**Univerzitet u Banjoj Luci**

**Elektrotehnički fakultet**

**Katedra za automatiku**

**Metodi vještačke inteligencije**

Izvještaj o urađenom projektnom zadatku

**Neuronske mreže**

Student:

Stefan Milinčić, 1182/15

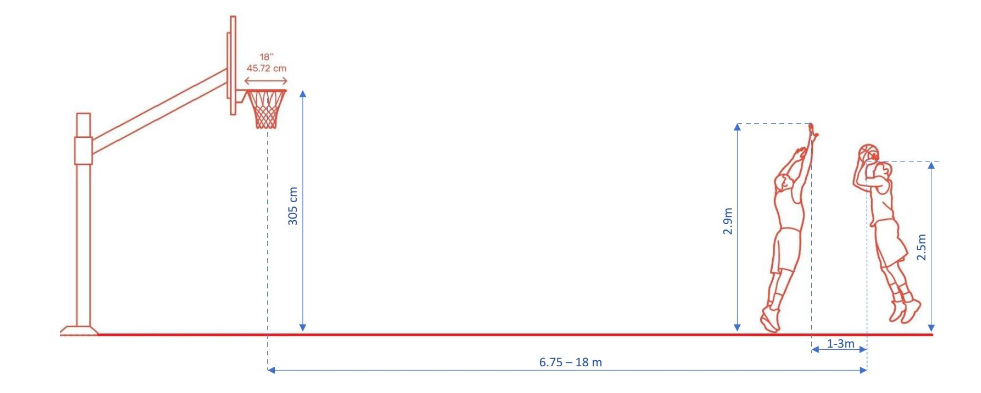
# Opis problema

Humanoidni robot šutira za tri poena na košarkaškom terenu. Sve dimenzije su prikazane na slici Slika 1. Potrebno je projektovati i obučiti neuronsku mrežu koja će da odredi brzinu i ugao kojim robot treba da izbaci loptu pri različitim udaljenostima od koša, kao i pri različitim udaljenostima košarkaša od robota koji pokušava da blokira šut robota, tako da se uvijek postigne pogodak. Pogoci od tablu nisu dozvoljeni. Jednačina kosog hica je

(1.1)

gdje je:

* x – domet hica
* y – visina hica
* α – ugao izbačaja hica
* v – brzina izbačaja hica



Slika 1. *Skica košarkaškog terena sa potrebnim dimenzijama i rastojanjima*

# Rješenje

Neuronske mreže su računarski sistemi koji su donekle inspirisani biološkim neuronskim mrežama od kojih je sačinjen ljudski mozak. Takvi sistemi „uče“ (popravljaju preformanse) zadatke razmatrajući konkretne primjere. Na primjer, oni mogu da nauče da identifikuju slike koje sadrže mačke putem analiziranja primjera slika koje su bile označene kao „mačka“ ili „nije mačka“ i da zatim koriste te rezultate za identifikaciju mačaka na drugim slikama. Oni to čine bez bilo kakvog ranijeg znanja o mačkama, npr. da imaju krzno, repove, brkove, mačija lica, itd. Umjesto toga oni evoluiraju svoj sopstveni set relevantnih karakteristika iz materijala za učenje koji oni obrade. Jedna neuronska mreža je bazirana na kolekciji povezanih čvorova zvanih neuroni (pojednostavljene verzije bioloških neurona u mozgu). Svaka veza između neurona može da transmituje signal od jednog do drugog, tj. koristiće se potpuno povezana topologija mreže, što znači da postoji veza između svih neurona u susjednim slojevima. Neuron koji primi signal može da ga obradi i zatim preda ishod neuronima sa kojima je povezan. U običajenim implementacijama, signal na vezi između neurona je realan broj i izlaz svakog neurona se računa koristeći aktivacione funkcije sume ulaza. Neuroni i veze imaju težine koje se podešavaju tokom učenja. Težina pojačava ili smanjuje jačinu signala na vezama. Neuroni su organizovani u slojeve. Različiti slojevi mogu da izvode različite vrste transformacija na svojim ulazima. Signali putuju od ulaznog (prvog), preko skrivenog, pa do izlaznog (zadnjeg) sloja. Slojevi u ovoj mreži:

