Primena dubokog Q učenja na automatsko igranje video igara

Matematički fakultet
Univerzitet u Beogradu

Student:
Nikola Milev

Mentor:
Mladen Nikolić

Beograd, 2018.

Sadržaj

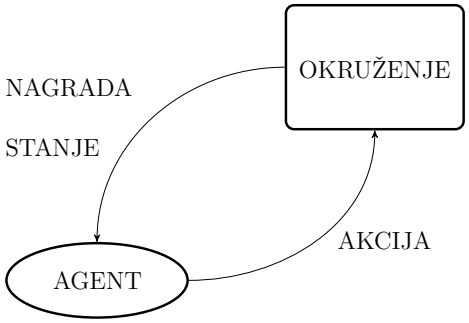
Glava 1

Uvod

U maju 1997. godine, Gari Kasparov, tadašnji svetski šampion u šahu, izgubio je meč protiv računarskog sistem pod nazivom Deep Blue. Skoro dvadeset godina kasnije, program pod nazivom AlphaGo pobedio je profesionalnog ljudskog igrača u igri go. Iako su obe igre strateške i igraju se na tabli, između šaha i igre go postoji ogromna razlika. Pravila igre go dosta su jednostavna u odnosu na šah ali je prostor koji opisuje poteze igre go više od 10^{100} puta veći od prostora koji opisuje poteze šaha. Programi koji igraju šah često se zasnivaju na korišćenju stabala pretrage i ovaj pristup jednostavno nije primenljiv na igru go.

Na čemu je onda zasnovan AlphaGo? U pitanju je učenje potkrepljivanjem (eng. reinforcement learning). Ovo je vrsta mašinskog učenja koja počiva na sistemu kazne i nagrade. Podrazumeva se da se sistem sastoji od agenta i okruženja u kom agent dela (vrši akcije) i dobija o numeričku nagradu i informaciju o promeni stanja okruženja. Osnovni dijagram ove komunikacije može se videti na slici 1.1. Poput dresiranja psa, nagradama se ohrabruje poželjno ponašanje dok se nepoželjno kažnjava. Cilj jeste ostvariti što veću dugoročnu nagradu. Međutim, agent mora sam kroz istraživanje da shvati kako da dostigne najveću nagradu tako što isprobava različite akcije. Takođe, preduzete akcije mogu da utiču i na nagradu koja se pojavljuje dugo nakon što je sama akcija preduzeta. Ovo zahteva da se uvede pojam dugoročne nagrade. Pojmovi istraživanja i dugoročne nagrade su ključni pri učenju potkrepljivanjem.

Pri učenju potkrepljivanjem, najčešće se pretpostavlja da je skup svih mogućih stanja



Slika 1.1: Dijagram komunikacije agenta sa okruženjem pri učenju potkrepljivanjem

okruženja diskretan. Ovo dozvoljava primenu Markovljevih procesa odlučivanja i omogućuje jednostavan formalan opis problema koji se rešava i pristupa njegovog rešavanja. Formalno predstavljanje problema i rešenja dato je u poglavlju 5.

Učenje potkrepljivanjem jedna je od tri vrste mašinskog učenja, pored nadgledanog i nenadgledanog učenja. Pri nadgledanom učenju, sistem dobija skup ulaznih i izlaznih podataka s ciljem da izvrši generalizaciju nad tim podacima i uspešno generiše izlazne podatke na osnovu do sada neviđenih ulaznih podataka. Pri učenju potkrepljivanjem, ne postoje unapred poznate akcije koje treba preduzeti već sistem na osnovu nagrade mora zaključi koji je optimalni sled akcija. Iako široko korišćeno, nadgledano učenje nije prikladno za učenje iz novih iskustava, kada izlazni podaci nisu dostupni. Kod nenadgledanog učenja, često je neophodno pronaći neku strukturu u podacima nad kojima se uči bez ikakvog pređašnjeg znanja o njima. Iako učenje potkrepljivanjem liči i na nadgledano i na nenadgledano učenje, agent ne traži strukturu niti mu je unapred data informacija o optimalnom ponašanju već teži maksimizaciji nagrade koju dobija od okruženja.

Učenje potkrepljivanjem ima mnogobrojne primene kao što su samostalna vožnja automobila i letelica, automatsko konfigurisanje algoritama, trgovina na berzi, igranje igara, itd. Ovaj vid mašinskog učenja pokazao se kao dobar i za igranje video igara. U radu objavljenom 2015. godine u časopisu Nature, DeepMind predstavlja sistem koji uči da igra video igre sa konzole Atari 2600, neke čak i daleko bolje od ljudi[1]. U avgustu 2017. godine, OpenAI predstavlja agenta koji isključivo kroz igranje igre i bez pređašnjeg znanja o igri stiče nivo umeća dovoljan da pobedi i neke od najboljih ljudskih takmičara u video igri Dota 2[2].

U naučnom radu koji je objavila kompanija DeepMind u časopisu Nature predložen je novi algoritam, DQN (eng. *deep Q - network*), koji koristi spoj učenja uslovljavanjem i duboke neuronske mreže i uspesno savladava razne igre za Atari 2600 konzolu. Sve informacije dostupne agentu jesu pikseli sa ekrana, trenutni rezultat u igri i signal za kraj igre. U sklopu ovog rada, ispitana je struktura algoritma DQN i data je implementacija čije su performanse testirane na manjoj skali od one date u radu, zbog ograničenih resursa. Takođe je eksperimentisano sa elementima samog algoritma i opisano je kako oni utiču na njegovo ponašanje.

[MOZDA NESTO O REZULTATIMA KADA IH BUDE]

U sklopu rada opisani su osnovni pojmovi mašinskog učenja (glava 2), zadržavajući se na neuronskim mrežama uopšte (glava 3) i na konvolutivnim neuronskim mrežama (glava 3.2). Glava 5 posvećena je učenju potkrepljivanjem dok je algoritam DQN u celosti opisan u glavi 6. U glavi 7 data je implementacija kao i njena evaluacija, dok su eksperimenti i njihovi rezultati opisani u glavi 8.

[1] UBACI NEKU REFERENCU

[2] https://blog.openai.com/dota-2/

Glava 2

Mašinsko učenje

Mašinsko učenje počelo je da stiče veliku popularnost devedesetih godina prošlog veka zahvaljujući potrebi i mogućnosti da se uči iz velike količine dostupnih podataka i uspešnosti ovog pristupa u tome. Za popularizaciju mašinskog učenja početkom 21. veka najzaslužnije su neuronske mreže, u toj meri da je pojam mašinskog učenja među laicima često poistovećen sa pojmom neuronskih mreža. Ovo, naravno, nije tačno; sem neuronskih mreža, postoje razne druge tehnike, kao što su metod potpornih vektora, linearni modeli, probabilistički grafovski modeli, itd.

Mašinsko učenje nastalo je iz čovekove želje da oponaša prirodne mehanizme učenja kod čoveka i životinja kao jedne od osnovnih svojstava inteligencije i korišćenja dobijenih rezultata u praktične svrhe. Termin mašinsko učenje prvi je upotrebio pionir veštačke inteligencije, Artur Semjuel[1], koji je doprineo razvoju veštačke inteligencije istražujući igru dame (eng. checkers) i tražeći način da stvori računarski program koji na osnovu iskusva može da savlada ovu igru[2].

Mašinsko učenje može se definisati kao disciplina koja se bavi izgradnjom prilagodljivih računarskih sistema koji su sposobni da poboljšaju svoje performanse koristeći informacije iz iskustva.[2] No, u biti mašinskog učenja leži generalizacija, tj. indukcija. Dve vrste zaključivanja, indukcija[3] i dedukcija[4] imaju svoje odgovarajuće discipline u sklopu veštačke inteligencije: mašinsko učenje i automatsko rezonovanje. Kao što se indukcija i dedukcija razlikuju, i mašinsko učenje i automatsko rezonovanje imaju različite oblasti primene. Automatsko rezonovanje zasnovano je na matematičkoj logici i koristi se kada je problem relativno lako formulisati ali ga, često zbog velikog prostora mogućih rešenja, nije jednostavno rešiti. U ovoj oblasti, neophodno je dobiti apsolutno tačna rešenja, ne dopuštajući nikakav nivo greške. S druge strane, mašinsko učenje pogodnije je kada problem nije moguće precizno formulisati i kada se očekuje neki novo greške. Čovek neke od ovih problema lako rešava a neke ne. Ukoliko je neophodno napraviti sistem koji prepoznaje životinje na slikama, kako definisati problem? Koji su tačno elementi oblika životinje? Kako ih prepoznati? Metodama automatskog rezonovanja bilo bi nemoguće definisati ovaj

[1] https://en.wikipedia.org/wiki/Machine_learning – da li da citiram Wiki ili njihov izvor?
[2] http://infolab.stanford.edu/pub/voy/museum/samuel.html – kako citirati izvor sa veba?
[3] Indukcija – zaključivanje od pojedinačnog ka opštem
[4] Dedukcija – zaključivanje od opšteg ka konkretnom

problem i rešiti ga. Mašinsko učenje, s druge strane, pokazalo se kao dobar pristup. Ono što je još karakteristično za mašinsko učenje jeste da rešenje ne mora biti savršeno tačno, iako se tome teži, i nivo prihvatljivog odstupanja zavisi od problema i konteksta primene.

