UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Milan M. Čugurović

AUTOMATSKO PRILAGOĐAVANJE KLASIFIKATORA RUKOM PISANOG TEKSTA POJEDINAČNIM KORISNICIMA

master rad

Beograd, 2019.

Mentor:

dr Mladen Nikolić, docent
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Predrag Janičić, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Jovana Kovačević, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: 23.09.2019.

Dugujem zahvalnost mentoru doc. dr Mladenu Nikoliću na saradnji tokom izrade projekta i pisanja rada, članovima komisije prof. dr Predragu Janićiću i doc. dr Jovani Kovačević na detaljnom čitanju i komentarima na tezu, dr Novaku Novakoviću iz kompanije Majkrosoft razvojni centar Srbija na saradnji tokom izrade projekta, kao i kolegi Ognjenu Kociću na korisnim savetima i sugestijama.

Naslov master rada: Automatsko prilagođavanje klasifikatora rukom pisanog teksta pojedinačnim korisnicima

Rezime: Iako je razvoj računarstva značajno uticao na način na koji pišemo, obrađujemo i čuvamo tekst, pisanje teksta rukom je i dalje verovatno dominantna praksa. Ipak, i u tom kontekstu je poželjno iskoristiti prednosti obrade i čuvanja podataka koje moderni računari pružaju. Jedan od prvih koraka ka tom cilju je prepoznavanje rukom napisanog teksta, odnosno njegovog prevođenja iz slike u niz karaktera koji se čuvaju u digitalnom obliku. Široka upotreba tablet računara takođe daje podstrek razvoju aplikacija koje obrađuju rukom pisani tekst i imaju potrebu za njegovim prepoznavanjem. Takvi sistemi su u novije vreme često zasnovani na metodama mašinskog učenja i kod njih je povremena greška očekivana pojava. Naravno, učestalost grešaka je poželjno smanjiti.

Smanjenje učestalosti grešaka predstavljeno u ovom radu koristi činjenicu da se specifičnosti pojedinačnih rukopisa praktično ne menjaju i implementira metod koji vrši prepoznavanje specifičnosti rukopisa korisnika na čijem računaru se izvršava. Ovakvo prilagođavanje je izvršeno praćenjem na kojim karakterima izvorni prepoznavač greši, inkrementalnim obučavanjem alternativnih modela koji se fokusiraju na te karaktere i ocenom statističkog modela koji bira između polaznog prepoznavača i alternativnih modela. Pri tome, alternativni modeli ne uključuju skupo obučavanje s obzirom da se ono izvršava na računaru korisnika. U radu je korišćen programski jezik Python sa svojim bibliotekama za mašinsko učenje, numerička izračunavanja i slično. Kao skupovi podataka za obučavanje i evaluaciju korišćeni su skup rukom pisanih karaktera NIST Special Database 19 Američkog nacionalnog instituta za standarde i tehnologiju kao i skup podataka Deepwriting objavljen od strane grupe istraživača sa ETH Univerziteta u Cirihu.

Ključne reči: neuronske mreže, rukom pisani tekst, klasifikacija stila, algoritam najbližih suseda, klasterovanje

Sadržaj

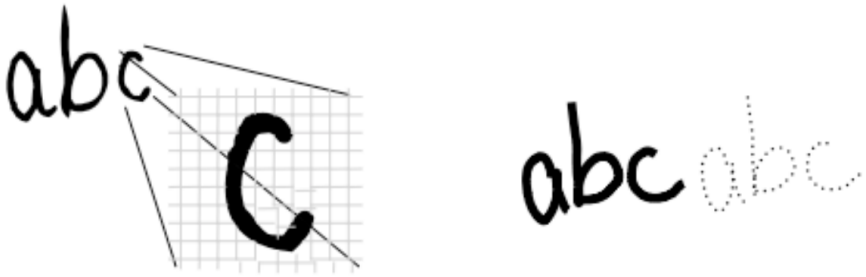
Glava 1

Uvod

Oblast optičkog prepoznavanja karaktera (eng. *optical character recognition, OCR*) predstavlja domen istraživanja više podoblasti računarstva kao što su veštačka inteligencija (eng. *artificial inteligence*), prepoznavanje obrazaca (eng. *pattern recognition*), računarska vizija (eng. *computer vision*) i druge. Optičko prepoznavanje karaktera odnosi se na opšte tehnike koje se koriste u oblastima obrade rukom pisanih karaktera ili digitalizacije slika, tako da one mogu biti elektronski skladištene, pretraživane i modifikovane. Automatsko prepoznavanje rukom pisanih karaktera može se smatrati podoblašću pomenute oblasti.

Problem automatskog prepoznavanja rukom pisanih karaktera predstavlja jedan nadasve praktičan problem. Inspirisan primenom u praksi razvija se kako u okviru akademske zajednice, tako i u okviru industrijskog sektora. U industriji se rešenja ovog problema direktno komercijalizuju uključivanjem prepoznavača rukom pisanih karaktera u uređaje poput tableta, pametnih telefona i slično. Zahvaljujući tome svakodnevni život prosečnog korisnika nekada postaje jednostavniji. Problem automatskog prepoznavanja rukom pisanih karaktera sreće se u dve varijante: oflajn i onlajn varijanti. Prva rukopis predstavlja kao sliku gledajući na njega kao na statički objekat, dok druga na rukopis gleda sa dinamičke strane, uzimajući u obzir kretanje ruke odnosno olovke prilikom njegovog nastanka. Ilustracija ovoga data je na slici 1.1.

Pionirski pokušaj automatskog prepoznavanja rukom pisanih karaktera datira još iz pedesetih godina prošlog veka [25]. Nakon ovog početnog pokušaja nekoliko grupa istraživača radilo je nezavisno na ovom problemu. U okviru softverskih i hardverskih ograničenja tadašnjeg računarstva ostvareni su zapaženi rezultati. U poslednjih desetak godina intenzivan razvoj neuronskih mreža doveo je do pome-



Slika 1.1: Rukom pisani karakteri, oflajn i onlajn varijanta, redom

ranja granica u mnogim oblastima, pa tako i u oblasti prepoznavanja rukom pisanih karaktera. Preciznost klasifikacije koju ostvaruju konvolutivne neuronske mreže nadmašuje rezultate svih metoda razvijenih do tada.