* ***Ulazni sloj*** : Broj neurona u ulaznom sloju je određen brojem ulaza posmatranog sistema. U ovom slučaju se posmatra funkcija dvije promjenljive, pa sistem ima dva ulaza, samim tim biramo da mreža ima dva neurona u ulaznom sloju.
* ***Skriveni sloj*** : Skriveni sloj mreže može da sadrži veći broj slojeva. Prema tome, prvo je potrebno odrediti koliko slojeva će sadržati skriveni sloj mreže. U ovom slučaju je odabrano da postoji jedan skriveni sloj, pri čemu taj sloj sadrži 4 neurona. Broj neurona za ovaj sloj je određen eksperimentalno kroz iterativan proces.
* ***Izlazni sloj*** : Broj neurona u izlaznom sloju određen je brojem izlaza posmatranog sistema. U ovom slučaju, broj izlaznih neurona je dva.

Rješenje problema je realizovano kao konzolna aplikacija (napisana u JAVA programskom jeziku u Eclipse razvojnom okruženju) koja implementira neuronsku mrežu. Program čita trening parove iz teksutalnog fajla **treningParovi.txt** . Prilikom završetka obrade jednog trening para, na konzlu se ispisuje dobijeni ugao i brzina , željene vrijednosti ugla i brzine, kao i visina hica. Program se izvršava dok god se svi trening parovi ne obrade. Prije nego što se počne sa opisom klasa potrebno je uvesti odgovarajuće notacije označavanja:

Tabela 2.1 *Prikaz notacije označavanja*

|  |  |
| --- | --- |
| **Oznaka** | **Značenje** |
|  | Ukupan ulaz i-tog neurona, q-tog sloja |
|  | Izlaz i-tog neurona, q-tog sloja |
|  | Težinski fakor između j-tog neurona (q-1)-vog sloja i i-tog neurona q-tog sloja |
|  | Privremena greška |
| E | Kumulativna greška |
|  | Maksimalna dozvoljena greška |

Kao rješenje je priložena datoteka u kojoj se nalazi Java kod sa jednim tekstualnim fajlom i sledećim klasama:

* **Neuron** – predstavlja klasu koja opisuje jedan neuron u mreži. Sadrži dvije promjenljive **id** i **sloj** tipa int, kao i dvije promjenljive **ulaz** i **izlaz**, tipa double. Id ustvari predstavlja redni broj neurona u mreži i na osnovu njega je izvršeno povezivanje neurona međusobno, sloj predstavlja sloj na kome se neuron nalazi (1 - ulazni, 2 - skriveni, 3 - izlazni), ulaz i izlaz predstavljaju ulazne i izlazne vrijednosti neurona. Pošto su sve promjenljive privatne, klasa sadrži i odgovarajuće setere i getere za svaku od promjenljivih. Takođe, u klasi postoji i podrazumijevani konstruktor, kao i konstruktor sa četiri argumenta koji služe za kreiranje objekata ove klase.
* **TezinskiFaktor** – klasa koja predstavlja težinski faktor između dva neurona. Sadrži jednu matricu **matricaTezinskihFaktora** tipa double veličine 8x8 koja predstavlja kao i što samo ime kaže matricu koja sadrži težinske faktore između povezanih neurona. Ova matrica je simetrična jer se iste vrijednosti nalaze na pozicijama i (npr. pozicije i imaju iste vrijednosti), dok se po glavnoj dijagonali nalaze nule. Koordinata x ustvari predstavlja id neurona koji je u nižem sloju, dok koordinata y predstavlja id neurona koji je u susjednom sloju od koordinate x tj. u sledećem višem sloju od sloja u kom je koordinata x. Ispod glavne dijagonale tj. u donjoj trouganoj matrici su stvari obrnute. Klasa sadrži podrazumijevani konstruktor koji služi da inicijalizuje matricu nulama. Od metoda postoji metoda **odredjivanjePocetnihTezinskihFaktoraIzmedjuNeurona** tipa void koja prima dva argumenta tipa Neuron. Kao što i samo ime kaže, ova metoda služi za određivanje početnih težinskih faktora između svaka dva povezana neurona. U metodi postoji java ugrađena klasa Random koja uz pomoć podrazumijevanog konstruktora kreira objekat rand koji nam generiše slučajne vrijednodnosti iz određenog opsega. Taj opseg je određen iskustveno i on je neki cijeli broj na intervalu . Nakon određenih vrijednosti, te vrijednosti se postavljaju u matricu na pozicije koje su dobijene iz argumenta metode. Takođe postoji i metoda **odredjivanjeKorigovanihTezinskihFaktora** koja je tipa void, a prima dva argumenta tipa Neuron. Kao i što sam naziv metode kaže, ova metoda određuje korigovane težinske faktore između svaka dva povezana neurona. U metodi postoji pomoćna promjenljiva delta koja je tipa double i određuje delta faktor neurona iz nižeg sloja na osnovu jedne switch petlje i dva proslijeđena argumenta (neurona). Nakon što se odredi delta faktor vrši se određivanje korekcije težinskih faktora na osnovu formule (2.1):