Ova oblast je kroz manje od 20 godina od popularizacije postala deo svakodnevice. U sklopu društvene mreže Fejsbuk (eng. Facebook) implementiran je sistem za prepoznavanje lica koji preporučuje profile osoba koje se nalaze na slikama. Razni veb servisi koriste metode mašinskog učenja radi stvaranja sistema za preporuke (artikala u prodavnicama, video sadržaja na platformama za njihovo gledanje, itd). i sistema za detekciju prevara. Mnoge firme koje se bave trgovinom na berzi imaju sisteme koji automatski trguju deonicama. U medicini, jedna od primena mašinskog učenja jeste za uspostavljanje dijagnoze. Još neke primene su u marketingu, za procesiranje prirodnih jezika, bezbednost, itd.

2.1 Vrste mašinskog učenja

Kada se govori o određenoj vrsti mašinskog učenja, podrazumevaju se vrste problema, kao i načini za njihovo rešavanje. Prema problemima koji se rešavaju, mašinsko učenje deli se na tri vrste: nadgledano učenje (eng. supervised learning), nenadgledano učenje (eng. unsupervised learning) i učenje potkrepljivanjem (eng. reinforcement learning). Iako se podela mnogih autora sastoji samo iz nadgledanog i nenadgledanog učenja, postoji razlika između učenja potkrepljivanjem i preostale dve vrste. U nastavku su dati opisi pristupa nadgledanog i nenadgledanog učenja. Učenju uslovljavanjem, kao centralnoj temi ovog rada, posvećeno je više pažnje u poglavlju 5.

2.1.1 Nadgledano mašinsko učenje

Pri nadgledanom mašinskom učenju, date su vrednosti ulaza i izlaza koje im odgovaraju za određeni broj slučajeva. Sistem treba na osnovu već datih veza za pojedinačne parove da ustanovi kakva veza postoji između tih parova i izvrši generalizaciju, odnosno, ukoliko ulazne podatke označimo sa x a izlazne sa y, sistem treba da odredi funkciju f takvu da važi

$$y \approx f(x)$$

Pri uspešno rešenom problemu nadgledanog učenja, funkcija f davaće tačna ili približno tačna rešenja i za podatke koji do sada nisu viđeni. Ulazne vrednosti nazivaju se atributima (eng. features) a izlazne ciljnim promenljivima (eng. target variables). Ovim opisom nije određema dimenzionalnost ni za ulazne ni za izlazne promenljive, iako je dimenzija izlazne promenljive uglavnom 1. Funkcija f naziva se modelom.

Skup svih mogućih funkcija odgovarajuće dimenzionalnosti bio bi previše veliki za pretragu i zbog toga se uvode pretpostavke o samom modelu. Pretpostavlja da je definisan skup svih dopustivih modela i da je potrebno naći najpogodniji element tog skupa. Najčešće je taj skup određen parametrima, tj. uzima se da funkcija zavisi od nekog parametra w koji je u opštem slušaju višedimenzioni i tada se funkcija označava sa $f_w(x)$.

Neophodno je uvesti funkciju greške modela (eng. loss function), odnosno funkciju koja opisuje koliko dati model dobro određuje izlaz za dati ulaz. Ova funkcija se najčešće označava sa L i $L(y, f_w(x))$ predstavlja razliku između željene i dobijene vrednosti za pojedinačni par promenljivih. No, nijedan par vrednosti promenljivih nije dovoljan za opis kvaliteta modela već treba naći funkciju koja globalno ocenjuje odstupanje modela od stvarnih vrednosti. U praksi, podrazumeva se postojanje uzorka:

$$D = \{(x_i, y_i)|i = 1, ..., N\}$$

i uvodi sledeća funkcija:

$$E(w, D) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f_w(x_i))$$

koja se naziva prosečnom greškom. Uobičajeno, algoritmi nadgledanog mašinskog učenja zasnivaju se na minimizaciji prosečne greške.

Postoje dva osnovna tipa zadataka nadgledanog mašinskog učenja:

- Klasifikacija
- Regresija

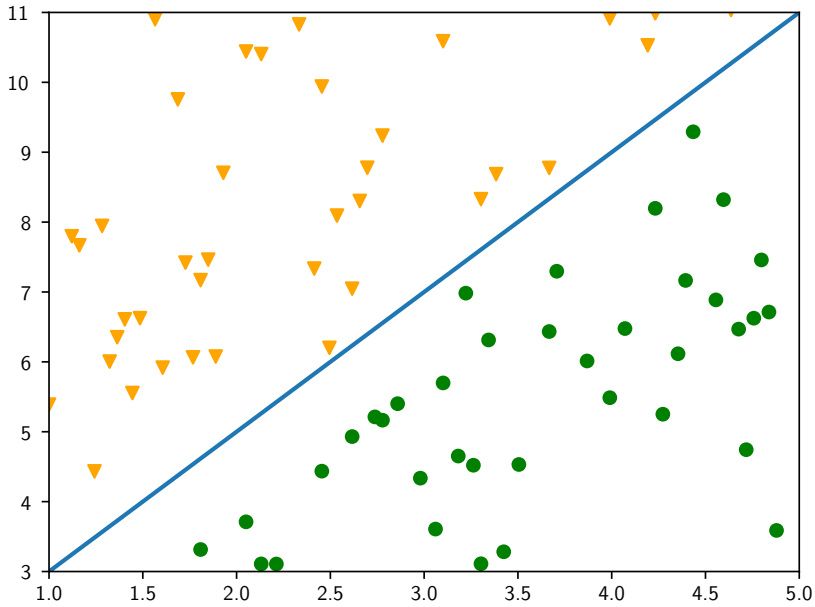
Klasifikacija (eng. *classification*) predstavlja zadatak mašinskog učenja gde je cilj predvideti klasu u kojoj se ciljna promenljiva nalazi. Neki od primera klasifikacije su svrstavanje slika na one koje sadrže ili ne sadrže lice, označavanje nepoželjne (eng. *spam*) elektronske pošte i prepoznavanje objekata na slikama. Jednostavan primer klasifikacije može se videti na slici 2.1, gde su trouglovima označeni podaci iznad prave $y = 2x + 1$ a krugovima podaci ispod date prave. Dakle, ta prava je klasifikacioni model sa parametrima $(w_0, w_1) = (1, 2)$.

