koriste se kvadrati udaljenosti kako bi se riješio problem negativnih udaljenosti.

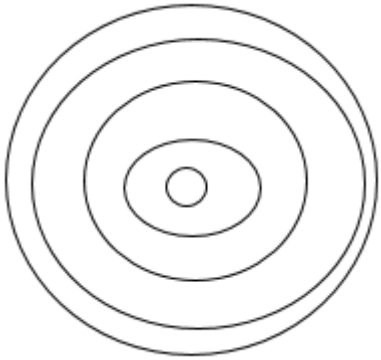
Gradijentni spust

Recimo da imamo povez na očima i želimo se popeti na vrh brda, u što manje koraka, koje izgleda prema slici 1.6.



Sl. 2.6. *Skica brda*

Budući da je brdo strmije, a vrh je daleko, prvi koraci bit će nam veći. Zbog toga možemo si priuštiti velike korake. Kako se približavamo vrhu brda, ono je sve ravnije, stoga želimo raditi manje korake kako ne bi premašili vrh brda.



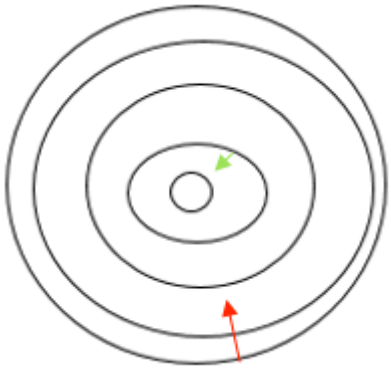
Sl. 2.7. *Pogled na brdo odozdo*

Ova skica predstavlja brdo ako ga pogledamo odozdo, gdje svaka „kružnica" predstavlja razine brda iste visine. Budući da je brdo ravnije na vrhu, razlike u visini su manje.

Gradijent skalarnog polja je vektorsko polje koje pokazuje u pravcu najvećeg porasta skalarnog polja te čiji je intenzitet najveća promjena u polju. [8]

U svakoj točki brda gradijent će pokazati smjer u kojem nagib opada najviše. Intenzitet gradijenta će odrediti koliko se brzo nagib smanjuje u tom pravcu.

Ako se uzme gradijent (crvena strelica) na samom početku brda vektor će biti duži budući da je, u tom dijelu, brdo najstrmije (njegova točna dužina ovisi o strmosti brda). Vektor pokazuje direktno na vrh brda. Ako se gradijent uzme na kraju brda (zelena strelica), također će pokazivati na vrh brda, ali će biti znatno manji budući da je nagib znatno manji pri vrhu. Sa smanjenim nagibom dolazi smanjeni gradijent i smanjeni koraci koje ćemo napraviti. [7]



Sl. 2.8. *Pogled na brdo odozdo s gradijentima*

Naive-Bayes algoritam

Recimo da imamo 100 e-mailova (elektronička pošta) od kojih je 25 spam pošta. Na temelju ovih podataka, želimo algoritam koji će moći prepoznati je li mail spam ili nije. Za početak uzmemo neku riječ ili svojstvo koje bi spam mail mogao sadržavati, kao naprimjer „jeftin" ili „*cheap*". Nakon analize podataka otkrili smo da od 25 spam mailova 20 sadrži riječ „jeftin" ili „*cheap*" a da od 75 ostalih mailova samo 5 sadrži riječ „jeftin" ili „*cheap"*. Od 25 mailova koji sadrže riječ „jeftin" ili „*cheap*", 20 su spam mailovi, tj. 80% . Zaključujemo, ako mail sadrži riječ „jeftin" ili „*cheap*" sa sigurnošću od 80% možemo reći da je spam. Na ovo svojstvo možemo dodati još neka, kao npr. nedostajanje naslova, gramatička greška i slično. [9]

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{2-1}$$

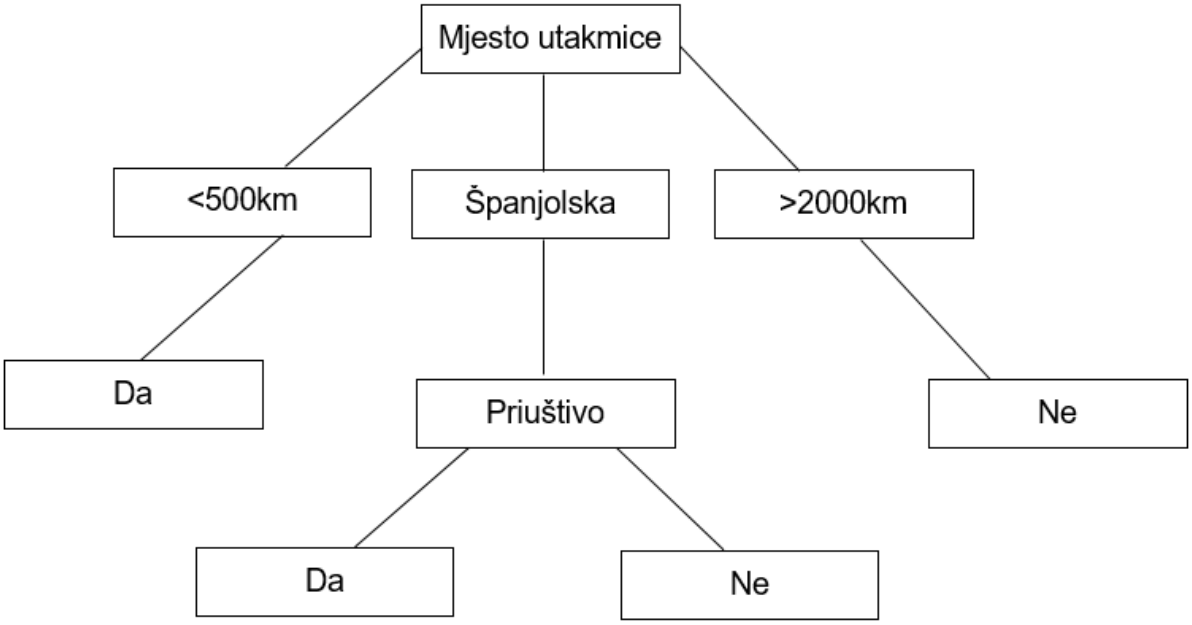
$P(h)$ - vjerojatnost hipoteze h
$P(D)$ - vjerojatnost podataka za učenje D
$P(D|h)$ - vjerojatnost D za dani h
$P(h|D)$ - vjerojatnost h za dani D

Naive – Bayes algoritam može se koristiti u rješavanju problema klasifikacije. Ako ga primijenimo na primjeru upisa na fakultet, on će na temelju skupa podataka i izračunatih vjerojatnosti procijeniti u koje područje student spada, tj. je li primljen ili nije primljen. Algoritam koristi jednostavnije matematičke operacije između „manjeg" skupa podataka i iz tog razloga je brz i jednostavan, ali u usporedbi s ostalim algoritmima je manje efikasan.

Stabla odluke

Stabla odluke klasificiraju primjere tako što ih sortiraju pomoću stabla od korijena do čvorova (listova). Svaki čvor stabla predstavlja test nekog svojstva, a svaka grana koja proizlazi iz čvora predstavlja neku od mogućih vrijednosti koje svojstvo može poprimiti.

Mali Kruno cijeloga života silno želi otići na utakmicu Real Madrida, ali nikako nije bio u prilici. Stablo odluke za odlazak na utakmicu Real Madrida malog Krune je dano na slici 2.9.