(2.1)

Nakon što se odredi korekcija težinskih faktora, određuju se nove vrijednosti težinskih faktora na osnovu formule (2.2) tj. u matricu matricaTezinskihFaktora na odgovarajuću poziciju se dodaje prethodno izračunata korekcija.

(2.2)

* **TreningPar** – klasa koja predstavlja postupak treniranja (obučavanja) neuronske mreže. Od promjenljivih sadrži **ulaz, izlaz, privremenaGreska1, privremenaGreska2, deltaN1, deltaN2, deltaN3, deltaN4, deltaN5, deltaN6, deltaN7, deltaN8** tipa double. Svaka od ovih nabrojanih promjenljivih ima smisleno ime, pa nema potrebe za dodatnim objašnjavanjem. Klasa sadrži metodu **odrediUlaziIzlaz** koja je tipa void i koja je bez argumenata. Kao što i samo ime metode kaže ona služi za određivanje ulaza i izlaza svih neurona. Potrebno je odrediti izlaze neurona ulaznog sloja, ali pošto je aktivaciona funkcija ovog sloga a(x) = x, onda su izlazi ovih neurona jednaki posmatranim ulazimai oni se i ne određuju u ovoj metodi. Zatim, potrebno je odrediti ukupne ulaze svih neurona u skrivenom i izlaznom sloju. Metoda sadrži pomoćne promjenljive ulaz1, ulaz2, ulaz3, ulaz4, ulaz5, ulaz6 koje su tipa double i koje se računaju pomoću formule (2.3):

(2.3)

Nakon određenih ulaza za sve neurone u skrivenom i izlaznom sloju, potrebno je odrediti i izlaze svih neurona u skrivenom i izlaznom sloju. Izlazi se određuju na osnovu aktivacione funkcije (2.4):

(2.4)

Nakon određeneih ulaza i izlaza, oni se za sve neurone setuju uz pomoć odgovarajućih setera za ulaze i izlaze. Postoji i pomoćna metoda **racunajAktivacionuFunkciju** čiji je povratni tip double, a kao argument prima promjenljivu ulaz tipa double. U većini slučajeva aktivacione funkcije neurona jednog sloja su iste za sve neurone u posmatranom sloju. Izbor aktivacionih funckija je stvar iskustva i eksperimentisanja sa mrežom, a u našem slučaju je izabrana unipolarna sigmoidalna funkcija (2.4). Takođe, u klasi se nalazi još jedna pomoćna metoda **izvodAktivacioneFje** koja vraća povratni tip double, a kao argument prihvata promjenljivu ulaz tipa double. Ova metoda računa izvod unipolarne sigmoidalne aktivacione funkcije po formuli (2.5):

(2.5)

Pomoćna metoda **racunajPrivremenuGresku** je povratnog tipa void i ona je bez argumenata. Privremena greška se računa kao najmanje srednje kvadratno odstupanje između trenutno izračunatog vektora izlaza i željenog vektora izlaza (tačnog izlaza), a predstavljena je preko formule (2.6):

(2.6)