Regresija je zadatak predviđanja neprekidne ciljne promenljive. Na primer, cene nekretnina mogu se predvideti na osnovu površine, lokacije, populacije koja živi u komšiluku, itd. Često korišćena vrsta regresije jeste linearna regresija. U slučaju linearne regresije, podrazumeva se da je funkcija $f_w(x)$ linearna u odnosu na parametar w. Iako se ovo na prvi pogled čini kao prilično jako ograničenje, to nije nužno slučaj; kako za atribute ne postoji zahtev za linearnosti, oni pre pravljenja linearne kombinacije mogu biti proizvoljno transformisani. Primer linearne regresije jeste aproksimacija polinomom:

$$f_w(x) = w_0 + \sum_{i=1}^{N} w_i x^i$$

2.1.2 Nenadgledano mašinsko učenje

Nenadgledano učenje obuhvata skup problema (i njihovih rešenja) u kojima sistem prihvata ulazne podatke bez izlaznih. Ovo znači da sistem sam mora da zaključi kakve



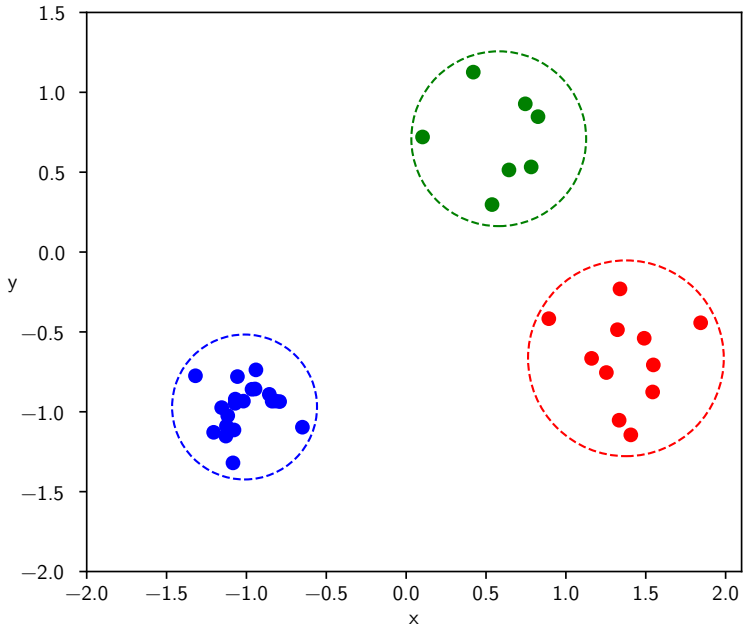
Slika 2.1: Binarna klasifikacija tačaka u skladu sa položajem u odnosu na pravu $2x + 1$

zakonitosti važe u podacima. Kako nije moguće odrediti preciznost sistema, cilj je naći najbolji model u odnosu na neki kriterijum koji je unapred zadat. Jedan primer nenadgledanog mašinskog učenja je klasterovanje: sistem grupiše neoznačene podatke u odnosu na neki kriterijum. Svaka grupa (klaster) sastoji se iz podataka koji su međusobno slični i različiti od elemenata preostalih grupa u odnosu na taj kriterijum. Cilj algoritma je određivanja tog kriterijuma grupisanja. Jednostavan primer klasterovanja po numeričkim atributima x i y može se videti na slici 2.2.

2.2 Dizajn sistema za mašinsko učenje

Okvirno, koraci u rešavanju problema su sledeći:[2]

- Prepoznavanje problema mašinskog učenja (nadgledano učenje, nenadgledano učenje, učenje potkrepljivanjem);
- Prikupljanje i obrada podataka, zajedno sa odabirom atributa;
- Odabir skupa dopustivih modela;
- Odabir algoritma učenja; moguće je odabrati postojeći algoritam ili razviti neki novi koji bolje odgovara problemu
- Izbor mere kvaliteta učenja;
- Obuka, evaluacija i, ukoliko je neophodno, ponavljanje nekog od prethodnih koraka radi unapređenja naučenog modela



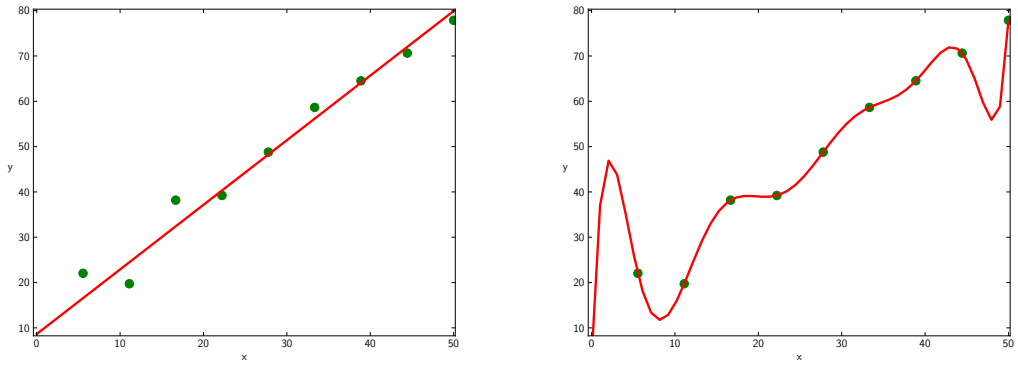
Slika 2.2: Klasterovanje

Prilikom odabira modela treba imati na umu vrstu problema koja se rešava, količinu podataka, zakonitosti koje važe u podacima, itd.

2.2.1 Podaci

Mašinsko učenje bavi se generalizacijom nad nepoznatim objektima na osnovu već viđenih objekata. Pod pojmom objekta misli se na pojedinačni podatak koji sistem vidi. Koriste se još i izrazi primerak i instanca. Vrednosti podataka pripadaju nekom unapred zadatom skupu. Podaci mogu biti različitog tipa: numerički ili kategorički. Skupovi koji određuju vrednosti kojima se instance određuju nisu unapred zadati i neophodno ih je odrediti na način pogodan za rešavanje konkretnog problema. Na primer, ukoliko je neophodno razvrstati slike životinja i biljaka na te dve kategorije, informacija o količini zelene boje na slici može biti prilično korisna, dok pri razvrstavanju vrste biljaka u zavisnosti od lista ovaj podatak skoro nije upotrebljiv (ali podatak o nijansi zelene boje može biti). Dakle, dobar izbor atributa imaće veliki uticaj na kasnije korake učenja. Podaci se sistemu daju kao vektori atributa. Nije uvek neophodno vršiti ekstrakciju atributa. Na primer, neuronske mreže u stanju su da uče nad sirovim podacima.

Podaci se neretko pre slanja sistemu obrađuju na neki način; ovaj postupak zove se pretprocesiranje. Postoje mnogi razlozi za pretprocesiranje a glavni cilj jeste da se dobiju objekti nad kojima učenje može da se vrši, imajući u vidu zahteve algoritama učenja. Međutim, i to zavisi od problema. Nekada će nepotpuni objekti, podaci koji ne sadrže sve informacije neophodne za učenje, biti izbačeni iz skupa podataka koji se razmatra, a u nekom drugom slučaju, i oni će biti korišćeni. Primeri pretprocesiranja su pretvaranje slike koja je u boji u crno beli zapis, normalizacija, umetanje nedostajućih vrednosti, itd.



Slika 2.3: Primer odabira modela pri linearnoj regresiji polinomom

2.2.2 Evaluacija modela

Nakon obučavanja (treniranja), neophodno je izvršiti evaluaciju dobijenog modela. Na koji god način se ovo izvršava, podaci korišćeni za obučavanje ne smeju se koristiti za evaluaciju modela. Često se pribegava podeli podataka na skupove za obučavanje i za testiranje. Skup za obučavanje obično iznosi dve trećine skupa ukupnih podataka. No, kako različite podele skupa mogu izazvati dobijanje različitih modela, slučajno deljenje nije najbolji izbor. Često korišćena tehnika jeste unakrsna validacija. Ovaj pristup podrazumeva podelu skupa podataka D na K podskupova približno jednake veličine, S_i za $i = 1, ..., K$. Tada se za svako i model trenira na skupu $D \setminus S_i$ a evaluacija se vrši pomoću podataka iz S_i. Posle izvedenog postupka za sve i, kao konačna ocena uzima se prosečna ocena svakog od K treniranja i evaluacija modela. Za vrednosti K uobičajeno se uzimaju vrednosti 5 ili 10. Ovaj metod vodi pouzdanijoj oceni kvaliteta modela.

2.3 Problemi pri mašinskom učenju

Kao što je podrazumevano pri pomenu pojma generalizacije, nije dovoljno odrediti funkciju koja dobro određuje izlazne vrednosti na osnovu promenljivih nad kojima se uči već je poželjno i novim ulaznim podacima dodeliti tačnu izlaznu vrednost. Odavde se može videti da je primer lošeg sistema za mašinsko učenje onaj sistem koji će izuzetno dobro naučiti da preslikava ulazne vrednosti iz skupa za učenje u odgovarajuće izlazne vrednosti ali u situaciji kada se iz tog skupa izađe neće davati zadovoljavajuće rezultate. Ovaj problem ima svoje ime: preprilagođavanje. Postoji i problem potprilagođavanja, koji podrazumeva da se sistem nije dovoljno prilagodio podacima. I preprilagođavanje i potprilagođavanje predstavljaju veliki problem ukoliko do njih dođe. Primer preprilagođavanja može se videti na slici 2.3, koja prikazuje razliku između dva modela iz skupa dopustivih modela za linearnu regresiju polinomom nad 10 različitih tačaka. Na levom delu slike prikazan je polinom reda 1 (prava) a na desnom delu prikazan je polinom reda 10. Iako će polinom reda 10 savršeno opisivati 10 tačaka sa slike, prava će verovatno bolje generalizovati nad novim podacima.