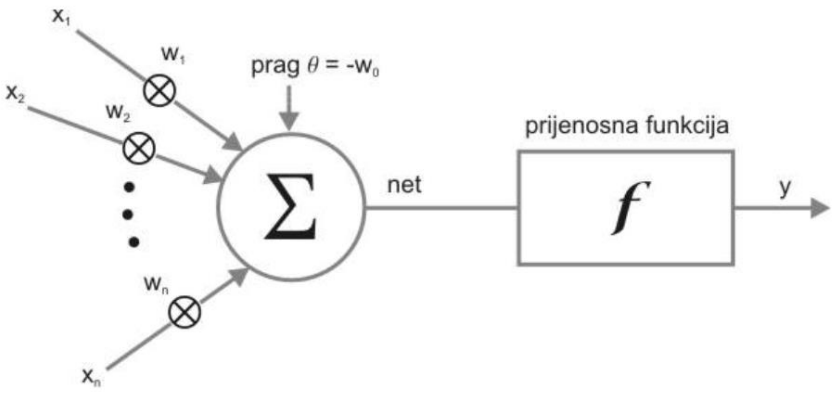


Sl. 2.9. *Stablo odluke za odlazak na utakmicu Real Madrida*

Slika 2.9. Ilustrira tipično stablo odluke. Svaki puta kada Real Madrid igra utakmicu stablo klasificira hoće li Kruno na nju otići ili neće. Recimo da igraju Barcelona i Real Madrid u Španjolskoj utakmicu koja će presuditi domaće prvenstvo, a Kruno silno želi ići i svojim očima vidjeti najbolje nogometaše svijeta i Danila, kako igraju protiv Barcelone. Međutim, Kruno je student i ne može si priuštiti takav trošak. Ovakav primjer (Mjesto utakmice=Španjolska, priuštivo=ne) bio bi sortiran u najniži desni čvor i stoga bio bi klasificiran kao ponovno Krunino razočaranje i neodlazak na utakmicu.

Neuronske mreže

Neuronska mreža, u širem smislu, umjetna je replika ljudskog mozga kojom se nastoji simulirati postupak učenja. Ona je skup međusobno povezanih jednostavnih procesnih elemenata, jedinica ili čvorova, čija se funkcionalnost temelji na biološkom neuronu. Pri tome je obrambena moć mreže pohranjena u snazi veza između pojedinih neurona tj. težinama do kojih se dolazi postupkom prilagodbe odnosno učenjem iz skupa podataka za učenje. Neuronske mreže odlično rješavaju probleme klasifikacije i predviđanja, odnosno općenito sve probleme kod kojih postoji odnos između ulaznih i izlaznih varijabli, bez obzira na visoku složenost te veze (nelinearnost). [11]



Sl. 2.10. *Umjetni neuron*
Izvor: [11]

Ulazne signale, njih ukupno n, označavamo s x_1, x_2, …x_n. Težine označavamo s ω_1, ω_2, … ω_n. Ulazni signali općenito su realni brojevi u intervalu [-1, 1], [0, 1] ili samo elementi iz {0 ,1}, kada govorimo o Booleovom ulazu. Težinska suma *net* dana je s (prema [11])

$$net = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n - \theta \tag{2-2}$$

ali se zbog kompaktnosti često dogovorno uzima da je vrijednost praga $\theta = -\omega_0$ te se dodaje ulazni signal x_0 s fiksiranom vrijednošću 1, pa pišemo jednostavnije prema

$$net = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n = \sum_{i=0}^{n} \omega_i x_i \tag{2-3}$$

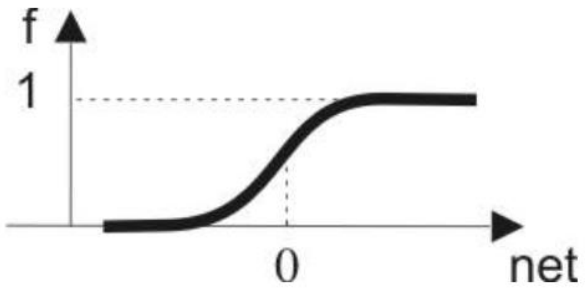
dok je izlaz y rezultat prijenosne funkcije [11]

$$y = f(\sum_{i=i}^{n} \omega_i x_i) = f(net) \tag{2-4}$$

Najčešći oblik prijenosne funkcije jest sigmoidalna funkcija. Ova je funkcija derivabilna što je bitna prednost pri postupku učenja umjetne neuronske mreže. Sigmoidalna funkcija definirana je kao:

$$f(net) = \frac{1}{1 + e^{-a*net}} \tag{2-5}$$

uz parametar *a* koji određuje nagib funkcije. Sigmoidalna funkcija ponekad se naziva i logističkom funkcijom.



Sl. 2.11. *Sigmoidalna funkcija*
Izvor: [11]

U slučaju korištenja sigmoidalne funkcije tipično se gubi nadzor nad načinom kako mreža obrađuje podatke. U tom slučaju uobičajeno je da se definira arhitektura mreže, i prije postupka obrade podataka obavi postupak učenja ili treniranja. Za razliku od konvencionalnih tehnika obrade podataka, gdje je postupak obrade potrebno analitički razložiti na određeni broj algoritamskih koraka, kod ovog tipa neuronskih mreža takav algoritam nije potreban. Znanje o obradi podataka, tj. znanje o izlazu kao funkciji ulaza, pohranjeno je implicitno u težinama veza između neurona. Te se težine postupno prilagođavaju kroz postupak učenja neuronske mreže sve

do trenutka kada je izlaz iz mreže, provjeren na skupu podataka za testiranje, zadovoljavajući. Pod postupkom učenja kod neuronskih mreža podrazumijevamo iterativan postupak predočavanja ulaznih primjera (uzoraka, iskustva) i eventualno očekivana izlaza.

Recimo da želimo procijeniti koliki postotak će studenti ostvariti na kolokviju na temelju vremena provedenog učeći i vremena provedenog spavajući.

Tab. 2.2. *Rezultati kolokvija na temelju sati spavanja i učenja*

X (broj sati spavanja, broj sati učenja)	y (rezultat na kolokviju)
(0,7)	0.65
(6,3)	0.70
(10,2)	0.80
(8,3)	?

Ovakav problem je regresijski nadzirani problem. Spada u kategoriju nadziranog jer naši primjeri imaju ulaze i izlaze, te u kategoriju regresijskog problema jer predviđamo rezultat kolokvija, koji je kontinuirana varijabla. Na temelju tablice 2.2. možemo napraviti neuronsku m

Težinska suma svakog čvora skrivenog sloja dobije se prema formuli (2-3) na čiji se rezultat primijeni prijenosna funkcija, u ovom slučaju sigmoidalna funkcija (prema formuli (2-5)). Težine veza između čvorova nisu nam poznate stoga je mrežu potrebno istrenirati. Treniranje se može izvesti pomoću nekih od programskih jezika kao što je Python. Uzmimo za početak nasumične vrijednosti težina veza. Dobiveni rezultati će vjerojatno poprilično odstupati od stvarnih. Za poboljšanje modela koristi se funkcija troška koja govori koliko je model pogrešan:

$$J = \frac{1}{2N}\sum_{i=1}^{N}(y - y_i)^2 \tag{2-6}$$

gdje je y stvarni rezultat, a y_i i-ti provedeni test. Funkcija računa zbroj kvadriranih pogrešaka umanjen za 2N. Treniranje mreže zapravo predstavlja minimiziranje funkcije troška. Ako X uzmemo kao matricu ulaza i W kao matricu težina, veza između ulaznog sloja i skrivenog sloja slijedi da je Z matrica težinskih suma skrivenog sloja:

$$X * W_1 = Z_1 \tag{2-7}$$

tj.: $\begin{bmatrix} 0 & 7 \\ 6 & 3 \\ 10 & 2 \end{bmatrix}\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} = \begin{bmatrix} 0w_{11} + 7w_{21} & 0w_{12} + 7w_{22} & 0w_{13} + 7w_{23} \\ 6w_{11} + 3w_{21} & 6w_{12} + 3w_{22} & 6w_{13} + 3w_{23} \\ 10w_{11} + 2w_{21} & 10w_{12} + 2w_{22} & 10w_{13} + 2w_{23} \end{bmatrix}$

Primjenom sigmoidalne funkcije na matricu Z_1 dobije se matrica A. Nadalje, iz matrice A i matrice W_2, koja predstavlja težinske veze između skrivenog sloja i izlaznog sloja, dobije se matrica Z_2.

$$A * W_2 = Z_2 \tag{2-8}$$

Konačno, primjenom sigmoidalne funkcije na Z_2 dobije se procjena kolokvija y_i

Jednadžbe 2-4, 2-6, 2-7 i 2-8 možemo spojiti u jednu jednadžbu:

$$J = \frac{1}{2N}\sum_{i=1}^{N}(y - f(f(XW_1)W_2))^2 \tag{2-9}$$

Zanima nas stopa promjene J u odnosu na W, tj. parcijalna derivacija J. Ako je derivacija pozitivna, funkcija troška raste, ako je derivacija negativna, funkcija troška se smanjuje. Budući da znamo u kojem smjeru se funkcija smanjuje, vrijeme trajanje treniranja mreže je znatno smanjeno

(gradijentni spust). Ako bismo mrežu trenirali tako da testiramo sve moguće kombinacije težina veza, trebalo bi nam 1027 testova kako bismo došli do željenih rezultata, dok nam uz pomoć gradijentnog spusta treba manje od 100 testova.