Pomoćna metoda **racunajKumultavnuGresku** je metoda bez argumenata, a njen povratni tip je void. Kumulativna greška se računa kao zbir trenutne kumulativne greške i trenutne privremene greške i predstavljena je preko formule (2.7):

(2.7)

Metoda **odrediDeltaFaktor** je metoda koja računa delta faktore za sve neurone u mreži. Ona je povratnog tipa void i ne sadrži argumente. Sadrži dvije pomoćne promjenljive ukupanUlaz1 i ukupanUlaz2 tipa double koje služe da se izračuna ukupan ulaz posmatranog neurona za izlazni i skriveni sloj. DeltaN8 i deltaN7 su promjenljive u koje se smještaju delta faktori izlaznog sloja i računaju se po formuli (2.8), dok promjenljive deltaN6, deltaN5, deltaN4 i deltaN3 služe da se izračunaju delta faktori skrivenog sloja, a oni se računaju po formuli (2.9):

(2.8)

(2.9)

Metoda **citajIzFajlaTreningParove** je pomoćna metoda koja vrši čitanje trening parova iz fajla treningParovi.txt. Metoda je bez argumetana, a povratni tip joj je void. Metoda sadrži java ugrađene klase FileReader i BufferedReader za pronalaženje fajla na fajl sistemu, zatim jednu while petlju koja kupi liniju po liniju iz fajla i vrši parsiranje odgovarajućih vrijednosti. Udaljenost robota od koša se smješta u ArrayList-u x1, udaljenost košarkaša od robota se smješta u ArrayList-u x2, željeni ugao se smješta u ArrayList-u alfa, dok se željena brzina smješta u ArrayList-u v. Pomoćna metoda **racunajY** nam vrši računanje visine hica. Povratni tip metode je double, a kao argumente prima tri promjenljive x, v i alfa tipa double. Visina hica se računa pomoću formule (1.1).