Na još jedan od mogućih problema nailazi se u slučaju neprikladnih podataka. Nekada

ulazni atributi ne daju dovoljno informacija o izlaznim. Takođe, moguće je da podataka jednostavno nema dovoljno. U ovom slučaju, sistem ne dobija dovoljno bogat skup informacija kako bi uspešno izvršio generalizaciju. S druge strane, moguće je da postoji prevelika količina podataka. Tada se pribegava pažljivom odabiru podataka koji se koriste za učenje ali ovo u opštem slučaju treba izbegavati jer su podaci izuzetno vredan element procesa mašinskog učenja. Još jedan problem vezan za podatke može biti njihova nepotpunost. Na primer, moguće je da u nekim instancama postoje nedostajuće vrednosti atributa.

Kako je najčešče potrebno pretprocesirati podatke u sklopu procesa mašinskog učenja, moguće je da u ovom postupku dođe do greške. Primera radi, prilikom rada sa konvolutivnim neuronskim mrežama, o kojima će biti reči u jednom od narednih glava, nekada se slike u boji pretvaraju u crno-bele. Ako se primeni transformacija koja onemogućuje razlikovanje objekata koji su različiti u početnoj slici a razlikovanje je neophodno za ispravno učenje, tada proces treniranja neće teći kako je planirano.

Problem može da nastane i ukoliko nije odabran pravi algoritam za učenje, ukoliko se loše pristupilo procesu optimizacije, prilikom lošeg procesa evaluacije i, naravno, prilikom loše implementacije algoritma. Sve ove prepreke često je moguće prevazići ali je jasno da je neophodno biti izuzetno pažljiv prilikom celog procesa mašinskog učenja.

Glava 3

Neuronske mreže

Neuronske mreže (eng. *neural networks*) predstavljaju danas izuzetno popularan vid mašinskog učenja. Ovi modeli izuzetno su fleksibilni i imaju široku primenu. Koriste se za prepoznavanje govora, prevođenje, prepoznavanje oblika na slikama, upravljanje vozilima, uspostavljanje dijagnoza u medicini, igranje igara itd. [Neuronskim mrežama može se aproksimirati proizvoljna neprekidna funkcija.] Pun naziv je veštačka neuronska mreža (eng. *artificial neural network*, skr. *ANN*) jer se ovakvi modeli idejno zasnivaju na načinu na koji mozak funkcioniše. Osnovne gradivne jedinice, neuroni, zasnovani su na neuronima u mozgu, dok veze između njih predstavljaju sinapse[1]. Te veze opisuju odnose između neurona i obično im se dodeljuje numerička težina.

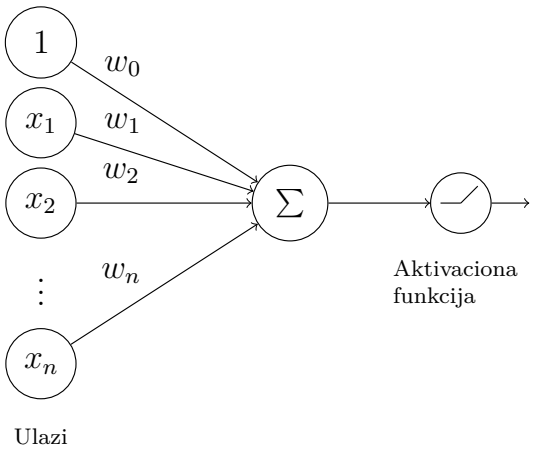
Postoji nekoliko različitih vrsta neuronskih mreža. Tipičan primer jesu neuronske mreže sa propagacijom unapred. Ime proističe iz činjenice da podaci teku od ulaza mreže do izlaza, bez postojanja ikakve povratne sprege. Neuronske mreže sa propagacijom unapred sastoje se iz slojeva neurona. Ukoliko se u ovaj model uvede neki tip povratne sprege, tada se govori o rekurentnim neuronskim mrežama. Pri radu sa slikama i raznim drugim vrstama signala, najčešće se koriste konvolutivne neuronske mreže, o kojima će biti reči kasnije. Ono što je zajedničko je da su neuronske mreže sposobne za izdvajanje određenih karakteristika u podacima koji se obrađuju. To znači da se vrši kreiranje novih atributa na osnovu već postojećih. Taj proces naziva se ekstrakcijom atributa i smatra se da je to jedan od najbitnijih razloga za delotvornost neuronskih mreža.

Za uspeh neuronskih mreža zaslužna je njihova fleksibilnost ali su rezultati dobijeni najpre eksperimentisanjem. Naime, veliki deo zaključaka o ponašanju neuronskih mreža u raznim situacijama nije teorijski potkrepljen. Stoga, istraživački rad vezan za neuronske mreže zahteva dosta pokušaja da bi se došlo do uspeha.

3.1 Neuronske mreže sa propagacijom unapred

Neuronske mreže sa propagacijom unapred jedna su od najkorišćenijih vrsta neuronskih mreža. Gradivni elementi ovakvog modela, neuroni (koji se još nazivaju i jedinicama),

[1]Sinapsa je struktura koja omogućuje komunikaciju između neurona.



Slika 3.1: Neuron

organizuju se u slojeve koji se nadovezuju i time čine neuronsku mrežu. Organizacija neurona i slojeva, uključujući i veze između neurona, predstavlja arhitekturu mreže. Prvi sloj mreže naziva se ulaznim slojem dok se poslednji sloj naziva izlaznim slojem. Neuroni prvog sloja kao argumente primaju ulaze mreže dok neuroni svakog od preostalih slojeva kao svoje ulaze prihvataju izlaze prethodnog sloja. Svi slojevi koji svoje izlaze prosleđuju narednom sloju nazivaju se skrivenim slojevima. Mreže sa više od jednog skrivenog sloja nazivaju se dubokim neuronskim mrežama. Broj slojeva mreže određuje njenu dubinu. Termin duboko učenje nastao je baš iz ove terminologije.

Svaki neuron opisuje se pomoću vektora $w = (w_0, ..., w_n)$ koji se naziva vektorom težina. Ulazni parametrar $x = (x_1, ..., x_n)$ linearno se transformiše na sledeći način:

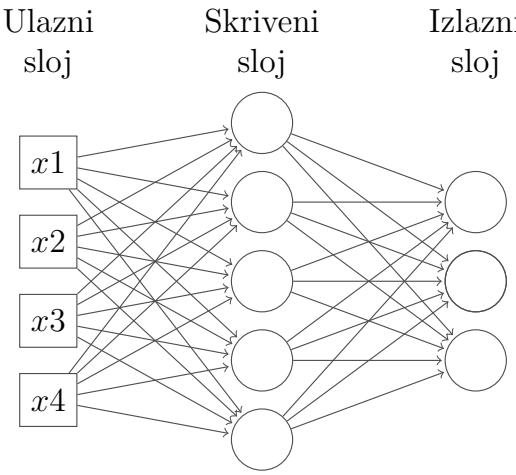
$$f_w(x) = w_0 + \sum_{i=1}^{n} x_n w_n \tag{3.1}$$

a zatim se primenjuje takozvana aktivaciona funkcija, g. Izlaz iz neurona je $g(f_w(x))$ i, uprkos linearnosti prve transformacije, izlaz ne mora biti linearna transformacija ulaza, tj. g najčešće nije linearna funkcija. Za g_i bira se nelinearna funkcija jer se u suprotnom kao celokupna transformacija koju neuron vrši dobija linearna funkcija; na ovaj način, mreža bi predstavljala linearnu funkciju i ne bi bilo moguće njom aproksimirati nelinearne funkcije dovoljno dobro. Vrednost w_0 naziva se slobodnim članom. Nekada se vektor x transformiše tako da bude oblika $x = (1, x_1, ..., x_n)$ kako bi izraz (3.1) imao kraći zapis $f_w(x) = w \cdot x$, gde $\cdot$ označava skalarni proizvod.