Kao što prethodni pasus ističe, konvolutivne neuronske mreže uvele su revoluciju i u ovu oblast značajno nadmašujući performanse dosadašnjih klasifikatora. I pored velikog poboljšanja, one imaju još prostora za napredak. Cilj ovog rada jeste upravo povećavanje preciznosti neuronskih mreža kreiranih u svrhu klasifikacije rukom pisanih karaktera. Poboljšanje se bazira na činjenici da je rukopis svakog pojedinca nešto jedinstveno, odnosno nešto što ga u velikoj meri identifikuje. Metod poboljšanja, kog ovaj rad prezentuje, teži povećanju preciznosti baznog klasifikatora učenjem specifičnosti rukopisa korisnika.

Novi pristup zasniva se na ideji poznavanja stila pisanja karaktera za svakog pojedinačnog korisnika. Poboljšanje pretreniranog klasifikatora rukom pisanih karaktera zasniva se na dinamičkom praćenju kako grešaka tako i uspešnih predviđanja tog klasifikatora. Ovim praćenjem kreira se takozvana istorija pisanja korisnika. Na osnovu nje se formira skup od nekoliko modela k najbližih suseda, za razne vrednosti k i na principijelan način (koji će biti opisan kasnije) se ocenjuje pouzdanost kako baznog klasifikatora, tako i ovih modela. Na osnovu tih ocena se odlučuje koju labelu predvideti.

Predloženi metod daje značajno poboljšanje preciznosti. Na skupu podataka korišćenom u ovom radu, preciznost baznog klasifikatora povećana je za više od dva procenta. Time su nadmašeni dosadašnji rezultati objavljeni na ovom skupu podataka. Preciznost (eng. *accuracy*) dostignuta u ovom radu je 89.60%.

Rad je podeljen u nekoliko poglavlja. U okviru poglavlja 2 opisani su relevantni pojmovi mašinskog učenja na koje se oslanja ovaj rad. Uvedene su osnovne definicije, diskutovane odgovarajuće podele, i detaljno opisane tehnike koje se koriste u

okviru ovog rada: metod k najbližih suseda, algoritam klasterovanja k sredina kao i konvolutivne neuronske mreže. Svaka od prethodnih tema predstavlja značajan deo prezentovanog metoda poboljšanja. Poglavlje 3 odnosi se na precizne matematičke osnove koje pružaju teorijsku potporu metodu. U okviru njega diskutuje se kako Bernulijeva raspodela slučajne promenljive, tako i Beta funkcija odnosno raspodela. Uvode se pojmovi apriorne i aposteriorne verovatnoće i izvodi dokaz teoreme o konjugovanosti Beta i Bernulijeve raspodele. Poglavlje 4 razmatra predloženo rešenje, kako sa teorijskog tako i sa praktičnog aspekta. Diskutuju se tehnički detalji implementacije, kao i metod poboljšanja. Poglavlje 5 navodi precizne rezultate evaluacije predloženog pristupa.

Glava 2

Osnovni pojmovi mašinskog učenja

Mašinsko učenje predstavlja trenutno vrlo popularnu granu veštačke inteligencije, odnosno računarstva uopšte. Iako je pomenuta oblast nastala još pedesetih godina prošlog veka kada je razvijen perceptron, prvi sistem koji je bio sposoban da uči jednostavne zavisnosti (Rozenblat 1957.), tek početkom ovog veka mašinsko učenje dolazi u žižu kako stručne, tako i laičke javnosti. Zahvaljujući razvoju tehologije grafičkih karata računari postaju sposobni da u kratkom vremenskom peridu odrade obimna izračunavanja, što doprinosi ponovnom oživljavanju neuronskih mreža, koje su nastale još šezdesetih godina prošlog veka. Industrija, kao i akademska zajednica poslednjih godina daju veliki doprinos kako praktičnoj primeni, tako i novim naučnim rezultatima oblasti.

U nastavku će biti izloženi neki od osnovnih pojmova mašinskog učenja. Oblastima koje se koriste u ovom radu biće posvećena posebna pažnja. Prezentacija ove teme pratiće izlaganje iz knjige „Mašinsko učenje" autora Mladena Nikolića i Anđelke Zečević [28].

2.1 Osnovni pojmovi i podele

U okviru mašinskog učenja podaci su izuzetno značajni. Otuda je neophodno definisati šta oni predstavljaju i koje vrste podataka postoje. Postoje dve osnovne vrste podataka koje se pominju u metodama mašinskog učenja. Za skup podataka kažemo da je **labelovan** ako ga možemo predstaviti u obliku $D = \{(x^{(n)}, y^{(n)}) \in A^d \times B\}_{n=1}^{N}$ gde je A^d prostor atributa a B skup labela, dok je sa N označen broj primeraka skupa D. Za skup podataka kažemo da **nije labelovan** ako se predstavlja u obliku $D = \{x^{(n)} \in A^d\}_{n=1}^{N}$. Svaki primerak $x^{(n)}$ predstavlja vektor u d-dimenzionom pro-

storu atributa. Svaki element ovog vektora nazivamo atributom. Element $y^{(n)}$, kada postoji odnosno u slučaju labelovanog skupa podataka, predstavlja element skupa labela odnosno oznaka. Njime se ostvaruje veza između vektora atributa i skupa labela.

Fokusirajući se na prirodu problema učenja, može se doći do osnovne podele metoda mašinskog učenja. Na osnovu toga, oblast se deli na nadgledano učenje (eng. *supervised learning*), nenadgledano učenje (eng. *unsupervised learning*) i učenje uslovljavanjem (eng. *reinforcement learning*) [28].