III. MODELIRANJE TEXAS HOLD'EM POKER IGRAČA

Dok je u igrama potpunih informacija, kao što je šah, računalo uspjelo nadmašiti i najbolje svjetske šahiste, igre nepotpunih informacija kao što su poker, slabije su proučavane i botovi umjetne inteligencije su i dalje ispod nivoa profesionalnih igrača pokera. Za razliku od šaha, gdje je cijelo stanje igre vidljivo svima, poker je komplicirana igra, s aspektom nasumičnosti, koja sadrži brojne atribute kao što su nesavršeno znanje (skrivene ruke), brojnost sudionika (dva ili više igrača), upravljanje rizikom (ulaganje) i obmana (blefiranje). Ovo sve znatno komplicira razvijanje poker agenta umjetne inteligencije. Za uspješno modeliranje poker agenta, potrebno je razumjeti osnovna pravila i koncepte same igre.

3.1. Pravila

Svaka „ruka" ili partija se sastoji od četiri runde: *pre-flop*, *flop*, *turn* i *river*. Ruka *Texas Hold'em*

-a započinje s *pre-flop*-om, gdje su svakom igraču dodijeljene dvije karte licem dolje (*hole*-poznate samo njima), nakon čega slijedi prva runda ulaganja (*bet, betting*). Zatim, tri zajedničke karte podijeljene su licem gore na stol (*flop*), nakon čega slijedi druga runda ulaganja. Nakon *flop*-a, okreće se još jedna karta licem gore (*turn*) i slijedi treći krug ulaganja. Na *river*-u se dijeli peta i posljednja karta, također licem gore, nakon koje slijedi posljednja runda ulaganja. Svi igrači, koji su i dalje u igri, okreću svoje dvije (do tada samo njima poznate) karte licem gore i dolazi do *showdown*-a. Najbolja kombinacija (tablica 3.1.) pet karata formirana od dvije vlastite karte i pet zajedničkih karata, pobjeđuje i odnosi *pot* (sav uloženi novac u toj ruci). Ako dva ili više igrača imaju jednako jaku ruku, *pot* se dijeli između njih.

Tab. 3.1. Kombinacije u pokeru od najslabije prema najjačoj

Ime kombinacije	Značenje
High card	Nije kombinacija. U slučaju jednako jake kombinacije igrač s najjačom high card pobjeđuje.
Jedan par	Dvije karte iste vrijednosti. U slučaju da dva igrača imaju jedan par, igrač s parom jačih karata pobjeđuje. (npr. AA)
Dva para	Dva para karata istih vrijednosti. Primjer: AA i KK
Tris	Tri karte istih vrijednosti. Primjer: AAA
Skala	Pet karata u nizu. Primjer: 23456
Boja	Pet karata iste boje. Primjer: 5◆A◆K◆2◆10◆
Full house	Tris + jedan par. Primjer: 22555
Poker	Četiri karte istih vrijednosti. Primjer: AAAA
Skala u boji	Pet karata iste boje u nizu. Primjer :2◆3◆4◆5◆6◆

Karte dijeli *dealer* (djelitelj) kojeg predstavlja jedan od igrača za stolom (igrač koji je dealer ne mora nužno biti i stvarni fizički *dealer*, tko stvarno dijeli karte je nebitno, bitna je pozicija *dealear*-a prema kojoj se određuju pozicije ostalih igrača). Karte se dijele u lijevo, gdje prvi igrač s lijeva predstavlja poziciju *small blind*, a drugi *big blind*. *Blind* je obavezni ulog koji svaki igrač mora platiti u prvoj rundi ako želi biti u ruci. Ovisno o dogovorenom velikom *blind*-u, mali *blind* automatski ulaže pola dogovorenog velikog *blind*-a, a veliki ulaže cijeli *blind*. Nakon što su karte podijeljene, prvi na potezu je igrač koji je prvi s lijeva do velikog *blind*-a. U svakoj rundi ulaganja igrač ima sljedeće opcije: *bet*, *check*, *raise*, *fold* ili *call*. Opcija *check* je moguća ako nije potreban nikakav iznos kako biste i dalje ostali u igri. *Check* je zapravo ekvivalentan ulaganju iznosa 0. Ako nitko nije ulagao (eng. *bet*), osim opcije check, postoje opcije *bet* i *fold*. *Bet* predstavlja vaše ulaganje u *pot*, koje od drugih igrača zahtijeva plaćanje minimalno tog iznosa ako žele ostati u igri. Igrač može *fold*-ati u bilo kojem trenutku (ako je na redu), što znači da odustaje od igre i svih

uloga koji su trenutno u igri. Ako je *bet*-ao, osim opcije *fold*, postoje opcije *call* i *raise*. *Call* znači da plaćate uloženi iznos, a *raise* da na uloženi iznos ulažete još više i zahtijevate od drugih igrača da plate iznos za koji ste išli *raise* ako žele ostati u ruci. Valja napomenuti da se redoslijed igre mijenja nakon *pre-flop*-a: na *flop*-u, *river*-u i *turn*-u prvi na igri je igrač na poziciji *small blind*.

3.2. Procjena ruke

Za modeliranje dobrog igrača pokera potreban je niz ključnih komponenti. Svaka od tih komponenti, direktno ili indirektno, utjecana je modeliranjem protivnika.

Pozicije

Jedna od bitnih stavki koju profesionalni igrači uzimaju u obzir je pozicija za stolom u odnosu na djelitelja (koliko je igrača igralo prije nas i koliko igra poslije nas). Kasnija pozicija se smatra boljom pozicijom jer je dostupno više informacija i stoga se može donijeti bolja odluka.

Slika 3.1. *Pozicija u Texas Hold'em Pokeru*

Pre-flop procjena

Procjena jačine ruke prije *flop*-a je detaljno izučavano područje u literaturi pokera. Prema [17] klasificiraju se sve moguće kombinacije prve dvije karte (prije *flop*-a) u 9 različitih kategorija ulaganja. Za svaku kategoriju, na temelju jačine ruke, broja protivnika, pozicije za stolom (s obzirom na djelitelja) i tipa protivnika, predložena je strategija ulaganja.