Model, tj. neuronska mreža, formalno se definiše na sledeći način:

$$h_0 = x$$
$$h_i = g_i(W_i h_{i-1} + w_{i0}), \text{ za } i = 1, ..., L$$

gde je x vektor ulaza u mrežu predstavljen kao kolona, W_i je matrica čija j-ta vrsta predstavlja vektor težina j-tog neurona u sloju i a w_{i0} je kolona slobodnih članova svih jedinica u sloju i. Funkcije g_i su nelinearna aktivaciona funkcija i za vektor $t = (t_1, ..., t_n)$, $g_i(t)$ predstavlja kolonu $(g_i(t_1), ..., g_i(t_n))^T$. Na ovaj način dobija se funkcija čiji su parametri W_i i w_{i0} za $i = 1, ..., L$. Ako se parametri označe sa w, tada se model zapisati



Slika 3.2: Neuronska mreža koja sadrži jedan skriveni sloj

kao f_w. Parametri w mogu se pronaći matematičkom optimizacijom nekog kriterijuma kvaliteta modela. Taj proces opisan je u delu 3.1.2.

3.1.1 Aktivacione funkcije

Preteča neuronskih mreža, perceptron, je model koji se sastoji samo iz jednog neurona čija je aktivaciona funkcija data sledećim izrazom:

$$g(x) = \begin{cases} 1, & \text{ako } x \geq 0 \\ 0, & \text{inače} \end{cases}$$

Definicija aktivacione funkcije perceptrona znači da njegova primena ima relativno jako ograničenje. S obzirom na to da će ulaz u funkciju g biti linearna kombinacija ulaza i parametara, perceptron će moći da napravi podelu prostora određenu hiperravni[2]. Ukoliko skup ulaznih podataka nije moguće podeliti linearnom funkcijom, ovakvo ponašanje nije zadovoljavajuće.

Dakle, neophodno je naći druge funkcije koje služe kao aktivacione funkcije. Poželjna svojstva aktivacione funkcije su:

- Nelinearnost: Kao što je objašnjeno ranije, kompozicija linearnih funkcija daje linearnu funkciju, što onemogućuje dovoljno preciznu aproksimaciju nelinearnih funkcija;
- Diferencijabilnost: Optimizacija se najčešće vrši metodima koji koriste gradijent funkcije;
- Monotonost: Ako aktivaciona funkcija nije monotona, povećavanje nekog od težinskih parametara neurona, umesto da poveća izlaz i time proizvede jači signal, može imati suprotan efekat;

[2]Hiperravan je uopštenje ravni u trodimenzionom prostoru; predstavlja potprostor dimenzije za 1 manje od prostora u kom se nalazi.

- Ograničenost: Ukoliko vrednosti unutar neuronske mreže nisu ograničene, moguće je da dođe do pojavljivanja ogromnih vrednosti koje potencijalno dovode do prekoračenja. Ograničene aktivacione funkcije ovo znatno ublažuju.

Dozvoljeno je da aktivaciona funkcija ne poseduje neko od navedenih svojstava ali ovime se može znatno smanjiti brzina konvergencije.

Najčešće korišćene aktivacione funkcije su:

- Sigmoidna funkcija: $\sigma(x) = \dfrac{1}{1+e^{-x}}$
- Tangens hiperbolički: $tanh(x) = \dfrac{e^{2x}-1}{e^{2x}+1}$
- Ispravljena linearna jedinica: $ReLU(x) = max(0, x)$

Sigmoidna funkcija bila je najkorišćenija aktivaciona funkcija pri radu sa neuronskim mrežama. Ograničena je (sve slike nalaze se u intervalu $(-1, 1)$), monotona i diferencijabilna u svakoj tački skupa $\mathbb{R}$. Međutim, što se argument više udaljava od nule, to nagib funkcije postaje manji. To znači da će gradijent funkcije biti mali i da će učenje teći jako sporo.
Tangens hiperbolički srodan je sigmoidnoj funkciji ($tanh(x) = 2\sigma(x) - 1$) ali je imala veći uspeh od sigmoidne funkcije. U okolini nule, ova funkcija slična je identičkoj, što olakšava učenje. Međutim, i pri korišćenju ove funkcije može se naići na problem sa malim gradijentima ukoliko se argument dovoljno udalji od nule.
Uprkos tome što za razliku od prethodne dve funkcije nije ni ograničena ni diferencijabilna u svim tačkama domena, danas je ispravljena linearna jedinica najpopularniji izbor za aktivacionu funkciju. Funkcija je jednaka identitetu desno od nule i stoga se gradijent ne menja. Takođe, verovatnoća da se traži gradijent u tački u kojoj funkcija nije diferencijabilna je mala. Ipak, ni ova funkcija nije bez mana; problem često pravi deo levo od nule, gde je funkcija konstantna. To znači da je gradijent nula i da se prilikom optimizacije težine neurona neće izmeniti. Zbog nedostatka promene, može se desiti da neki neuroni u mreži postanu pasivni, tj. da im izlaz postane 0. Za ovaj problem postoje rešenja; jedno jeste da izlaz funkcije desno od nule ne bude konstanta 0 već αx, za neko malo α. Ta modifikovana ReLU funkcija naziva se nakošena ispravljena linearna jedinica (eng. *leaky rectified linear unit*).
Iako sve ove funkcije imaju prednosti i mane u odnosu na preostale, ne postoji jedinstveni izbor nego je na osnovu problema neophodno zaključiti koju je aktivacionu funkciju najbolje koristiti.

Izlazni sloj

Neuronske mreže koriste se pri regresiji, određivanju funkcije koja opisuje vezu izmedju ulaza i izlaza, i klasifikaciji, svrstavanje ulaznih vektora u jednu od konačnog broja kategorija. Pri regresiji, u poslednjem sloju ne primenjuje se aktivaciona funkcija. Proces

optimizacije svodi se na minimizaciju funkcije greške. Kod rešavanja problema klasifikacije (u N kategorija), koristi se funkcija mekog maksimuma (eng. *softmax*):

$$softmax(x) = \left(\frac{e^{x_1}}{\sum_{i=1}^{N} e^{x_i}}, \ \dots \ , \frac{e^{x_N}}{\sum_{i=1}^{N} e^{x_i}} \right)$$

Suma ovako dobijenog vektora je 1 i stoga može predstavljati diskretnu raspodelu verovatnoća. Za vrednost aproksimacije uzima se kategorija kojoj odgovara najviša vrednost izlaznog vektora. Za optimizaciju pri radu sa probabilističkim problemima, kao što je problem klasifikacije, primenjuje se metod maksimalne verodostojnosti (eng. *maximum likelihood estimate*), odnosno traži se maksimum sledećeg izraza:

$$P(y_1, ..., y_N | x_1, ..., x_N)$$

.

3.1.2 Optimizacija

Ukoliko je neuronska mreža predstavljena kao funkcija f_w, gde su w parametri mreže, neophodno je izvršiti minimizaciju[3] funkcije koja predstavlja kriterijum kvaliteta aproksimacije. Problem optimizacije u slučaju neuronskih mreža težak je zbog nekonveksnosti. Ona čini neke metode teško primenljivim ili izuzetno sporim. Moguće je i završiti u lokalnom optimumu funkcije. Uobičajeno se koriste metodi zasnovani na gradijentu funkcije. Postoje metodi drugog reda, zasnovani na hesijanu [4] ali je njegovo računanje u slučaju većeg broja parametara preskupo.

Učenje funkcioniše na sledeći način za fiksirane ulaze x posmatra se njima uparen izlaz y i $f_w(x)$ a zatim i $L(y, f_w(x))$, odnosno funkcija greške između stvarne i očekivane vrednosti. Koristeći algoritam propagacije unazad (3.1.2) uz neki od algoritama za optimizaciju, vrši se minimzacija funkcije L u odnosu na parametre mreže, w.

Metod gradijentnog spusta i stohastičkog gradijentnog spusta

Gradijent funkcije $f : \mathbb{R}^n \to \mathbb{R}$ u tački $x = (x_1, ..., x_n)$ označava se sa ∇f i predstavlja vektor parcijalnih izvoda u toj tački:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), ..., \frac{\partial f}{\partial x_n}(x) \right)$$

.

Gradijent funkcije u tački x predstavlja pravac i smer najbržeg rasta funkcije pa $-\nabla f(x)$ predstavlja pravac smer najbržeg opadanja funkcije. Kako se najčešće minimizuje funkcija greške, u oznaci L, na dalje je korišćeno to ime za funkciju umesto f.

[3] Naravno, optimizacioni metodi primenjuju se i na maksimizaciju ali je u slučaju mašinskog učenja najčešće neophodno minizovati funkciju greške.

[4] Hesijan je matrica parcijalnih izvoda drugog reda.