Nadgledano učenje se karakteriše činjenicom da je na raspolaganju labelovan skup podataka. Dakle skup podataka se sastoji od uređenih parova onoga na osnovu čega se uči i onoga šta je iz toga potrebno naučiti. Ovaj vid učenja pokušava da pronađe vezu između skupa atributa i odgovarajućeg skupa labela. Ako svakom vektoru atributa x odgovara jedna ciljna labela $y \in L$, $L = \{l_1, l_2, ..., l_c\}$, problem nadgledanog učenja naziva se problemom klasifikacije. Odgovarajuće vrednosti ciljne promenljive y nazivamo klasama. Sa druge strane, ako svakom vektoru atributa x odgovara vrednost ciljne promenljive $y \in B$, za neki neprekidan skup B, problem učenja naziva se problemom regresije. Modeli naučeni metodama nadgledanog učenja obično se koriste za predviđanja i prepoznavanja. Ovo je u praksi najznačajniji vid mašinskog učenja. Neki od primera problema nadgledanog učenja jesu prepoznavanje saobraćajnih znakova, detekcija kao i prepoznavanje lica na slikama, predviđanje cena akcija na berzi, detekcija kancera, vremenska prognoza.

Nenadgledano učenje se karakteriše činjenicom da skupu podataka na osnovu kojeg se uči nedostaju labele. Ovo odgovara nedostatku informacije o tome šta je potrebno naučiti. Metodi nenadgledanog učenja obično traže neku relevantnu strukturu u podacima. Neki od metoda nenadgledanog učenja jesu metode klasterovanja, ocene gustine raspodele, pronalaženje veza između skupa atributa, smanjenje dimenzionalnosti podataka. Metod nenadgledanog mašinskog učenja može uključiti i kombinaciju prethodno navedenih metoda. Primena metoda nenadgledanog učenja obično predstavlja vid preprocesiranja podataka za metode nadgledanog učenja. Ovo nije nužno, odnosno metod nenadgledanog učenja može imatu korisnu ulogu sam po sebi.

Učenje potkrepljivanjem predstavlja kompromis prethodna dva pristupa. Koristi se u situacijama kada je potrebno problem rešiti donošenjem niza odluka. Pretpostavlja se da je sistem predstavljen agentom koji opaža tekuće stanje okruženja, na osnovu koga preduzima odgovarajuće akcije, za koje dobija odgovarajuće na-

grade. Rezultat toga je učenje odgovarajuće optimalne politike ponašanja. Ključna pretpostavka je da u vreme učenja nije poznato koja akcija je optimalna u odgovarajućem kontekstu, s obzirom da bi se tada radilo o problemu nadgledanog učenja. Neki od problema učenja potkrepljivanjem su samovozeći automobili, roboti koji se sami kreću kao i automatsko igranje šaha. O ovom vidu mašinskog učenja neće biti više reči u daljem tekstu, čitalac za više informacija može pogledati odgovarajuću literaturu.

2.2 Model, funkcija greške i regularizacija

Na osnovu toga kakve zavisnosti modeluju, metodi mašinskog učenja dele se na probabilističke generativne modele (koji modeluju zajedničku raspodelu podataka kojom se opisuju zavisnosti kako između atributa i ciljne promenljive, tako i zavisnosti među samim atributima), probabilističke diskriminativne modele (koji modeluju uslovnu raspodelu ciljne promenljive u odnosu na vrednosti atributa) i neprobabilističke diskriminativne (koji ne modeluju raspodelu, već ocenjuju funkciju zavisnosti ciljne promenljive u odnosu na date vrednosti atributa) [28].

Ocena kojom se aproksimira bilo ciljna gustina raspodele bilo funkcija veze ciljne promenljive u odnosu na zadate vrednosti atributa naziva se *modelom*. Prema načinu reprezentacije, modele delimo na dve kategorije: parametarske i neparametarske.

Parametarska reprezentacija modela pretpostavlja da je model funkcija $f_w(x)$ koja modeluje zavisnost ciljne promenljive y u odnosu na date vrednosti atributa x a koju konfiguriše konačan skup parametara w. Izbor modela svodi se na izbor vrednosti prametara w.

Neparametarska reprezentacija modela ne pretpostavlja inicijalnu formu modela, već se forma modela određuje na osnovu datih podataka. Ovi modeli, uprkos svom nazivu, imaju parametre. Ti parametri su ipak u tesnoj vezi sa količinom podataka odnosno brojem instanci i samim instancama u skupu za obučavanje, stoga model ne može biti predstavljen konačnim skupom parametara. Otuda i naziv ove vrste modela mašinskog učenja.

Prednost neparametarskih modela u odnosu na parametarske predstavlja činjenica da inicijalno nije potrebno pretpostaviti određenu formu modela. Veća količina pretpostavki umanjuje fleksibilnost samog modela, što u krajnjem slučaju može rezultirati lošijim kvalitetom modela (ukoliko se pretpostavi forma modela koja nije odgovarajuća). U daljem tekstu biće razmatrani modeli sa parametarskom repre-

zentacijom.

Izborom vrednosti odgovarajućih parametara dobijaju se modeli različitog kvaliteta. Ocena kvaliteta modela vrši se funkcijom greške. Funkcija greške (eng. *loss function*) jeste funkcija koja meri odstupanja predviđenih i stvarnih vrednosti ciljne promenljive. Označimo je sa L, a sa $L(y_i, f_w(x_i))$ označavamo grešku parametarskog modela f_w na jednoj instanci skupa za obučavanje. Otuda ukupnu grešku modela, grešku na celom skupu podatataka koji su nam na raspolaganju, računamo kao $\sum_{i=1}^{N} L(y_i, f_w(x_i))$. Tada izbor parametara w, koji nazivamo i fazom treninga modela, predstavlja proces minimizacije ukupne greške:

$$\min_w \sum_{i=1}^{N} L(y_i, f_w(x_i))$$

Pored funkcije greške, postoje i drugi načini kojima se evaluira kvalitet modela. U slučaju problema klasifikacije jedna od važnijih mera kvaliteta modela jeste njegova preciznost (eng. *classification accuracy*). Preciznost klasifikatora definiše se kao količnik broja tačno klasifikovanih instanci i ukupnog broja instanci na kojima je klasifikator dao predviđanje.

Proces izbora parametara w, odnosno prethodni proces minimizacije, možemo shvatiti kao prilagođavanje modela podacima. Prevelika fleksibilnost modela, zajedno sa prethodnim procesom minimizacije, može voditi ka tome da se model previše prilagodi podacima u smislu da je ukupna greška modela na skupu podataka koji nam je dostupan za obučavanje mala, dok je greška predviđanja modela na novim podacima velika. Ova pojava naziva se preprilagođavanje modela podacima.