Iako postoji 1326 mogućih kombinacija (prema formuli 3-2) dvije karte u deku, postoji samo 169 različito jakih početnih ruku (A♥A♣ i A♦A♠ su dvije različite, ali jednako jake ruke). Za svaku od ovih 169 ruku provedena je simulacija od 1 000 000 partija protiv jednog, tri i šest nasumičnih protivnika (kako bi se pokrili scenariji s dva, tri-četiri i pet ili više protivnika). Simulacije su izvedene na način da se svaka partija igrala do *showdown*-a bez odustajanja. Za svaku ruku i broj protivnika dobivena je približna stopa prihoda IR (engl. *income rate*), mjera povrata investicije. [15]

$$IR = \frac{ukupan\ prihod}{broj\ odigrnaih\ ruku} \tag{3-1}$$

gdje je:

- IR - stopa prihoda
- ukupan prihod - ukupan prihod tijekom svih odigranih partija
- broj odigranih ruku - broj odigranih partija

***Tab. 3.2**. Stopa prihoda za jednog protivnika [14]*

	2	3	4	5	6	7	8	9	T	J	Q	K	A
2	7	-351	-334	-314	-318	-308	-264	-217	-166	-113	-53	10	98
3	-279	74	-296	-274	-277	-267	-251	-201	-148	-93	-35	27	116
4	-263	-225	142	-236	-240	-231	-209	-185	-130	-75	-17	46	134
5	-244	-206	-169	207	-201	-189	-169	-148	-114	-55	2	68	153
6	-247	-208	-171	-138	264	-153	-134	-108	-78	-43	19	85	154
7	-236	-200	-162	-125	-91	324	-99	-72	-43	-6	37	104	176
8	-192	-182	-143	-108	-75	-43	384	-39	-4	29	72	120	197
9	-152	-134	-122	-84	-50	-17	16	440	28	65	106	155	215
T	-104	-86	-69	-56	-19	12	47	81	499	102	146	195	254
J	-52	-35	-19	0	11	46	79	113	149	549	161	212	271
Q	2	21	34	55	72	86	121	153	188	204	598	228	289
K	63	79	98	116	132	151	168	200	235	249	268	647	305
A	146	164	180	198	198	220	240	257	291	305	323	339	704

***Tab. 3.3.** Stopa prihoda za 3 protivnika [14]*

	2	3	4	5	6	7	8	9	T	J	Q	K	A
2	-121	-440	-409	-382	-411	-432	-394	-357	-301	-259	-194	-116	16
3	-271	-42	-345	-312	-340	-358	-371	-328	-277	-231	-165	-87	54
4	-245	-183	52	-246	-269	-287	-300	-308	-252	-204	-135	-55	84
5	-219	-151	-91	152	-200	-211	-227	-236	-227	-169	-104	-24	118
6	-247	-177	-113	-52	256	-145	-152	-158	-152	-145	-74	9	99
7	-261	-201	-129	-65	3	376	-76	-79	-68	-66	-44	48	148
8	-226	-204	-140	-73	-2	66	503	0	15	24	45	84	194
9	-191	-166	-147	-79	-5	68	138	647	104	113	136	177	241
T	-141	-116	-91	-69	-4	75	150	235	806	226	255	295	354
J	-89	-67	-41	-12	7	82	163	248	349	965	301	348	410
Q	-29	-3	22	51	80	108	185	274	379	423	1141	403	473
K	47	76	101	128	161	199	230	318	425	473	529	1325	541
A	175	211	237	266	249	295	338	381	491	539	594	655	1554

***Tab. 3.4**. Stopa prihoda za 5 protivnika*

	2	3	4	5	6	7	8	9	T	J	Q	K	A
2	-6	-462	-422	-397	-459	-495	-469	-433	-383	-336	-274	-188	-39
3	-180	21	-347	-304	-365	-418	-447	-414	-356	-308	-248	-163	-1
4	-148	-69	67	-227	-273	-323	-362	-391	-334	-287	-223	-133	32
5	-121	-38	31	122	-198	-230	-270	-303	-309	-259	-200	-103	64
6	-174	-95	-10	64	206	-151	-175	-204	-217	-235	-164	-72	23
7	-208	-135	-47	35	108	298	-87	-106	-112	-128	-124	-26	72
8	-184	-164	-83	2	93	168	420	-5	6	-10	-10	22	126
9	-146	-128	-111	-26	64	153	245	565	134	118	118	151	189
T	-88	-68	-46	-29	59	155	268	383	765	299	305	336	373
J	-38	-15	1	30	51	147	256	377	536	996	380	420	462
Q	35	49	72	99	127	162	268	384	553	628	1279	529	574
K	117	141	167	190	223	261	304	423	591	669	764	1621	712
A	269	304	333	363	313	365	416	475	644	720	815	934	2043

Ove tri tablice predstavljaju rezultate simulacija za svih 169 mogućih kombinacija ruku. Svaki ulaz je IR_x[red][stupac] gdje x predstavlja broj igrača (x-1 protivnika). Karte su u istoj boji ako je red > stupac. Iz tablice 3.3. vidimo da je IR_7[A][K]=934 dok je IR_7[K][A]=712.

Ruka 72o (o-*offsuit* (različita boja), s-*suit*(ista boja)) u tablici 3.3. ima stopu prihoda manju od 53o (IR_7[2][7]=-495, IR_7[3][5]=-397), ali direktnom sučeljavanju 72o ima veće izglede za pobjedu zbog jače high karte 7 (IR_2[2][7]=-308, IR_2[3][5]=-314). Međutim, ako igraju protiv više (boljih) ruku za punim stolom, 53o ima veće izglede za pobjedu zbog potencijalne skale.

No-fold'em tip simulacije može na neki način iskriviti rezultate, npr. u stvarnoj situaciji KK će pobijediti u više od 26% slučajeva budući da većina *fold*-a prije *river*-a, pa je tako i šansa da će neka slabija ruka pobijediti KK manja.

Tablica također ne uzima u obzir poziciju. Ruka, kao npr. 10♥J♣, neće dobro proći u ranoj poziciji, ali se ponekad može igrati na *button*-u. Budući da se vrijednost ruke mijenja s obzirom na poziciju

i broj igrača, ovakva tablica nije ni blizu stvarne priče, ali tablica je korisna za uvid u relativnu vrijednost ruke.

Jačina ruke

Jačina ruke govori koliko je ruka jaka u odnosu na ostale moguće ruke. Osnovna jačina ruke je funkcija vaših karata i zajedničkih karata. Ako procjenu jačine ruke želimo učiniti boljom i preciznijom, u obzir se uzima broj igrača koji su trenutno u igri, pozicija za stolom i povijest ulaganja za određenu ruku. Još preciznija procjena u obzir uzima vjerojatnosti za svaku moguću ruku protivnika.

Recimo da u ruci imamo A♥A♣ i na stolu je 4♦2♥10♥. Prije dijeljenja špil karata sadrži 52 karte. Ako od 52 oduzmemo dvije karte, koje su poznate samo nama i tri koje su poznate svima, ostaje 47 nama nepoznatih karata. Prema formuli (3-2) (kombinacije bez ponavljanja) znamo da postoji 1081 moguća ruku koju protivnik može imati. [5]

$$C_n^k = \frac{n!}{(n-k)!*k!} = \binom{n}{k} = \frac{47!}{45!*2!} = 1081,\ 0 \leq k \leq n \qquad (3\text{-}2)$$

gdje je:

- C_n^k - broj mogućih kombinacija
- n-broj preostalih karata u špilu
- k – broj karata koje su promatrane

Kako bismo saznali jačinu naše ruke, možemo se poslužiti algoritmom JačinaRuke (prema [14]) koji funkcionira tako da sve moguće protivnikove ruke uspoređuje s našom i broji je li ona bolja, lošija ili jača od naše. Nakon konačnog broja svake kategorije, dobiveni broj se dijeli s brojem mogućih protivnikovih ruku i dobije se postotak koji kaže koliko je naša ruka jaka. Za navedeni slučaj protivnik će biti jači samo u slučaju da ima triling ili dva para (devet kombinacija za triling i 36 za dva para, sve zajedno 45 slučaja) i samo jedan slučaj za neriješeno (preostala dva asa). To znači da je 1035 ruku slabijih od naše, tj. naša ruka je jača u 95.74% slučaja. Kada računamo jačinu ruke, broju slučajeva u kojem je naša ruka jača, dodamo polovicu neriješenih slučajeva (0.5), tj.