Metod gradijentnog spusta jedan je od najstarijih metoda optimizacije. Iterativnim pristupom minimizuje se konveksna diferencijabilna funkcija. Polazeći od nasumično odabrane tačke i prateći pravac i smer gradijenta u svakom koraku, dolazi se do minimuma funkcije. Iterativni korak definisan je na sledeći način:

$$w_{k+1} = w_k - \alpha_k \nabla L(w_k), k = 0, 1, 2, ..., \tag{3.2}$$

gde je w_0 ta nasumično odabrana početna tačka a α_k je pozitivan realan broj koji se naziva veličinom koraka ili stopom učenja (eng. *learning rate*). Za funkciju greške u ovom slučaju uzima se srednjekvadratno odstupanje:

$$\frac{1}{2N} \sum_{i=1}^{N} \|y_i - f_w(x_i)\|_2^2$$

Bitno je pažljivo odabrati veličinu koraka jer ova vrednost može uticati na konvergenciju. Jedan primer odabira veličine koraka jeste niz za koji važe Robins Monroovi uslovi[5] :

$$\sum_{k=0}^{\infty} \alpha_k = \infty \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty$$

Jednostavniji pristup bio bi da se odabere mali pozitivan parametar α i da za svako k važi $\alpha_k = \alpha$.

Postavlja se i pitanje koliko koraka načiniti pre zaustavljanja. U praksi se koristi nekoliko kriterijuma kao što su zaustavljanje kada su dve uzastopne vrednosti w_k i w_{k+1} dovoljno bliske, kada su vrednosti funkcije za dve uzastopne vrednosti dovoljno bliske ali se može zaustaviti i nakon unapred određenog broja koraka. Postoji još kriterijuma i moguće ih je kombinovati.

Iako jednostavan i široko primenljiv metod optimizacije, gradijentni spust nije najbolji izbor. Naime, pravac najbržeg uspona funkcije nije uvek i pravac koji osigurava najbrže približavanje optimumu funkcije. U praksi, gradijentni spust ume da proizvodi cik-cak kretanje koje dovodi do spore konvergencije. Takođe, za jedan iterativni korak neophodno je proći kroz sve parove ulaza i izlaza, što u slučaju velikog skupa podataka za obučavanje može biti jako velika količina podataka.

Za obučavanje neuronskih mreža češće se koristi metod stohastičkog gradijentnog spusta. Pretpostavka je da je funkcija koja se optimizuje oblika:

$$L(w) = \frac{1}{N} \sum_{i=1}^{N} L_i(w)$$

odnosno da se može predstaviti kao prosek nekih N funkcija. Kako je neuronska mreža jedan od metoda mašinskog učenja, na raspolaganju je skup za obučavanje pa se funkcija greške na celom skupu može predstaviti kao prosek grešaka na pojedinačnim instancama skupa. Novi oblik jednakosti (3.2) je:

$$w_{k+1} = w_k - \alpha_k \left(\frac{1}{N} \sum_{i=1}^{N} \nabla L_i(w_k) \right), k = 0, 1, 2, ...$$

[5] https://en.wikipedia.org/wiki/Stochastic_approximation

. Pri korišćenju stohastičkog gradijentnog spusta za minimizaciju funkcije greške, iterativni korak izgleda ovako:

$$w_{k+1} = w_k - \alpha \nabla L_i(w_k)$$

Postoje razni načini za odabir i u nekom koraku, kao što je $i = k(modN) + 1$, gde je N veličina skupa za obučavanje. Još jedan primer je nasumični odabir instance u svakom koraku. Kakav god način izbora bio, neophodno je iskoristiti sve greške. Moguće je proći greške iz skupa za obučavanje i nekoliko puta dok se ne postigne željeni nivo aproksimacije.

Kako ova aproksimacija može biti prilično neprecizna, pribegava se kompromisu: prilikom iterativnog koraka ne koriste se pojedinačne instance već prosek nekog podskupa skupa za obučavanje (eng. *minibatch*). Pri treniranju neuronskih mreža, ovo je uobičajeni pristup.

Ovom aproksimacijom mogu se izbeći lokalni minimumi funkcije. Metod stohastičkog gradijentnog spusta manje je računski zahtevna od gradijentnog spusta ali je manje precizna i neophodan je veći broj iteracija kako bi se dostigao minimum.

Postoje razni metodi optimizacije koji se koriste pri mašinskom učenju. Neki menjaju veličinu koraka u zavisnosti od prethodnih izračunatih koraka i gradijenata. Takvi metodi nazivaju se adaptivnim metodima optimizacije. Primer adaptivnih metoda optimizacije su Adam i RMSProp.

RMSProp

Algoritam RMSProp (eng. *root mean square propagation*) predložio je Džof Hinton na jednom od svojih predavanja na sajtu Kursera [6]. Ovo je algoritam optimizacije korišćen prilikom razvijanja DQN algoritma. Glavna ideja je čuvanje dosadašnjeg otežanog proseka kvadrata gradijenta funkcije koji će biti obeležen sa g_k. Simbol $\odot$ obeležava pokoordinatno množenje dva vektora. Kako algoritam nije objavljen u radu, može se naći veliki broj implementacija. U nastavku je predstavljen algoritam u skladu sa implementacijom iz biblioteke Keras, koja je korišćena za implementaciju DQN algoritma u ovom radu.

$$g_0 = 0$$
$$g_{k+1} = \gamma g_k + (1 - \gamma) \nabla L(w_k) \odot \nabla L(w_k)$$
$$\alpha_0 = \alpha$$
$$\alpha_{k+1} = \frac{\alpha_k}{1 + d(k+1)}$$

Tada se iterativni korak definiše:

$$w_{k+1} = w_k - \frac{\alpha_{k+1}}{\sqrt{g_{k+1}} + \varepsilon} \nabla L(w_k)$$

Sve operacije vrše se pokoordinatno. Parametar γ pripada poluotvorenom intervalu $[0, 1)$. U svom predavanju, Hinton predlaže da njegova vrednost bude 0.9. Preporučena vrednost za veličinu koraka odnosno stopu učenja, u oznaci α, je 0.001 dok d označava faktor opadanja za parametar α. Parametar ε služi da bi se izbeglo deljenje nulom i obično je reda veličine 10^{-8}.

[6] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Adam

Adam (eng. *adaptive moment estimation*) jedan je od najčešćih algoritama za optimizaciju korišćen pri obučavanju neuronskih mreža. Algoritam Adam zasnovan je na korišćenju ocena prvog i drugog momenta gradijenta, datim sledećim formulama:

$$m_0 = 0$$
$$v_0 = 0$$
$$m_{k+1} = \beta_1 m_k + (1 - \beta_1)\nabla L(w_k)$$
$$v_{k+1} = \beta_2 v_k + (1 - \beta_2)\nabla L(w_k) \odot \nabla L(w_k)$$

Ocena prvog momenta, m_0, predstavlja otežani prosek pravca kretanja dok ocena drugog momenta, v_0, predstavlja otežani prosek kvadrata norme gradijenata. Međutim, ove dve ocene su pristrasne ka početnoj vrednosti, u ovom slučaju 0[7]. Da bi se to ispravilo, vrši se sledeća korekcija:

$$\hat{m}_{k+1} = \frac{m_{k+1}}{1 - \beta_1^{k+1}}$$
$$\hat{v}_{k+1} = \frac{v_{k+1}}{1 - \beta_2^{k+1}}$$

Na kraju, iterativni korak dat je ispod. Dodavanje skalara ε na vektor $\hat{v_{k+1}}$ predstavlja dodavanje tog skalara svakom članu datog vektora. Korenovanje, deljenje i oduzimanje vrše se pokoordinatno.

$$w_{k+1} = w_k - \alpha \frac{\hat{m}_{k+1}}{\sqrt{\hat{v}_{k+1}} + \varepsilon}$$

Parametar α naziva se veličina koraka ili stopa učenja. Vrednosti parametara β_1 i β_2 ograničene su na skup $[0, 1)$ i preporučene vrednosti su 0.9 i 0.999, redom, dok se za ε preporučuje vrednost 10^{-8}. Kao i u algoritmu RMSProp, svrha parametra ε je izbegavanje deljenja sa nulom. Takođe nalik algoritmu RMSProp opisanom iznad, moguće je uvesti stopu opadanja parametra α.