Smanjivanje fleksibilnosti modela ne predstavlja dobar način za prevazilaženje prethodno navedenog problema. Naime, modeli koji nisu dovoljno fleksibilni, ne samo da se ne mogu preprilagoditi podacima, oni im se često ne mogu čak ni u dovoljnoj meri prilagoditi, usled čega imaju visoku vrednost greške ne samo na skupu za testiranje, već i na skupu za obučavanje. Otuda se prethodni problem prevazilazi tehnikama *regularizacije*. Regularizacija predstavlja modifikaciju minimizacionog problema, koji postaje [28]:

$$\min_w \sum_{i=1}^{N} L(y_i, f_w(x_i)) + \lambda R(w)$$

gde $R(w)$ predstavlja regularizacioni izraz (funkcija ciljanih parametara w, često l_2 norma ili kvadrat iste), a λ regularizacioni metaparametar koji otežavaju prilagođavanje modela podacima. Regularizacija vodi modelima koji dobro generalizuju.

2.3 Algoritam k najbližih suseda

Algoritam k najbližih suseda predstavlja kako jedan od najjednostavnijih metoda mašinskog učenja, tako i jedan od najčešće korišćenih. Izuzetno se lako implementira, a neretko daje dobre rezultate. Može se koristiti kako za klasifikaciju, tako i za regresiju. Ovaj algoritam, u kontekstu gorenavedenih podela predstavlja neparametarski model mašinskog učenja, probabilistički diskriminativni. Jedini zahtev za primenu ovog modela jeste pretpostavka postojanja metrike odnosno funkcije rastojanja nad prostorom atrbuta skupa podataka.

Intuicija koja se krije iza ovog algoritma prilično je jednostavna. Razmotrimo jedan prilično naivan klasifikacioni model mašinskog učenja. Model se sastoji u tome da memoriše sve instance skupa za trening, a zatim instance skupa za testiranje klasifikuje samo ukoliko dođe do tačnog poklapanja sa skupom atributa neke, prethodno memorisane instance. Očigledna mana ovakvog algoritma jeste činjenica da je test instanca klasifikovana isključivo ukoliko ako se desilo tačno poklapanje sa određenom trening instancom. Dakle, metodu nedostaje moć generalizacije. Poboljšanje bi se moglo zasnivati na tome, da novoj instanci pridružujemo klasu instance memorisanog trening skupa koja joj je najbliža u smislu neke, prethodno definisane metrike. Time model predviđa vrednosti ciljne promenljive i za instance koje prethodno nije video. Ovo predstavlja konkretizaciju pojma najbližeg suseda [38].

Model f dobijen metodom najbližih suseda, biramo tako da važi [28]:

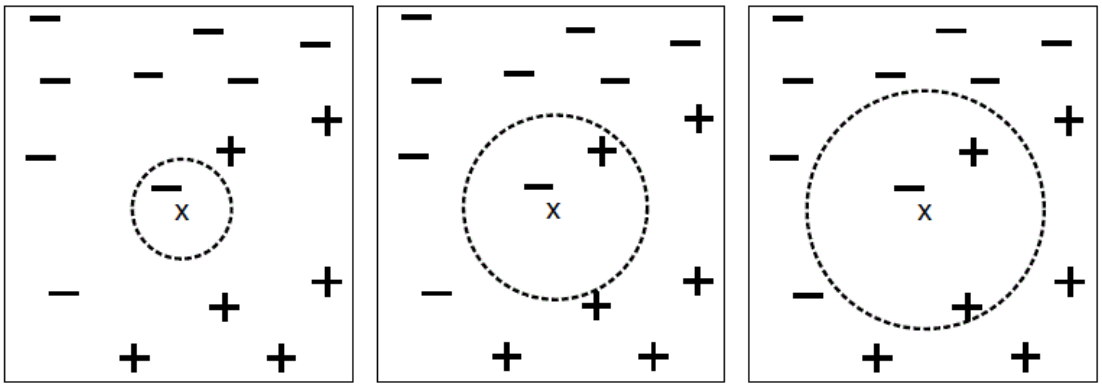
$$f(x) = \arg\max_{y} p(y|x)$$

gde se verovatnoća aproksimira udelima u skupu suseda. Laički rečeno, algoritam k najbližih suseda klasifikuje novu instancu tako što u memorisanom skupu za trening pronalazi njoj k najbližih instanci, u smislu predefinisane metrike, i pridružuje joj klasu koja je najčešća među pronađenim instancama. U slučaju regresije kao predviđanje može se uzeti bilo prosek, bilo težinski prosek nađenih instanci. Za težine se često uzimaju recipročne vrednosti kvadrata rastojanja ciljne instance do pomenutih.

Ovaj algoritam nema eksplicitnu formu modela, nema uobičajenu funkciju greške u odnosu na koju bi model bio obučavan, nedostaje i sama faza obučavanja, osim izbora vrednosti metaparametra k.

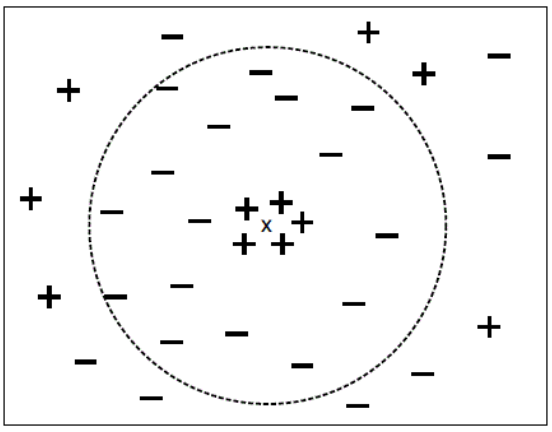
Slika 2.1 ilustruje algoritam najbližih suseda, kada se za broj suseda k fiksiraju redom vrednosti 1, 2 i 3. Metrika koja se koristi je euklidska[1], a ciljna instanca jeste

[1] l_2 metrika



Slika 2.1: Ponašanje algoritma za različite izbore vrednosti metaparametra k

ona u centru kružnice. Ako se koristi jedan najbliži sused, instanca se proglašava za negativnu. Ako je vrednost metaparametra k postavljena na tri, tada ciljana instanca u svojoj okolini ima dva pozitivna i jednog negativnog suseda, stoga se ona proglašava pozitivnom. U slučajevima kada se među najbližim susedima nalaze podjednako instance svih klasa, kao što to prikazuje prethodna slika kada je broj suseda jednak dva, za predviđanje ciljne instance može se slučajno birati neka od vrednosti, ili naprednije koristiti neka heuristika izbora. Ako je posmatrani broj suseda k relativno mali, dolazi do preprilagođavanja modela podacima, dok za velike vrednosti parametra k dolazi do potprilagođavanja, s obzirom da glasaju i susedi koji nisu prostorno blizu ciljnoj instanci. Ovo je ilustrovano na slici 2.2 [38].