(1035+0.5)/1081=95.79%. U slučaju da igramo protiv više od jednog protivnika dobiveni postotak podijelimo s brojem protivnika.

```
JačinaRuke (našekarte, zajedničkekarte)
        {
        bolje=jednake=lošije=0
        ranka_naše_ruke= rank(naše_karte, zajedničke_karte)

/* ranking_moje_ruke predstavlja koliko je naša ruka jaka naspram
svih mogućih ruku u pokeru, tj. njenu poziciju s obzirom na ostale
u danoj situaciji, npr. ruka AA u gore navedenom primjeru bi bila
46. najbolja moguća ruka, budući da postoje 45 bolje moguće ruke*/
/* funkcija rank na temelju naših i zajedničkih karata pozicionira
našu ruku na ljestvici jačine*/
        for each case(protivnikove_karte)
        {
          ranking_protivnikove_ruke=     rank(protivnikove_karte,
zajedničke_karte)
          if(ranking_naše_ruke>ranking_protivnikove_ruke)
        bolje+=1
          else    if(ranking_naše_ruke=ranking_protivnikove_ruke)
        neriješeno+=1
          else    if(ranking_naše_ruke<ranking_protivnikove_ruke)
        lošije+=1
        }
        jačina_ruke=
(bolje+neriješeno/2)/(bolje+neriješeno+lošije)
        return (jačina_ruke)
}
```

Jačina ruke A♥Q♣ / 4♣3♥J♥ iznosi 58.5%, no, ako se u *poker odds* kalkulator (prema [16]) unese ista ova ruku, izbacit će postotak od 50.8%. Razlog tome je što jačina ruke sama po sebi nije dobar pokazatelj koliko je naša ruka zapravo dobra i zanemaruje potencijal ruke.

Potencijal ruke

U stvarnosti, jačina ruke nije sama dovoljna da procjeni koliko je ruka dobra. Potencijal ruke procjenjuje vjerojatnost da se ruka poboljša uzimajući u obzir potencijalne zajedničke karte koje još trebaju izaći. Recimo da je stanje sljedeće: u ruci imamo 2♥3♥ i na *flop*-u je izašlo 4♦4♥10♥. Ovakva ruka ima malu jačinu ruke, ali ima dobar potencijal da postigne boju na *turn*-u ili *river*-u. Na temelju stanja igre (naše ruke i stola) može se izračunati pozitivni potencijal (Ppt - vjerojatnost da naša ruka postane bolja) i negativni potencijal (Npt - vjerojatnost da naša ruka postane lošija, tj. da se vjerojatnost trenutno najjače ruke smanji). Za svaku moguću protivnikovu ruku (1081 moguća kombinacija), uzimajući u obzir 990 mogućih kombinacija rivera i turna, računa se broj ishoda u kojima smo bolji, lošiji ili izjednačeni. Postoji devet mogućih kombinacija između stanja na *flop*-u i na *river*-u (bolje-bolje, bolje-neriješeno, bolje-lošije, lošije-bolje, lošije-neriješeno, lošije-lošije, neriješeno-bolje, neriješeno-neriješeno, neriješeno-lošije).

Ako za primjer uzmemo ruku A♥Q♣ i na stolu je 4♣3♥J♥, uz pretpostavku da smo bili lošiji na *flop*-u, postoji 91 981 način da budemo bolji na *river*-u, 1 036 da budemo jednaki i u 346 543 slučaja i dalje ćemo biti lošiji. Ako smo na *flop*-u bili lošiji od nepoznate ruke, imamo 21% šanse da pobijedimo na *showdown*-u. Gledajući 2 karte unaprijed (*turn* i *river*) naša ruka ima pozitivni potencijal 0.208 i negativni potencijal 0.274. [14]

Tab. 3.2. A♥Q♣ / 4♣3♥J♥ potencijal

5 karata	7 karata			
	Bolje	Jednake	Lošije	Suma
Bolje	449005	3211	169504	621720=628*990
Jednake	0	8370	540	8910=9*990
Lošije	91981	1036	346543	439560=444*990
Suma	540986	12617	516587	1080190=1081*990

$$Ppot = \frac{HPUkupni_{lb} + \frac{HPUkupni_{ln}}{2} + \frac{HPUkupni_{nb}}{2}}{HPUkupni_{l} + HPUkupni_{n}} \quad (3\text{-}3)$$

gdje je:

- $HPUkupni_{lb}$ - broj načina na koje možemo biti lošiji na *flop*-u a bolji na *river*-u
- $HPUkupni_{ln}$ - broj načina na koje možemo biti lošiji na *flop*-u a jednaki na *river*-u
- $HPUkupni_{nb}$ - broj načina na koje možemo biti jednaki na *flop*-u a bolji na *river*-u
- $HPUkupni_{n}$ - broj načina na koje možemo biti jednaki na *flop*-u
- $HPUkupni_{l}$ - broj načina na koje možemo biti lošiji na *flop*-u
- Ppot - pozitivni potencijal ruke

$$Npot = \frac{HPUkupni_{bl} + \frac{HPUkupni_{nl}}{2} + \frac{HPUkupni_{bn}}{2}}{HPUkupni_{b} + HPUkupni_{n}} \tag{3-4}$$

gdje je:

- $HPUkupni_{lb}$ - broj načina na koje možemo biti lošiji na *flop*-u a bolji na *river*-u
- $HPUkupni_{nl}$ - broj načina na koje možemo biti jednaki na *flop*-u a lošiji na *river*-u
- $HPUkupni_{bn}$ - broj načina na koje možemo biti bolji na *flop*-u a jednaki na *river*-u
- $HPUkupni_{n}$ - broj načina na koje možemo biti jednaki na *flop*-u
- $HPUkupni_{b}$ - broj načina na koje možemo biti bolji na *flop*-u
- Npot - negativni potencijal ruke

3.3. Strategija ulaganja

Sve znanje, formule, informacije i slično služe isključivo kako bi mogli donijeti što bolju i efikasniju odluku kada je na nama red za igru. Iako takva odluka nije trivijalna, strategija ulaganja nam pomaže pri odluci treba li *fold*, *call*/*check* ili *bet*/*raise*. U obzir se uzima potencijal ruke, *pot odds* (šansa za najbolju ruku u odnosu na mogući dobitak), blefiranje, modeliranje protivnika i nepredvidljivost. Prvenstveno, strategija ulaganja se bazira na dvije stvari:

1.*Effective hand strength* (EHS) - vjerojatnost da trenutno imamo najbolju ruku ili da će se ona poboljšati do *showdown*-a [14]

$$EHS = HS_n + (1 - HS_n) * Ppot - HS_n * Npot \quad (3\text{-}5)$$

Ako na ruku A♥Q♣ / 4♣3♥J♥ primijenimo formulu (3-5), uz Ppot=0.208, Npot=0.274, HSn=58.5%, dobit ćemo da je EHS=50.8% što odgovara postotku dobivenom pomoću *poker odds* kalkulatora.

Postoji problem pri uključivanju negativnog potencijala ruke u formulu. Kada ulažemo, ne znamo hoće li protivnik uopće igrati, te u brojnim situacijama gdje u obzir uzmemo veliki negativni potencijal, bolja strategija je ulaganje ili dizanje (*bet/raise*) kako bi protivnika prisilili da odustane od ruke. Stoga, kada se EHS koristi pri donošenju odluke bolje je koristiti formulu (3-6). [14]

$$EHS = HS_n + (1 - HS_n) \times Ppot \quad (3\text{-}6)$$

2. *Pot odds* - šansa za pobjedu uspoređena s očekivanim dobitkom

Primjer 3.1. Ako je u *pot*-u 40 kuna, a nas košta 20 kuna da pratimo nečije ulaganje, imamo izglede dva naprema jedan (dobitak naspram cijene). Procijenjeni postotak pobjede mora biti veći ili jednak postotku dobivenom prema formuli (3-7). U ovom slučaju, procijenjeni postotak pobjede mora biti minimalno 33% kako bi odluka praćenja bili opravdana. Ako odluku donosimo na *river*-u, naša jačina ruke mora biti veća od *pot odds* ili jednaka. [14]

$$Pot\_Odds = \frac{cijena\_praćenja}{veličina\_pota + cijena\_praćenja} \quad (3\text{-}7)$$

gdje je:

- Pot_Odds - minimalni postotak koji opravdava odluku praćenja
- cijena_praćenja - iznos koji moramo platiti kako bi ostali u igri
- veličina_pota - iznos koji se nalazi u *pot*-u

Pot odds sami po sebi ne moraju nužno biti precizni. Recimo da u ruci imamo K♥10♥ i na stolu je A♥6♥4♣Q♦. Trenutno imamo 28% šanse da na *river*-u izađe srce ili dečko (13/46). *Pot odds*-i iznose 33% i stoga nam se ne isplati platiti 20kn kako bismo osvojili 40. Međutim, ako pretpostavimo da ćemo imati najjaču ruku (ako izađe srce ili dečko) i također pretpostavimo da će protivnik igrati do *showdown*-a gdje iz njega možemo izvući dodatnih 20kn, osvojit ćemo sveukupno 60kn. To naše izglede pretvara u tri naprema jedan (3:1) gdje naš procijenjeni postotak pobjede mora biti minimalno 25% i u tom slučaju nam se isplati platiti 20kn kako bismo vidjeli zadnju kartu. Implicirani izgledi (engl. *implied odds*) su temeljeni na mogućnosti da osvojimo više novca u budućem stadiju partije. Implicirani izgledi predstavljaju omjer između iznosa koji očekujemo da ćemo dobiti kada kompletiramo svoju ruku (u ovom slučaju kada na *river*-u izađe srce ili dečko) i iznosa koji će nas koštati da partiju odigramo do kraja.