* **Main** – klasa je klasa u kojoj se vrši izvršavanje cijelog programa i samog algoritma. Sadrži promjenljive **tezinskiFaktor** i **brojIteracija** tipa int, **Emax**, **privremenaGreska**, **E1**, **zeljenaVrijednostIzlaza1** i **zeljenaVrijednostIzlaza2** tipa double. Izbor težinskog faktora je stvar iskustva, jer izbor velikog težinskog faktora obezbjeđuje brzu konvergenciju, koja je manje sigurna, dok izbor malog težinskog faktora obezbjeđuje sporiju, ali sigurniju konvergenciju. Zbog toga vrijednost promjenljive tezinskiFaktor je postavljena na 5. Promjenljivoj brojIteracija se dodjeljuje vrijednost od 50000 i to predstavlja broj iteracija koji će se sigurno izvršiti ukoliko ne bude ispunjen uslov za prekid petlje tj. da je E<Emax. Promjenljiva Emax predstavlja maksimalnu dozvoljenu grešku tj. ona predstavlja maksimalno dozvoljenu razliku između željenih izlaza i generisanih izlaza korištenjem neuronske mreže. Što je ova vrijednost manja, preciznost je već, pa je zbog tog vrijednost promjenljive Emax postavljena na 0.000000001. Promjenljiva privremenaGreska nam služi za čuvanje vrijednosti privremene greške koja se računa pomoću već opisane metode racunajPrivremenuGresku. Promjenljiva E1 je namijenjena za čuvanje vrijednosti kumulativne greške koja se računa pomoću metode racunajKumulativnuGresku. Nakon definisanja određenih promjenljivih koje su potrebne za izvršavanje algoritma, prelazi se na kreiranje neurona. Uz pomoć odgovarajućeg konstruktora kreiraju se sledeći neuroni: **n1** i **n2** su neuroni ulaznog sloja, **n3**, **n4**, **n5** i **n6** su neuroni skrivenog sloja, a **n7** i **n8** su neuroni izlaznog sloja. Kreiraju se i odgovarajuće ArrayList-e koje služe za čuvanje vrijednosti pročitanih iz fajla treningParovi.txt. ArrayList-a **x1** se koristi za čuvanje vrijednosti koje se odnose na udaljenost robota od koša, lista **x2** se koristi za udaljenost košarkaša od robota, lista **alfa** za čuvanje željenog ugla pod kojim se lopta izbacuje, a lista **v** za željenu brzinu lopte prilikom šuta. Nakon kreiranih svih promjenljivih, neurona i listi prelazi se na izvršavanje samog algoritma. Prvo se poziva pomoćna funkcija citajIzFajlaTreningParove koja nam popunjava prethodno kreirane ArrayList-e sa odgovarajućim vrijednostima iz fajla sa trening parovima. Nakon toga se određuju početni težinski faktori između svih povezanih neurona. To se vrši pomoću metode odredjivanjePocetnihTezinskihFaktoraIzmedjuNeurona kojoj se prosleđuju kao argumeti dva neurona (n1-n3, n1-n4, n1-n5, n1-n6, n2-n3, n2-n4, n2-n5, n2-n6, n3-n7, n3-n8, n4-n7, n4-n8, n5-n7, n5-n8, n6-n7, n6-n8) između kojih se vrši određivanje faktora. Zatim se ulazi u while petlju koja se izvršava dok god se ne završe sve iteracije i da je ispunjen uslov E>Emax, u suprotnom se završava petlja. Nakon toga se ulazi u još jednu while petlju koja nam ustvari predstavlja petlju koja uzima trening parove i željene vrijednosti iz već inicijalizovanih listi. Ulazi i izlazi neurona n1 i n2 iz ulaznog sloja se postavljaju na odgovarajuće vrijednosti koje su dobijene iz listi, kao i željene vrijednosti izlaza za ugao i brzinu. Nakon dobijenih vrijednosti za trening par prelazi se na računanje ulaza i izlaza za ostale neurone u mreže. To se izvršava pomoću metode odrediUlaziIzlaz. Potrebno je pozvati i metodu racunajPrivremenuGresku kako bi se izračunala privremena greška i kako bi se moglo nastaviti sa izvršavanjem algoritma i omogućiti ulazak u treću while petlju. Treća while petlja se izvršava sve dok je ispunjen uslov da je privremena greška veća od maksimalne dozvoljene greške. Kada se uđe u ovu petlju potrebno je izvršiti propagaciju greške unazad. Jedan od koraka da se to izvrši je poziv pomoćne metode racunajKumulativnuGresku koja vrši računanje kumulativne greške, nakon koje se poziva metoda odrediDeltaFaktor koja nam računa delta faktore za sve neurone u mreži i ustvari predstavlja pripremu za računanje korigovanih težinskih faktora. Posle uspješno određenih delta faktora prelazi se na računanje korigovanih težinskih faktora. Računanje korigovanih težinskih faktora se vrši pomoću već opisane metode odredjivanjeKorigovanihTezinskihFaktora kojoj se takođe kao argument prosleđuju dva neurona (n1-n3, n1-n4, n1-n5, n1-n6, n2-n3, n2-n4, n2-n5, n2-n6, n3-n7, n3-n8, n4-n7, n4-n8, n5-n7, n5-n8, n6-n7, n6-n8) između kojih se vrši određivanje korigovanih težinskih faktora. Nakon tako određenih korigovanih težinskih faktora vrši se ponovno računanje ulaza i izlaza za sve neurone u mreži sa tako izmijenjenim faktorima. To se ponovo radi sa metodom oderdiUlaziIzlaz. Posle tako izračunatih novih ulaza i izlaza potrebno je uz pomoć metode racunajPrivremenuGresku ponovo odrediti i privremenu grešku kako bi se petlja mogla pravilno izvršavati izvršavati. Nakon završene petlje, na konzolu se ispisuju dobijene vrijednosti za svaku odrađenu iteraciju, tj. ispisuje se dobijeni ugao i brzina i željeni ugao i brzinu za svaki od trening parova, respektivno. Ovo se izvršava sve dok je ispunjen uslov za prvu definisanu while petlju.
* **treningParovi.txt** – vrijednosti ugla su dobijene na osnovu slučajnog izbora iz intervala tj. , dok je vrijednost brzine dobijen na osnovu sistema jednačina (2.10), (2.11), (2.12), (2.13):

(2.10)

gdje je:

* + x – udaljenost robota od koša
  + α – ugao izbačaja hica

(2.11)

gdje je:

* x - udaljenost košarkaša od robota
* – ugao izbačaja hica

(2.12)

(2.13)

Nakon rješavanja ovog sistema jednačina dobiće se odgovarajuća jednačina za brzinu. Potrebno je ponovo izračunati sistem jednačina sa dobijenom jednačinom, izabranim uglom i jednačinom (2.11) sa odgovarajućom udaljenosti košarkaša od robota.