Slika 2.2: Metod najbližih suseda za klasifikaciju u slučaju velikog broja k

Prednosti metoda k najbližih suseda jesu jednostavna implementacija kao i činje-

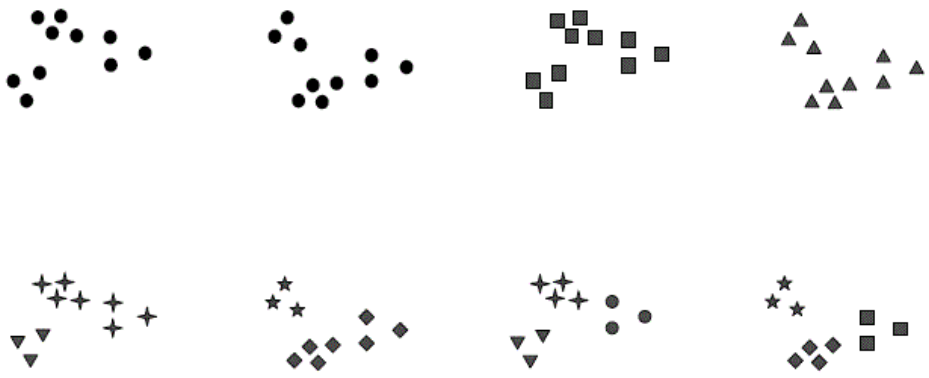
nica da nije neophodno pretpostavljati formu modela, dok su neke od mana potreba za čuvanjem svih instanci skupa za obučavanje kao i osetljivost u odnosu na izbor metaparametra k.

Važno je napomenuti da s obzirom na fundamentalnost ovog algoritma postoji veliki broj njegovih implementacija u raznim programskim jezicima i bibliotekama.

2.4 Algoritam klasterovanja K-sredina

U mnogim domenima i konkretnim problemima postoji potreba za grupisanjem podataka u odnosu na određene metrike. Oblasti poput biologije, veba (eng. *WorldWideWeb*), analize klimatskih promena, medicine i biznis aplikacija, po svojoj prirodi zahtevaju ovakve metode i alate [38].

Klasterovanje predstavlja identifikaciju grupa u dostupnim podacima, i kao takvo jeste metod nenadgledanog mašinskog učenja. Dakle odgovarajući skup podataka nije labelovan, a u njemu je potrebno pronaći određenu vrstu strukture, i u odnosu na istu podatke podeliti u grupe, koje se nazivaju klasterima. Prethodno uvedena neformalna definicija pojma klasterovanja nosi u sebi veliku dozu neodređenosti. Naime, prilikom klasterovanja date podatke grupišemo u klastere, pri čemu se postavlja pitanje šta to grupu instanci vezuje u smislu predstavljanja istog klastera, kao i šta ih to razlikuje od drugih instanci koje ne pripadaju tom klasteru. Dodatno, postavlja se i pitanje granularnosti odnosno grubosti klasterovanja - u koliko klastera podeliti podatke. Ovo je ilustrovano slikom 2.3.



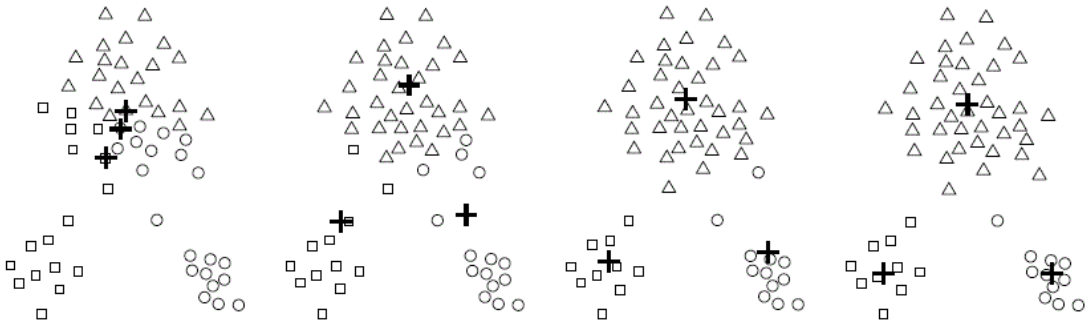
Slika 2.3: Klasterovanje datih tačaka (gore levo) u dva (gore desno), četiri (dole levo) odnosno šest (dole desno) klastera

Algoritam klasterovanja K-sredina predstavlja najpoznatiji algoritam klasterovanja. U podacima pronalazi strukturu grupišući instance u k klastera, pri čemu su klasteri reprezentovani pomoću težišta, koje predstavljaju prosek elemenata klastera. Odavde direktno sledi da je algoritam K-sredina primenljiv isključivo na podatke nad kojima je definisano uprosečavanje. Pod određenim, nešto slabijim uslovima postoje uopštenja ovog algoritma, ali ona neće biti razmatrana. Bitno je naglasiti da se, u osnovnoj varijanti ovom algoritmu mora fiksirati broj klastera k.

Kada se fiksira vrednost k, prvi korak algoritma predstavlja fiksiranje k težišta (obično se vrši nasumični izbor, mada se mogu koristiti i određene heuristike ili njihovo postavljanje od strane korisnika). Nakon toga, algoritam podrazumeva iteriranje sledećih koraka [28]:

- Preraspodeliti svaku instancu tako da pripada najbližem težištu, u odnosu na predefinisanu metriku
- Nova težišta računati po njihovoj definiciji (prosek instanci klastera)

Ovi koraci ponavljaju se bilo određeni, unapred definisani broj iteracija, bilo dok postoje promene težišta određene veličine, u odgovarajućem prostoru. Ilustracija izvršavanja data je na slici 2.4.