Nepredvidljivost i obmana

Dva načina kako se strategija ulaganja može poboljšati su nepredvidljivost i obmana. Ako stalno postupamo isto u danim situacijama, protivnik vrlo lagano može stvoriti naš model. Mijenjanje strategije s vremena na vrijeme može protivniku znatno otežati posao modeliranja protivnika i navesti ga na pogreške uzrokovane nepreciznim modeliranjem. Obmana uključuje strategije kao što je blefiranje. Navodi protivnika na krive zaključke o trenutnom stanju igre. Obmana može također biti korištena u donošenju odluka koje možda ne vode do velike očekivane zarade u trenutnoj partiji već služe kao „lažno oglašavanje" kako bi došli do zarade u budućim partijama. Zaključno, svi potezi koje netko može vidjeti trebali bi imati dvostruko značenje kako se iz njih ne bi moglo ništa zaključiti.

3.4. Modeliranje protivnika

U igrama kao što su šah praćenje protivnika ne igra nikakvu uloga i stoga ignoriranje protivnika neće rezultirati lošijim rezultatima, kao što ni praćenje protivnika neće poboljšati vaš rezultat (u većini slučajeva). Za razliku od šaha, modeliranje protivnika ima veliku ulogu u pokeru i jedan je

od ključnih faktora u modeliranju dobrog poker agenta. Ako određen broj igrača podjednako dobro zna osnove pokera, ono što će bolje igrače razlučiti od lošijih je upravo sposobnost dobrog modeliranja protivnika. Dobra predodžba o onome što će protivnik napraviti u danoj situaciji je vrlo značajna informacija.

Metoda prikupljanja informacija o protivniku i korištenje tih informacija u svoju korist je složen problem. Može se reći da je modeliranje protivnika najveća prepreka u modeliranju poker igrača na razini profesionalnog poker igrača zato što nije jasno na koji način bi se statistike trebale prikupljati i kategorizirati.

Mogu se razlikovati dvije strategije u pokeru: optimalna strategija i strategija maksimiziranja. Optimalni igrač svoje odluke donosi na temelju teorije igre i vjerojatnosti zanemarujući kontekst. Igrač koji se služi strategijom maksimizacije u obzir uzima ponašanje protivnika u određenim situacijama i prilagođava se kako bi iskoristio njegove slabosti. Igrač koji uočava i prilagođava se slabostima protivnika osvajati će više od igrača koji to ne čini.

Protiv jakog konzervativnog igrača odustajanje s drugom najjačom rukom bi bila dobra odluka, dok protiv slabijeg igrača koji blefira često, to bi se smatralo greškom. U pokeru je česta pojava suboptimalnih igrača. Maksimizacijski program protiv suboptimalnog igrača imat će bolje rezultate od optimalnog programa protiv suboptimalnog igrača jer će maksimizacijski program bolje iskoristiti slabosti suboptimalnog igrača. Stoga, dobro modeliranje protivnika je nužno u modeliranju poker agent najviše razine.

Prema [14] tri glavne komponente kontroliraju djelovanje programa: procjena ruke (uz pomoć modela protivnika i stanja igre, generira vrijednosti koje ugrubo odgovaraju vjerojatnostima najjače ruke), strategija ulaganja (koristi vrijednosti generirane uz pomoć procjene ruke, model protivnika i stanje igre da procjeni najbolji potez) i modela protivnika (uz pomoć povijesti ulaganja protivnika, generira informacije o njegovoj strategiji ulaganja i mogućoj ruci protivnika).

IV. DOSADAŠNJA PRIMJENA STROJNOG UČENJA U MODELIRANJU TEXAS HOLD'EM AGENTA

4.1. Modeliranje protivnika pomoću neuronskih mreža i stabala odluke (Patrick McNally & Zafar Rafi, Northwestern Universitz Evanston, USA)

U razdoblju između 1995 i 2001, 10 milijuna partija je zabilježeno pomoću bota Observer koji je pratio online partije pokera. Ova baza podataka je puštena u javnost kako bi pomogla u razvoju umjetne inteligencije na području pokera.

Iako se zabilježene partije nisu igrale za stvarni novac, niti se može znati koliko su igrači bili kvalitetni, zbog svoje širine čine dobro mjesto za provođenje eksperimenata na području primjene strojnog učenja u pokeru. Baza je preuređena u ulazno-izlazne vektore koji su korišteni u treniranju. Cilj je bio predvidjeti četiri akcije: *fold*, *check*, *call* ili *bet*. Odlučeno je da se pri treniranju koriste sljedeće stavke:

- vrijednost ruke - zajedničke karte dostupne svima su rangirane od 0 do 1.0, gdje je skala u boji predstavljala 1.0 a high card 0.0.
- broj karata u boji - najveći broj karata na stolu u istoj boji
- broj poredanih karata - najveći broj karata čije su vrijednosti bile poredane, npr. na karte 4 5 6 9 10 broj poredanih karata bi iznosio 3 (4 5 6)
- prethodni potez igrač a- prethodni potez koji je igrač napravio
- pozicija igrača za stolom
- iznos potreban za *call*
- ukupan broj ulaganja igrača u partiji - ova stavka broji samo ulaganja kada nikakav iznos nije potreban za ostajanje u partiji
- ukupan broj dizanja (*rasie*-ova) igrača u partiji - broji ulaganja kada je manji iznos potreban kako bi ostali u partiji
- ukupan broj *call*-ova igrača u partiji - broji koliko puta je igrač pratio (*call*) u partiji
- agresivnost igrača - ova stavka je već određena za svakog igrača u bazi podataka, predstavlja omjer agresivnih poteza (*bet* i *raise*) i svih poteza u igračevoj povijesti igranja
- runda ulaganja - odvija li se ulaganje prije *flop*-a, na *flop*-u, *turnu* ili *river*-u

Stabla odluke naspram neuronskih mreža

Baza podataka je takva da je 36% svih poteza check. Ostali potezi su se pojavljivali u približnim iznosima ali rjeđe od *check*-a. Koristeći stavku „iznos potreban za *call*" može se trivijalno ukloniti pogađanje akcije *check* jer, ako je potreban određen iznos kako bismo ostali u partiji, opcija *check* nije moguća zbog pravila igre. Drugim riječima, znanje kako *check* nije moguć može se iskoristiti kako bi se sljedeći potez mogao lakše predvidjeti. „Iznos potreban za call" je sam po sebi dovoljan za dizajniranje stabla odluke s točnošću od 57%.

Kako ni jedna druga stavka postotak točnosti ne povećava dramatično (jedino agresivnost i prethodni potez igrača postotak dižu na 61% i 62.8%), može se zaključiti da je odnos među stavkama složen i međusobno ovisan. Kada su sve stavke korištene zajedno pri treniranju stabla odluke, točnost raste na 66.7%. Bez stavke agresivnosti točnost pada na 63.6% što može reći da predodžba o protivnikovoj agresivnosti znatno pomaže u predviđanju njegovog djelovanja. Oko 35% svih poteza odvija se prije *flop*-a, a u zadnjoj samo 13%. Razlog je očit (kako partija napreduje, sve više igrača odustaje), ali ovaj podatak i znači da velik dio baze podataka čini ponašanje prije *flop*-a, što se može znatno razlikovati od ponašanja u sljedećim rundama.