Tabela 2.2 *Prikaz vrijednosti za trening parove i dobijeni rezultati*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Udaljenost robota od koša**  **(m)** | **Udaljenost košarkaša od robota (m)** | **Dobijena vrijednost ugla (rad)** | **Željena vrijednost ugla (rad)** | **Dobijena vrijednost brzine ()** | **Željena vrijednost brzine ()** |
| 6.75 | 1 | 1.3134521579865241 | 1.309 | 11.607599456322824 | 11.6076 |
| 6.75 | 2 | 1.1300129307539253 | 1.13446 | 9.438130352476398 | 9.43813 |
| 6.75 | 3 | 0.955472471277388 | 0.959931 | 8.588155586391762 | 8.58821 |
| 12.37 | 1 | 1.2175651463391353 | 1.22173 | 13.826109674856513 | 13.8277 |
| 12.37 | 2 | 1.0427301371624478 | 1.0472 | 11.95790001672756 | 11.9579 |
| 12.37 | 3 | 0.8766032347844874 | 0.872665 | 11.263935322560306 | 11.266 |
| 18 | 1 | 1.1823776859800441 | 1.18682 | 16.02100000003648 | 16.021 |
| 18 | 2 | 0.9120185348061641 | 0.907571 | 13.617899999986975 | 13.6179 |
| 18 | 3 | 0.7809485492684428 | 0.785398 | 13.4499999963888 | 13.45 |

# Zaključak

Primarni cilj neuronskih mreža je bilo rješavanje problema na isti način na koji bi ljudski mozak to činio, ali tokom vremena sve veća pažnja je usmjeravana na specifične zadatke, što je dovelo do odstupanja od biologije. Neuronske mreže se koriste na raznim zadacima, uključujući računarski vid, prepoznavanje govora, mašinsko prevođenje, filtriranje društvenih mreža, igranje igara na tabli, video igara, medicinsku dijagnostiku, vojne i policijske primjene, prognoza vremena, itd. Na osnovu tabele (2.2) u kojoj su prikazani trening parovi, željeni i dobijeni izlazi može se zaključiti da mreža daje približno dobre rezultate za posmatrane trening parove tj. da na osnovu izračunatih težinskih faktora neuronska mreža dobro aproksimira zadatu funkciju, tj. kosi hitac. Nakon izvršenog treniranja mreže za zadate trening parove, potrebno je bilo utrenirati mrežu i za neke druge vrijednosti ulaza, a koje nisu iz skupa trenažnih vrijednosti. Posle istrenirane mreže za neke druge vrijednosti moglo se primijetiti da mreža pravi značajnu grešku u procjeni za te vrijednosti. Dolazi se do zaključka da bi mreža bolje radila ako bi se istrenirala za više trenažnih vrijednosti, jer bi to dovelo do toga da mreža ima više „znanja“ tj. mreža bi mogla sa većom tačnosti aproksimirati bilo koju vrijednost. Takođe, ono što bi još moglo poboljšati mrežu je promjena aktivacionih funkcija ili promjena broja neurona u skrivenom sloju.

# Literatura

[1] <https://sr.wikipedia.org/wiki/Ve%C5%A1ta%C4%8Dka_neuronska_mre%C5%BEa>

[2]<https://sr.wikipedia.org/wiki/%D0%9D%D0%B5%D1%83%D1%80%D0%BE%D0%BD%D1%81%D0%BA%D0%B0_%D0%BC%D1%80%D0%B5%D0%B6%D0%B0>

[3] [06 - NEURONSKE MREŽE](https://el.etfbl.net/mod/resource/view.php?id=25063), eLearning, Elektrotehnički fakultet Banja Luka

.