Intuicija kojom se vodi algoritam Adam jeste da dužina svakog koraka zavisi od osobina funkcije u regionu u kom se trenutno vrši optimizacija. Ovaj algoritam pokazao se kao superioran u odnosu na ostale algoritme za optimizaciju, u opštem slučaju.

Metod propagacije unazad

Metod propagacije unazad jedan je od najznačajnijih algoritama pri radu sa neuronskim mrežama. Zasniva se na korišćenju parcijalnog izvoda složene funkcije kako bi se izračunao gradijent funkcije koju predstavlja neuronska mreža. Na primer, ukoliko su date funkcije $g : \mathbb{R}^n \to \mathbb{R}^m$ i $f : \mathbb{R}^m \to \mathbb{R}$, tada se parcijalni izvod funkcije $f \circ g$ odnosno

[7] Ovde se misli na 0 vektor istih dimenzija kao x_k u slučaju prvog momenta i skalar 0 u slučaju drugog momenta

$f(g)$ po i-toj promenljivoj za $i = 1, ..., n$ računa:

$$\partial_i(f \circ g) = \sum_{j=1}^{m} (\partial_j f \circ g)\partial_i g_j$$

Svaka iteracija algoritma propagacije unazad sastoji se od tri koraka:

- Proširivanja do sada izračunatog parcijalnog izvoda izvodom aktivacione funkcije za dati sloj po pravilima
  računanja parcijalnog izvoda složene funkcije;
- Računanja vrednosti gradijenta prema parametrima jedinica datog sloja. Kako se pre aktivacione funkcije računa linearna kombinacija, ovaj gradijent je vektor vrednosti koji taj sloj dobija i ovaj korak se vrši množenjem tim vektorom;
- Proširivanja do sada izračunatog parcijalnog izvoda izvodom te linearne kombinacije po ulazima prateći pravilo za računanje parcijalnog izvoda složene funkcije.

Jednostavan primer rada algoritma propagacije unazad za složenu funkciju $f \circ g \circ h$ izgleda:

$$\begin{aligned} f(g(h(x)))' = & \\ &= f'(g(h(x)))g(h(x))' \\ &= f'(g(h(x)))g'(h(x))h(x)' \\ &= f'(g(h(x)))g'(h(x))h'(x) \end{aligned}$$

Sada su prikazani potpuni alati za optimizaciju neuronske mreže sa propagacijom unapred. Treba imati u vidu da je algoritam propagacije unazad računski skup i da pri radu sa velikim neuronskim mrežama proces učenja može biti jako skup. Takođe, sem težina same neuronske mreže, na proces učenja utiču razni drugi parametri kao što su arhitektura mreže, podela podataka na skupove za obučavanje i testiranje i parametri algoritama za optimizaciju. Ovi parametri nazivaju se metaparametrima i neretko je neophodno pokušavati razne njihove kombinacije dok ponašanje mreže ne dostigne željeni nivo. Često se umesto traženja metaparametara pribegava korišćenju unapred ispitanih vrednosti za koje je već pokazano da daju željene rezultate pri rešavanju nekog problema.

3.1.3 Prednosti i mane

Neuronske mreže pokazale su se kao jako korisne za rešavanje praktičnih problema zbog svoje izuzetne fleksibilnosti. Međutim, za proces obučavanja neuronske mreže neophodno je imati veliku količinu podataka. Proces učenja može biti izuzetno spor, posebno ukoliko se uvede isprobavanje raznih vrednosti metaparametara. Velika fleksibilnost može izazvati i preprilagođavanje podacima i time učiniti performanse mreže nad novim podacima lošim. Postoje i problemi pri optimizaciji kao što su takozvani problemi nestajućih i eksplodirajućih gradijenata. Iako su u stanju da konstruišu nove atribute na osnovu starih, struktura obučene mreže nije čitljiva za ljude. U nekim situacijama, ovo može izazvati probleme. Na primer, ukoliko klijent podnese zahtev za kredit i neuronska mreža

odluči da nije podoban, nije moguće objasniti razlog odbijanja. Neuronske mreže takođe su dosta računski i razvojno zahtevne. Nekada će neki već poznat algoritam dati zadovoljavajuće rešenje dok razvoj neuronske mreže može biti skup i po pitanju vremena razvijanja sistema i po pitanju kasnijeg rada sistema. Kako ne postoje teorijske smernice za rad sa neuronskim mrežama, odluke vezane za razvoj sistema neophodno je donositi empirijski.

3.2 Konvolutivne neuronske mreže

Konvolutivne neuronske mreže (eng. *convolutional neural networks*; skr. *CNN*), nekad nazivane samo konvolutivne mreže, često se koriste pri obradi signala kao što su slike. Razlog za njihovu uspešnost u ovom poslu je sposobnost konstruisanja novih atributa ali iz sirovog signala. Za konstruisanje novih atributa nije neophodna ljudska intervencija već mreža sama treba da ustanovi koja svojstva signala su bitna učenjem takozvanih filtera. Kako je iz jednostavnijih atributa neophodno konstruisati složenije, konvolutivne mreže skoro uvek su duboke mreže.

Opšta arhitektura konvolutivne neuronske mreže podrazumeva smenjivanje dve vrste slojeva, konvolutivnih, na koje se primenjuje nelinearna aktivaciona funkcija, i agregacionih (eng. *pooling*), koji će biti opisani u nastavku. Nekada se ista vrsta sloja ponavlja više puta. Na izlaze poslednjih od ovakvih slojeva obično se nadovezuje mreža sa propagacijom unapred zarad učenja nad atributima koje su prethodni slojevi konstruisali.

3.2.1 Konvolucija i konvolutivni slojevi

Osnovna operacija koju koriste konvolutivne mreže jeste konvolucija, po kojoj je ovaj tip modela i dobio ime. Konvolucija dve funkcije, f i g obeležava se simbolom $*$ i u neprekidnom slučaju izgleda:

$$(f * g)(x) = \int f(t)g(x-t)dt$$

Neprekidni slučaj nije uvek moguće primeniti i najčešće se koristi dvodimenziona diskretna konvolucija. Neka su f i g matrice dimenzija $m \times n$ i $p \times q$. Konvolucija matrica f i g data je izrazom:

$$(f * g)_{i,j} = \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} f_{i-k,j-l} g_{k,l} \tag{3.3}$$

Ovaj izraz računa se za sve (ispravne) kombinacije indeksa i i j i dobijena matrica je konvolucija matrica f i g. U ovom radu, ovaj oblik konvolucije je bitan i biće jedini razmatran. Matrica f naziva se ulazom a g filterom ili kernelom jer se pomoću nje izdvajaju neke informacije iz ulaza. Oduzimanje indeksa u sumi može biti zamenjeno sabiranjem; u ovom kontekstu nema bitne razlike.

Treba primetiti da izraz (3.3) nije definisan za sve i i j. Bitno je da su svi indeksi u svojim granicama: $i - k$ i $j - l$ treba da budu nenegativni i manji od, redom, m i n.

Dimenzija novodobijene matrice biće $m - k + 1 \times n - l + 1$ odnosno rezultat primene konvolucije manje je dimenzije nego početna matrica f. Kako to nekada nije poželjno, uvodi se proširivanje (eng. *padding*) matrice f takvo da rezultat konvolucije bude istih dimenzija kao početna matrica f. Jedan od načina da se ovo postigne je da se proširuje nulama ili da se proširi vrednostima koje su na obodu matrice f. Uz to, neretko se prekače izračunavanje za neke i i j, u zavisnosti od određenog pomeraja (eng. *stride*).

Konvolutivni sloj sastoji se takođe od jedinica, kao i sloj u neuronskoj mreži sa propagacijom unapred, ali je bitna razlika to što sve jedinice jednog konvolutivnog sloja dele parametre, odnosno filter. Ovo znači da se primena jednog sloja u konvolutivnoj mreži može posmatrati kao prevlačenje filtera koji je predstavljen tim parametrima preko celog signala. Ova pojava naziva se deljenjem parametara. U jednodimenzionom slučaju, jedinice sloja su organizovane u niz a u dvodimenzionom slučaju u matricu. U dvodimenzionom slučaju, primena jedinice na poziciji (i, j) može se shvatiti kao konkretizacija izraza (3.3) gde je f izlaz prethodnog sloja (eventualno proširen) a g matrica parametara datog sloja. Umesto primene jednog filtera po sloju, obično postoji više filtera koji se paralelno primenjuju na izlaz prethodnog sloja. Ovakvi konvolutivni slojevi nazivaju se višekanalnim slojevima. Isto tako, nekada se filteri primenjuju na više kanala izlaza prethodnog sloja u isto vreme.