Slika 2.4: Promena težišta prilikom iteriranja algoritma

Razmotrimo malo detaljnije korake prethodnog algoritma. Da bismo dodelili instanci najbliže težište, potrebna nam je predefinisana metrika u odnosu na koju definišemo pojam rastojanja, odnosno pojam blizine. Ako je prostor atributa $\mathbb{R}^n$, obično se koristi l_2 metrika, dok se recimo kosinusna sličnost može koristiti ako instance pripadaju skupu dokumenata. Svakako postoji više mogućih izbora metrike koje mogu odgovarati istom prostoru atributa. Recimo Menhetn rastojanje

(l_1 metrika) može biti korišćeno u prostoru $\mathbb{R}^n$, dok *Žakardova* mera može biti pogodna za prostor dokumenata. Obično, metrika koju algoritam koristi da definiše rastojanje predstavlja jednostavnu funkciju čije računanje ne troši previše resursa. Naime, algoritam u svakom koraku računa rastojanje od svake instance do svakog težišta, u cilju pridruživanja najbliže. Napredne varijante ovog algoritma uključuju poboljšanja koja smanjuju broj potrebnih računanja.

Ukoliko algoritam koristi euklidsko rastojanje, jednostavnim računanjem parcijalnih izvoda može se pokazati da on minimizuje funkciju [38]:

$$\sum_{i=1}^{k} \sum_{x \in C_i} l_2(c_i, x)^2$$

gde C_i predstavlja i-ti klaster, a c_i i-to težište. Na osnovu toga, jasno se vide pojedine specifičnosti ovog algoritma. Naime, koristeći Euklidsko rastojanje, klasteri dobijaju oblik sfere. Dodatno, kako je rastojanje kvadrirano, odudarajući podaci (eng. *outlier*) imaju nešto veći uticaj na pozicije težišta, s obzirom na činjenicu da rastojanje raste sa kvadratom. U slučaju homogene gustine tačaka algoritam preferira klastere sa približnim brojem tačaka u njima.

Osnovni nedostatak ovog algoritma predstavlja činjenica da je istom neophodno fiksirati broj klastera k. Ovo se prevazilazi, kao što je već rečeno, određenim proširenjima samog algoritma ili pojedinim heuristikama.

2.5 Neuronske mreže

Neuronske mreže (eng. *Neural networks*) predstavljaju trenutno najaktuelniju oblast mašinskog učenja. Primenjuju se u mnogim oblastima. Mnogi problemi, koji do sada nisu bili savladani, bivaju rešeni. U novije vreme kako naučna zajednica tako i industrijski sektor sve su više uključeni u razvoj oblasti, kako kroz publikacije, tako i kroz praktične aplikacije. Važno je napomenuti da se u laičkim krugovima mašinsko učenje često poistovećuje sa neuronskim mrežama, što svakako predstavlja veliku grešku.

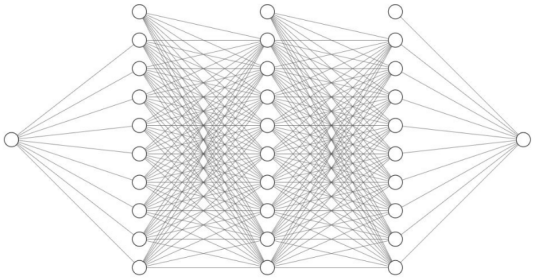
Osnovni pojmovi i podele

Neuronske mreže najčešće se sreću u nekoliko osnovnih varijacija. To su *potpuno povezane neurnoske mreže* (eng. *fully connected neural network*) koje predstavljaju osnovnu varijantu ovog metoda, *konvolutivne neuronske mreže* (eng. *convolutional*

neural network) koje se koriste u obradi slike, zvuka kao i drugih podataka u sirovom (izvornom) obliku, *rekurentne neuronske mreže* (eng. *recurrent neural network*) koje se koriste u obradi podataka nalik nizovima promenljive dužine, kod kojih se prirodno nameće rekurzivna struktura. Pored njih, postoje i druge arhitekture, o kojima na ovom mestu neće biti reči. Na ovom mestu posebna pažnja biće posvećena konvolutivnim neuronskim mrežama, s obzirom na činjenicu da se u konkretnoj implementaciji vrši poboljšanje klasifikatora predstavljenog upravo konvolutivnom neuronskom mrežom. Pre toga biće dat kratak osvrt na potpuno povezane neuronske mreže, obzirom da one predstavljaju sastavni deo konvolutivnih neuronskih mreža.

Potpuno povezana neuronska mreža

Potpuno povezana neuronska mreža predstavlja osnovnu arhitekturu ovog modela mašinskog učenja. Mreža se sastoji od neurona, odnosno osnovnih računskih jedinica u kojima se vrše jednostavne matematičke operacije. Neuroni su međusobno grupisani tako da formiraju slojeve, pri čemu neuroni svakog sloja kao svoje ulaze uzimaju izlaze neurona prethodnog sloja (i samo njih), dok svoje izlaze na raspolaganje stavljaju neuronima narednog sloja. Svaki od neurona računa jednostavnu nelinearnu transformaciju linearne kombinacije svojh ulaza, odnosno linearne kombinacije izlaza neurona prethodnog sloja. Ove nelinearne transformacije nazivaju se aktivacione funkcije. Prvi sloj naziva se ulaznim, ostali slojevi skrivenim. Neuronska mreža naziva se dubokom ako ima više od jednog skrivenog sloja. Ulazi prvog sloja nazivaju se ulazima mreže, dok se izlazima mreže nazivaju izlazi poslednjeg od njih. Naziv potpuno povezana neuronska mreža dolazi od toga što neuroni jednog sloja svoje izlaze šalju svim neuronima narednog sloja, bez izuzetka. Shema potpuno povezane neuronske mreže data je na slici 2.5.