Otkriveno je da je znatno lakše predvidjeti ponašanje u kasnijim rundama nego ranijim. Tablica 3.1. pokazuje točnost stabla odluke i neuronske mreže trenirane po rundama, tj. dani su im podaci samo za pojedinu rundu.

Tab. 4.1. *Točnost stabla odluke i neuronske mreže po rundama prema [13]*

Runda	Točnost stabla odluke	Točnost neuronske mreže
Pre-flop	63,00%	55,80%
Flop	65,70%	65,19%
Turn	70,10%	70,65%
River	68,80%	69,00%

Iz tablice se može vidjeti da stabla odluke nadmašuju neuronske mreže u prvoj rundi te da tek za malo zaostaju u ostalim rundama. Međutim, neuronske mreže su kratko trenirane (mreža od 32 čvora je trenirana 5000 epoha) i daljnji napredak je moguć uz dulje treniranje. [13]

4.2. Libratus Poker AI

Ranije ove godine (2017.) došlo je do značajnijeg napretka u primijeni umjetne inteligencije u pokeru. Tim s Carnegie Mellon University-a razvio je kompjuterski program s ciljem da pobjedi ponajbolje igrače *Heads-Up No-Limit-Hold'em* pokera (Heads-Up znači jedan na jedan).

U razdoblju od 11.siječnja 2017. do 30.siječnja 2017. održan je turnir u kojem su sudjelovala četiri vrhunska Heads-Up igrača: Dong Kim, Jason Les, Jimmy Chou i Daniel McAulay. U tom razdoblju igralo se 120 000 partija - 30 000 po igraču. Svaka partija je igrana tako da svaki igrač ima 20 000 čipova gdje su blindovi bili 50/100. Ovo je omogućilo da se svaka partija igrala s veličinom zalihe od 200 velikih *blind*-ova - veličina koja je omogućila dosta prostora za različite strategije u svakoj partiji.

Kako bi se smanjio faktor sreće uvedena su dva pravila

1. Sve početne ruke su bile „zrcaljene" - Ako bi igrač A dobio par aseva protiv para kraljeva protiv programa, igrač B bi dobio par kraljeva protiv para aseva za drugim stolom.
2. Bez all-in-ova - Ako bi došlo do *all-in*-a prije rivera, ni jedna karta više ne bi bila podijeljena i svaki igrač bi dobio čipove u skladu sa svojom šansom za pobjedu. Npr. ako je igrač imao izglede 70/30 na turnu u *all-in* situaciji osvojio bi 70% pota, a protivnik bi dobio 30%. Ovo je smanjilo faktor sreće.

Nakon 20 dana igranja i 120 000 partija, Libratus je pobijedio svakog igrača i ostvario prihod od 14.72$ po partiji.

Tab. 4.2. *Ukupni gubici i gubici po partiji svakog igrača*

Igrač	Gubitak ($)	Po partiji ($)
Dong Kim	-85 649	-2.85
Jimmy Chou	-522 857	-17.43
Jason Les	-880 087	-29.34
Daniel McAuley	-277 657	-9.26
Suma / Prosjek	-1 766 250	-14.72

Princip rada Libratusa

Libratus se bazira na tri različita sustava koja rade zajedno. Većinom je baziran na podržanom učenju i algoritmu Counterfactual Regret Minimization (prema [19] odluke se donose na temelju prijašnjih akcija i njihovih rezultata) s ciljem pronalaska savršene strategije. Savršena strategija ili Nashova ravnoteža (engl. *Nash equilibrium*) je skup strategija, jedan za svakog igrača, takav da niti jedan igrač ne želi odstupati od njega bez obzira na postupke drugog igrača.

Program je igrao sam protiv sebe počevši od nule bez ikakvih predefiniranih pravila osim pravila pokera. Prva partija je bila potpuno nasumična. *Bet, call, check, raise* sve je imalo isto značenje. Libratus je analizirao svoju strategiju naspram strategije protivnika. Budući da je nagrada u pokeru lako mjerljiva (novac), *bot* bi nakon svake akcije uočio koji potez mu je mogao donijeti više novca te ga pohranio za buduće vrijeme kada se nađe u istoj situaciji. Program je dosegao razinu igre na nivou najboljih igrača, potencijalno i bolju razinu. Koristio je širi spektar ulaganja i uveo je svojstvo nasumičnosti u ta ulaganja te tako protivniku otežao zadatak procjene ruke. „Mi programu dajemo opis igre. Mi mu ne govorimo kako da igra.” - Noam Brown, glavni tvorac programa.

Ono što Libratusa razlikuje od drugih poker *bot*-ova je njegova druga komponenta zvana *end-game solver* [20]. *End-game solver* koristi procjene milijune milijuna odigranih ruku prve komponente kako bi pogodio što protivnik ima. Svaki put kada protivnik napravi potez, *end-game solver* ponovo računa, koristeći novo prikupljene informacije u svojoj strategiji. Ovakva rekurzivna metoda računanja omogućuje znatno smanjenje mogućnosti, na temelju vjerojatnosti da će protivnik napraviti određeni potez s određenom rukom.

End-game solver također *bot*-a čini odvažnim. Bot neće biti oprezan kako bi sačuvao novac niti u obzir uzima koliko je dosad osvojio ili izgubio. Libratusov cilj je zaraditi što više novca bez obzira na rizik. Ako Libratus ima 10% šanse da osvojio 20 000 kuna naspram 90% da osvoji ništa ili garantiranih 2000kn, on će uzeti rizik od 10%. Ljudi ne pristupaju igri kao Libratus. Čak i da znaju da je potez ispravan strah od gubitka ih sprječava da ga naprave.

Zadnja komponenta Libratusa ima ulogu pronalaska rupa u njegovoj obrani. Cijelo vrijeme, algoritam u pozadini traži koje slabosti je protivnik pronašao u njegovoj strategiji i koliko često ih je iskorištavao. Program će zatim te slabosti postaviti kao prioritet i tražiti bolje strategije u tim situacijama.

Za treniranje Libratusa korišteno je superračunalo Bridges. Ono je oko 30 000 puta brže od prosječnog stolnog računala, ima 274 terabajta RAM-a i košta 9.65 milijuna dolara.

V. ZAKLJUČAK I BUDUĆI RAD

Primjena strojnog učenja u pokeru je još u razvoju kao i ta dva područja zasebno. Libratus je samo početak. Iako je ostvario zapanjujuće rezultate, Libratus je napravljen samo za Heads-Up Poker koji nije toliko popularan kao normalni *Texas Hold'em Poker* sa šest do deset igrača po stolu. Nadalje, Libratus je toliko kompleksan da je potrebna izravna veza s ogromnim superračunalom gdje i dalje igra dosta sporo. Iz tih razloga ne postoji izravna opasnost da bude zloupotrebljavan u kasinu ili online igranju. Međutim, očekuje se da neće proći puno prije nego slične varijante budu dostupne na našim pametnim telefonima. Također, botovi čak ni ne moraju igrati na razini savršene strategije kako bi bili bolji od prosječnog poker igrača. Iako poker nije razriješen u cijelosti, razriješen je dovoljno da prosječan bot može pobijediti prosječnog poker igrača.

Na temelju ovog rada, uz dodatno izučavanje područja strojnog učenja, kao i područja pokera planirana je (u bližoj ili daljoj budućnosti) vlastita izrada poker agenta na zadovoljavajućoj razini efikasnosti.

VI. LITERATURA

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, Hsuan-Tien Lin, Learning from data, AMLbook, United States of America, 2012.

[2] Web stranica za online trgovanje, http://www.njuskalo.hr/prodaja-stanova [16.3.2017.]

[3] A. Ng, Lecture 1 Machine Learning, Stanford Computer Science Depratment, https://www.youtube.com/watch?v=UzxYlbK2c7E&t=2056s, [10.4.2017.]