Kao što je pomenuto, u konvolutivnoj mreži pojavljuju se i slojevi agregacije. Oni predstavljaju primenu neke funkcije agregacije nad nekim delom prethodnog sloja. I ovde je moguće uvesti pomeraj. Jedan primer je agregacija maksimumom gde se za svaku okolinu određene veličine računa maksimalni element i daje kao izlaz na odgovarajućoj poziciji. Pri agregaciji dolazi do gubitka informacije o tome gde se tačno nalazi neko svojstvo, što je često prihvatljivo ponašanje. Na primer, pri detekciji lica, u slučaju da je zaključeno da slika sadrži nos i dva oka, bez obzira na njihovu poziciju, jako je mala verovatnoća da ne sadrži i lice. S druge strane, nekada je pozicija uočenih karakteristika izuzetno bitna i tada se agregacija izbegava. Agregacija smanjuje dimenzije narednih slojeva pa se smanjuje broj parametara u njima i time računska zahtevnost opada. Pri primeni agregacionih, kao i konvolutivnih slojeva, može se uvesti pomeraj. Agregacija se vrši po kanalima, odnosno jednom kanalu ulaza u agregacioni sloj odgovara jedan kanal izlaza iz agregacionog sloja.

3.2.2 Prednosti i mane

U neuronskim mrežama sa propagacijom unapred, veze između dva sloja neurona uglavnom su guste, odnosno jedan neuron prihvata izlaze svih ili velikog broja neurona iz prethodnog sloja. Prilikom korišćenja konvolutivnih neuronskih mreža, jedna primena filtera ne koristi sve informacije iz prethodnog sloja, tj. jedna jedinica prihvata rezultat konvolucije sa filterom na jednoj poziciji. Ta pojava naziva se proređenim interakcijama. Na ovaj način se dosta smanjuje broj operacija za izračunavanje izlaza mreže i postiže se veća memorijska efikasnost. Kako se filter nezavisno primenjuje na sve ispravne pozicije izlaza prethodnog kanala, ove operacije je moguće paralelizovati što, koristeći moderan hardver podoban za to, umnogome ubrzava proces učenja.

Još jedna dobra osobina neuronskih mreža jesu deljeni parametri. Težine jednog neu-

rona u neuronskoj mreži sa propagacijom unapred koriste se tačno jednom pri izračunavanju izlaza mreže. S druge strane, jedan filter primenjuje se na sve pozicije ulaza[8] tako da je iskorišćenost njegovih parametara dosta veća nego u slučaju klasičnog neurona.

Još jedno bitno svojstvo konvolutivnih neuronskih mreža je to što su neosetljive na translaciju. Tačnije, translacija ulaza pa primena konvolucije ima isti efekat kao primena konvolucije koju prati translacija. Ovo znači da će neko svojstvo biti uočeno bez obzira na to kako je translirano u signalu.

I neuronske mreže sa propagacijom unapred bile bi u stanju da uče nad podacima kao što su slike ili zvuk ali bi raspored podataka u signalu bio raspoređen proizvoljno. Neuronske mreže sa propagacijom unapred ne uzimaju u obzir susednost podataka ali je ta susednost baš ono čemu konvolutivne mreže pridaju značaj, odnosno na ovaj način se izdvajaju karakteristike iz signala.

Nakon obučavanja konvolutivne mreže, moguće je izdvojiti naučene filtere i predstaviti ih. Na ovaj način moguće je shvatiti koje su to karakteristike signala izdvojene kao bitne u procesu učenja, što je jako pogodno svojstvo.

Međutim, konvolutivne mreže imaju i svoje mane. Pored problema koje imaju i mreže sa propagacijom unapred i koje nisu uklonjene osobinama konvolutivnih mreža, postoje i problemi specifični za konvolutivne mreže. Na primer, mreža je osetljiva na neke transformacije koje nisu translacija poput skaliranja ili rotacije. Još jedan problem javlja se pri obučavanju konvolutivnih mreža. Naime, ovaj proces zahteva dosta izračunavanja i bez specijalizovanog hardvera, može biti jako spor.

[8]U zavisnosti od pomeraja i proširivanja, postoje slučajevi kada se neki elementi u nekom koraku zanemaruju ali se filter uvek prevlači preko ulaza.

Glava 4

Markovljevi procesi odlučivanja

Markovljevi procesi odlučivanja (eng. *Markov decision process*, skr. *MDP*) daju teorijski okvir u kome je moguće relativno jednostavno postaviti i rešiti problem učenja potkrepljivanjem. Pretpostavlja se da postoji agent koji komunicira sa okruženjem. Agent u svakom trenutku ima informaciju o stanju okruženja i u mogućnosti je da preduzme akciju na šta od okruženja dobija odgovor u vidu novog stanja i numeričke nagrade. Još jedna važna pretpostavka jeste da agent ne dobija informaciju o tome da li je preduzeta akcija bila dobra ili ne. Okruženje se predstavlja pomoću Markovljevog procesa odlučivanja. U nastavku je dat teorijski opis Markovljevih procesa odlučivanja.

Pod pretpostavkom da se interakcija između agenta i okruženja izvršava u diskretnim trenucima, stanje okruženja u trenutku t označava se sa S_t dok se skup svih stanja označava sa $\mathcal{S}$. Agent u stanju S_t preduzima akciju $A_t \in \mathcal{A}(S_t)$ i prelazi u novo stanje, S_{t+1} dobijajući nagradu R_{t+1}. Skup nagrada, $\mathcal{R}$, je skup mogućih realnih nagrada. Sekvenca $S_0, A_0, R_1, S_1, A_1, R_2, S_2, ...$ naziva se putanjom i dobija se interakcijom agenta sa okruženjem. Putanja može biti i konačna i beskonačna. Neophodno je definisati i funkciju prelaska, p:

$$p(s', r \mid s, a) \stackrel{\text{def}}{=} P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

koja predstavlja verovatnoću prelaska u stanje s' i dobijanje nagrade r pod uslovom da je u stanju s preduzeta akcija a. Markovljevi procesi odlučivanja imaju takozvano Markovljevo svojstvo tj. osobinu da trenutno stanje i nagrada zavise isključivo od prethodnog stanja i u njemu preduzete akcije a ne od cele putanje koja je dovela do datog stanja. Formalno, ovo svojstvo zapisuje se sledećom jednakošću:

$$P(S_t, R_t \mid S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}) = P(S_t, R_t \mid S_{t-1}, A_{t-1})$$

U odnosu na putanju do trenutka t, može se govoriti o dugoročnoj nagradi, koja se još naziva i dobitkom:

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_t + i + 1$$

Metaparametar γ naziva se umanjenjem i ukazuje na to koliko se značaja pridaje kasnije dobijenim nagradama u odnosu na neposrednu nagradu. Za $\gamma = 0$, buduće nagrade nisu bitne dok postavljanje γ na 1 uzrokuje u smatranju svih nagrada putanje jednako bitnim.

Sada je moguće dati formalnu definiciju: Markovljev proces odlučivanja je uređena petorka $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$. Ova definicija je prilično jednostavna a ipak je dovoljno fleksibilna za formalne opise raznih modela. Ukoliko, na primer, treba modelirati okruženje takvo da u stanju s akcija a nije dozvoljena tada će $p(s', r \mid s, a)$ biti 0 za sve s' i r. Ako su skupovi $\mathcal{S}$, $\mathcal{A}$ i $\mathcal{R}$ konačni, tada se za Markovljev proces odlučivanja kaže da je konačan.

- vrednost stanja - q funkcija - politika - Belman?

- motivacija
- formalna postavka
- mozda neko objasnjenje
- mozda neki primer

Glava 5

Učenje potkrepljivanjem

Glava 6

DQN

Glava 7

Detalji implementacije

Glava 8

Eskperimentisanje sa elementima algoritma DQN

Literatura

[1] Mladen Nikolić i Anđelka Zečević. *Mašinsko učenje*. Matematički fakultet, Univerzitet u Beogradu, 2018. http://ml.matf.bg.ac.rs/readings/ml.pdf.

[2] Predrag Janičić i Mladen Nikolić. *Veštačka inteligencija*. Matematički fakultet, Univerzitet u Beogradu, 2018. http://poincare.matf.bg.ac.rs/~janicic//courses/vi.pdf.

[3] Yoshua Bengio i Aaron Courville Ian Goodfellow. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.