Slika 2.5: Shema mreže sa propagacijom unapred

Formalna postavka modela

Model potpuno povezane neuronske mreže formalno se uvodi sledećim definicijama [28]:

$$h_0 = x$$
$$h_i = g(W_i h_{i-1} + w_{i0}),\ i = 1, 2, ..., L$$

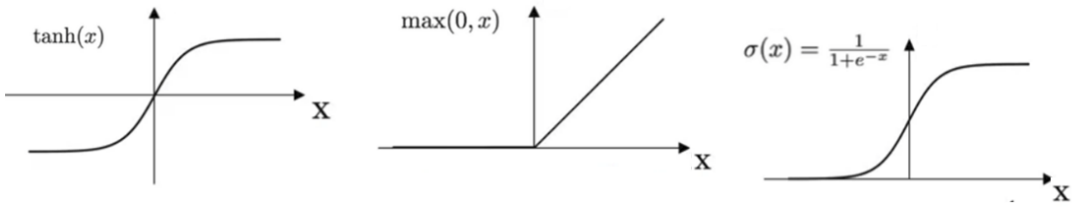
gde x predstavlja vektor ulaznih promenljivih, W_i matricu čija j-ta vrsta predstavlja vektor vrednosti parametara j-te jedinice u i-tom sloju, w_{i0} vektor slobodnih članova i-tog sloja, g nelinearnu aktivacionu funkciju, a L broj slojeva mreže. Kompletan model dat je jednačinom $f_w(x) = h_L$.

Jako važan teorijski rezultat koji predstavlja potporu modela neuronskih mreža jeste teorema o univerzalnoj aproksimaciji. Ova teorema kaže da je skup neuronskih mreža sa samo jednim skrivenim slojem svuda gust na skupu svih neprekidnih funkcija segmenta $[0, 1]^n$. Teorema tvrdi postojanje takve mreže, međutim ne daje nikakvo uputstvo kako do takve mreže u konkretnom problemu doći [28]. Ovo predstavlja jedan prilično ozbiljan problem, usled čega se treniranju neuronske mreže kao i procesu izbora arhitekture neuronske mreže posvećuje posebna pažnja.

Aktivacione funkcije

Aktivacione funkcije predstavljaju funkcije čiji izlazi predstavljaju izlaze samih neurona odgovarajuće mreže. Perceptron, prvi sistem po uzoru na koji su nastale moderne neuronske mreže, sastojao se od jednog neurona koji je za aktivacionu funkciju koristio indikatorsku funkciju: $g(x) = I(x \geq 0)$. Postoji nekoliko očiglednih mana ovakve aktivacione funkcije. Jedna očigledna jeste ta što je izvod pomenute funkcije gde postoji, jednak nuli. Postojanje izvoda koji nije konstantan ispostavlja se kao jako važno svojstvo prilikom korišćenja algoritama optimizacije neuronske mreže predstavljenje funkcijom $f_w(x)$.

Aktivacione funkcije idealno imaju svojstva nelinearnosti, diferencijabilnosti i monotonosti. Najčešće korišćene aktivacione funkcije su sigmoidna, tangens hiperbolički, ispravljena linearna jedinica (eng. $ReLU$) kao i nakošena ispravljena linearna jedinica (eng. *Leaky* $ReLU$). Grafici nekih od njih dati su na slici 2.6.



Slika 2.6: Grafici aktivacinih funkcija, redom tangens hiperbolički, ReLu, sigmoidna

Obučavanje neuronskih mreža

S obzirom da je neuronska mreža predstavljena funkcijom $f_w(x)$ koja je konfigurabilna u odnosu na parametre w, njih je potrebno izabrati tako da neuronska mreža predstavlja dobar regresor odnosno klasifikator. Potrebno je dakle izvršiti minimizaciju funkcije greške u odnosu na paramete.

Posmatrajući fiksirane ulaze x i njima odgovarajuće vrednosti labela y, ako funkciju greške označimo sa L, potrebno je minimizovati razliku između stvarne i očekivane vrednosti u smislu $L(f_w(x), y)$. Prethodni proces se naziva procesom optimizacije neuronske mreže. Isti je jako zahtevan, usled nekonveksnosti i velikog broja lokalnih minimuma koji su posledica kompleksnosti same mreže. Koristeći algoritam propagacije unazad u kombinaciji sa tehnikama optimizacije (u novije vreme najčešće Adam) vrši se minimizacija funkcije greške L u odnosu na vrednosti parametra w.

Algoritam propagacije unazad (eng. *backpropagation*) slobodno se može nazvati jednim od najvažnijih algoritama mašinskog učenja uopšte [16]. Zasniva se na pravilu računanja parcijalnih izvoda kompozicije funkcija. Za dve funkcije $f : R^n \to R$ i $g : R^m \to R^n$ i njihovu kompoziciju, važi:

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

Algoritam se kreće po slojevima mreže počev od poslednjeg ka prvom, i u svakom koraku proširuje do tog momenta izračunati parcijalni izvod.

Adam (eng. *Adaptive moment estimation*) predstavlja trenutno najkorišćeniji algoritam za obučavanje neuronskih mreža. Dat je formulama [22]:

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

gde m_{k+1} predstavlja našu ocenu očekivanja gradijenta koju nazivamo prvi moment gradijenta (ocena predstavlja težinski prosek) dok v_{k+1} predstavlja našu ocenu kvadrata norme gradijenta koju nazivamo drugi (necentrirani) moment[2] gradijenta. Algoritam iterativno vrši ažuriranje (sa eksponencijalnim uticajem parametara konvergencije β_1 i β_2) prvog i drugog momenta gradijenta. Prvi moment gradijenta (m_{k+1}) predstavlja akumulirani pravac kretanja, dok drugi moment gradijenta predstavlja brzinu kretanja tim pravcem [4].

Ove dve ocene su pristrasne ka početnoj vrednosti koja je u ovom sličaju nula (jer su početne vrednosti ovih ocena jednake nuli). Stoga se u algoritam uvodi sledeće poboljšanje, koje prethodne ocene čini nepristrasnim:

$$m_{k+1} = \frac{m_{k+1}}{1-\beta_1^{k+1}}$$
$$v_{k+1} = \frac{v_{k+1}}{1-\beta_2^{k+2}}$$

Iterativni korak, kojim se vrše ažuriranja odgovarajućh parametara, dat je sa:

$$w_{k+1} = w_k - \alpha \frac{m_{k+1}}{\sqrt{v_{k+1}}+\epsilon}$$

Parametar α predstavlja veličinu koraka učenja, dok parametri β_1 i β_2, kao što je rečeno, predstavljaju parametre konvergencije algoritma. Operacije deljenja i korenovanja u prethodnoj formuli vrše se pokoordinatno.