[4] Hrvatski novčanski zavod, http://www.hnz.hr, [14.4.2017.]

[5] B. Dalbelo Bašić, J.Šnajder, Strojno učenje – bilješka za predavanja (v2.2), Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu, ak. god. 2011/12. r

[6] L.Serrano, A friendly introduction to machine learning, https://www.youtube.com/watch?v=IpGxLWOIZy4&t=1336s, [18.4.2017.]

[7] Udacity, Inc, Gradient-Descent Artificial intelligence for robotics, https:// www.youtube.com/watch?v=umAeJ7LMCfU, [10.4.2017.]

[8] Gradient, http://www.dictionary.com/browse/gradient [10.4.2017.]

[9] J. Browniee, Naive Bayes for Machine learning, travanj 2016., http:// machinelearningmastery.com/naive-bayes-for-machine-learning/, [22.6.2017.]

[10] Chapter 3, Decision Tree Learning, Princeton, http://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf, [14.6.2017.]

[11] B. Dalbelo Bašić, M. Ćupić, J. Šnajder, Umjetne neuronske mreže, Zagreb, 2008.

[12] S. Welch, Neural Networks Demystified , https://www.youtube.com/watch?v=bxe2T-V8XRs, [6.6.2017.]

[13] P. McNally, Z. Rafii, Opponent modeling in poker using machine learning techniques, Northwestern University Evanston, USA, http://www.cs.northwestern.edu/~ddowney/courses/349_Fall2008/projects/poker/Poker.pdf, [20.6.2017.]

[14] D. Billings, D. Papp, J. Schaeffer, D. Szafron, Opponent Modeling in Poker, Department of Computing Science, University of Alberta, USA

[15] Kombinatorika, http://forum.matemanija.com/viewtopic.php?f=42&t=1142 , [29.6.2017.]

[16] Poker odds kalkulator, https://ca.pokernews.com/poker-tools/poker-odds-calculator.htm, [10.6.2017.]

[17] D. Skalinsky, M. Malmuth, Hold'em Poker for Advanced Players, Two Plus Two Publishing LLC,1994.

[18] A. Klöhn, Libratus Poker AI Beats Humans for $1.76m; Is End Near?, PokerListings, the all-in poker guide, http://www.pokerlistings.com/libratus-poker-ai-smokes-humans-for-1-76m-is-this-the-end-42839 , [20.6.2017.]

[19] T. W. Neller, Marc Lanctot, An Introduction to Counterfactual Regret Minimization, http://modelai.gettysburg.edu/2013/cfr/cfr.pdf, [21.6.2017.]

[20] N. Brown, T. Sandholm, Safe and Nested Endgame Solving for Imperfect-Information Games, Carnegie Mellon University, https://www.cs.cmu.edu/~sandholm/safeAndNested.aaa17WS.pdf, [21.6.2017.]

VII. KAZALO POJMOVA

flop - druga runda ulaganja, tri zajedničke karte koje se otkrivaju nakon prve runde ulaganja

pre-flop - prvi stadij igre, prva runda ulaganja, svaki igrač dobiva dvije karte i započinju ulaganja

turn - treći stadij igre, treća runda ulaganja, četvrta zajednička karta koja je otkrivena nakon *flop*-a

river - posljednji stadij igre, četvrta runda ulaganja, peta i zadnja zajednička karta koja je otkrivena

hole - dvije karte poznate samo igraču koji ih drži, dodijeljene prije flopa (*pre-flop*)

bet - ulaganje koje zahtijeva od drugog igrača zahtijeva da plati taj iznos ako želi ostati u partiji

partija - jedno dijeljenje sa svim stadijima igre (ako igrači ne odustanu ranije)

zajedničke karte - engl. c*ommunity cards*, karte na stolu vidljive svima, *flop, river* i *turn*

showdown - stadij igre u kojem svi igrači koji su i dalje u igri pokazuju svoje karte kako bi se moglo vidjeti tko ima jače karte i tko nosi *pot*

pot - novac za koji se trenutno igra, svaki puta kada netko uloži novac on ide u *pot*

dealer - djelitelj, pozicija za stolom po kojoj se gledaju ostale pozicije

blind - obvezan ulog koji svaki igrač mora platiti u prvoj rundi ako želi biti u ruci

mali *blind* - engl. *small blind,* igrač prvi slijeva do *dealer*-a, on mora dati polovinu iznosa velikog *blind-a* bez obzira želi li ostati u igri ili ne

veliki *blind* - engl. *big blind,* igrač drugi slijeva do dealer-a, on mora dati puni iznos *blind-a* bez obzira želi li ostati u igri ili ne

check - ova opcija je moguća jedino ako nije potreban određeni iznos kako bi igrač ostao u igri, odabirom opcije check igrač odlučuje da ne želi uložiti ništa ali i dalje želi ostati u igri

fold - igrač odustaje od igre/partije

raise - na nečiji uloženi iznos igrač ulaže dodatni iznos, veći od njegovog

call - praćenje, nakon što je netko uložio određeni iznos (*bet*-ao) igrač ga plaća kako bi ostao u igri

income rate - stopa prihoda, mjera povrata investicije

offsuit - skraćeno „o" (npr. AKo), karte u različitoj boji

suited - skraćeno „s" (npr. AKs), karte u istoj boji

No-fold'em - način igranja pokera gdje nitko ne folda

button - ujedno i pozicija dealer-a, dealer i button su ista pozicija

effectie hand strength - EHS, vjerojatnost da trenutno imamo najbolju ruku ili da će se ona

poboljšati do *shodown*-a

pot odds - šansa za pobjedu uspoređena s očekivanim dobitkom

implicirani izgledi - engl. i*mplied odds*, nadograđeni pot odds temeljeni na mogućnosti da osvojimo više novca u budućem stadiju igre

heads-up - jedan na jedan

all-in - igrač ulaže sve svoje novce

SAŽETAK

Cilj ovog ovog rada je istražiti i pokazati načine primjene strojnog učenja u svrhu modeliranja *Texas Hold'em Poker* agenta koji će igrati na zadovoljavajućoj razini. U prvom poglavlju rada objašnjen je pojam strojnog učenja zajedno s njegovim granama i metodama rješavanja problema određene prirode. *Texas Hold'em Poker* je kompleksna igra jednostavnih pravila što je pokazano u drugom poglavlju. Objašnjeni su najvažniji koncepti *Texas Hold'em Pokera* čije znanje može biti primjenjeno kako bi model igrača bio što efikasniji. Konačno, navedeni su neki od dosadašnjih pokušaja rješavanja ovog problema zajedno s načinom njegovog rješavanja i mjerom uspjeha, tj. rezultatima.

Ključne riječi: *Texas Hold'em Poker*, poker, strojno učenje, neuronske mreže, podržano učenje, Libratus, modeliranje protivnika, jačina ruke

ABSTRACT

Machine learning based on logic and probability

The aim of this paper is to explore and demonstrate how to use machine learning in order to simulate/ make a model of a Texas Hold'em Poker agent that will play at a satisfactory level. In the first chapter of the paper, the concept of machine learning is explained, along with its branches and methods of solving different problems. Texas Hold'em Poker is a complex game with simple rules, as it is shown in the second chapter. Furthermore, the most important concepts of Texas Hold'em Poker are explained. The knowledge of these concepts can be applied to make a model of the player more efficient. Finally, some of the past attempts to solve this problem are mentioned together with the way it is solved and the measure of success, i.e. the results.

Keywords: Texas Hold'em Poker, Poker, Machine Learning, Neural Networks, Learning Support, Libratus, Opponent Modeling, Hand Strength

ŽIVOTOPIS

Trpimir Zovak rođen je 19.9.1995. u Slavonskom Brodu i trenutno živi u Donjim Andrijevcima. Završio je Osnovnu školu “Viktor Car Emin” u Donjim Andrijevcima, te upisao prirodoslovno-matematički smjer u Gimnaziji “Matija Mesić” u Slavonskom Brodu. Nakon završene gimnazije upisuje predidplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.