I pored značajnog razvoja gradijentnih tehnika optimizacije kojima se obučavaju neuronske mreže, ovo i dalje predstavlja jako težak posao. U cilju što boljeg i efikasnijeg obučavanja neuronskih mreža razvijene su i mnoge pomoćne tehnike. Tehnika ranog zaustavljanja (eng. *early stopping*) zaustavlja obučavanje neuronske mreže u slučaju da se greška koja se minimizuje nije smanjila neki, unapred definisani broj epoha. Njome se povećava efikasnost obučavanja neuronske mreže. Tehnika smanjenja koraka učenja (eng. *reduce learning rate on plateu*) omogućava bolje obučavanje neuronskih mreža tako što smanjuje korak učenja u slučaju da se greška koja se minimizuje nije smanjila neki, unapred definisani broj epoha. Ovo omogućava optimizatorima da siđu u uske minimume.

Prednosti i mane

Kvalitet neuronskih mreža pre svega ogleda se u širokom spektru kompleksnih problema koje su u stanju uspešno rešavati. Neuronske mreže često su u stanju

[2]naziv moment gradijenta potiče iz članka o iterativnim metodima optimizacije, objavljenog 1964. godine [33].

da učeći nelinearne, jako komplikovane atribute kreiraju nove, pogodnije reprezentacije ulaznih podataka, nad kojima su onda u stanju da uče. Njihove mane, pre svega predstavljaju velika količina podataka potrebna za njihovo obučavanje, kao i složenost procesa optimizacije, s obzirom na nekonveksnost funkcije greške. Neinterpretabilnost je takođe jedna od većih mana ovog metoda.

Konvolutivne neuronske mreže

Konvolutivne neuronske mreže predstavljaju neuronske mreže specijalizovane za rad nad sirovim podacima. Najčešće se radi o slikama, zvuku kao i drugim signalima. Zasnivaju se na operaciji *konvolucije* kao i na sposobnosti ovih modela da sami kreiraju atribute iz datih podataka.

Neka su f i g dve funkcije. Konvolucija funkcija f i g u oznaci $f * g$ definisana je na sledeći način [16]:

$$(f * g)(v) = \int_{-\infty}^{+\infty} f(x)g(v - x)dx$$

Prethodnim izrazom definisana je operacija konvolucije u jednodimenzionom neprekidnom slučaju. Definiciju je moguće proširiti kako na diskretan domen, tako i na funkcije više dimenzija. Jednodimenziona diskretna konvolucija, na uzorku dimenzije n data je sa [29]:

$$(f * g)_i = \sum_{j=0}^{n-1} f_j g_{n-j},\ i = 0, 1, ..., n - 1$$

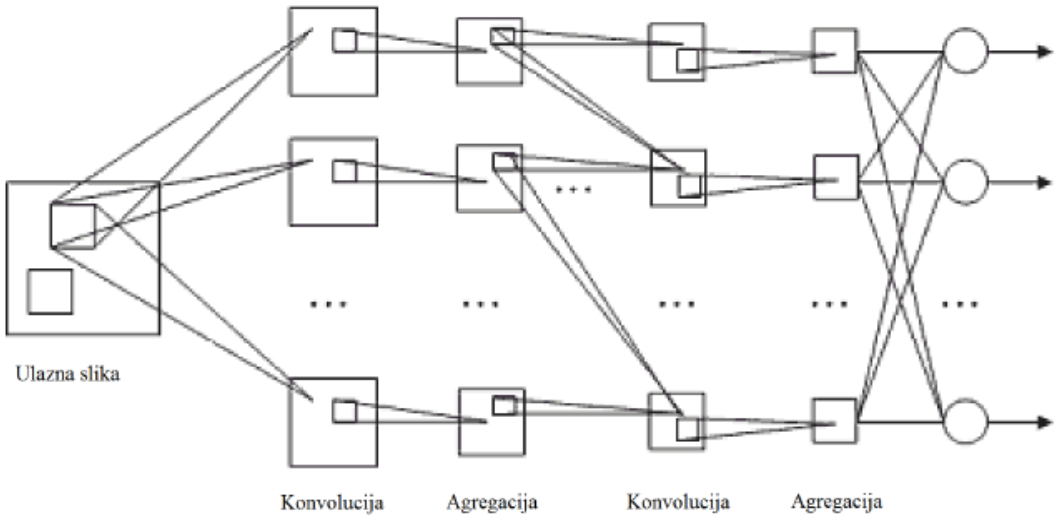
pri čemu se podrazumeva da je $g_k = g_{n+k}$ ukoliko je $k < 0$. Konvolucija se u višim dimenzijama uvodi na sledeći način:

$$(f * g)(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y)g(v - x, u - y)dxdy$$
$$(f * g)_{ij} = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} f_{kl} g_{i-k,j-l},\ i = 0, 1, ..., m - 1,\ j = 0, 1, ..., n - 1.$$

Složenost izračunavanja diskretne konvolucije dva signala prema prethodnoj formuli ima vremensku složenost $\Theta(n^2)$. Međutim, koristeći brzu Furijeovu transformaciju ova operacija se može izračunati u složenosti $\Theta(n \log(n))$.

U konvolutivnim neuronskim mrežama ulaz predstavlja sirovi signal. Konvolutivne mreže su, gotovo bez izuzetka, duboke neuronske mreže s obzirom na činjenicu da je potrebno od jednostavnih detalja detektovanih u nižim slojevima mreže

konstruistati složene oblike na njenom izlazu. Uobičajena struktura konvolutivne neuronske mreže data je na slici 2.7.



Slika 2.7: Shema tipične konvolutivne neuronske mreže

Kao što prethodna shema pokazuje, konvolutivne mreže grade se kombinacijom dve vrste slojeva, konvolutivnih slojeva (eng. *convolutional layer*) i slojeva agregacije (eng. *pooling layer*).

Konvolutivni slojevi implementiraju operaciju diskretne konvolucije date gornjim formulama. Funkcija f pomenute formule može označavati vrednost signala na određenoj lokaciji (jednodimenzionoj u slučaju zvuka, dvodimenzionoj u slučaju slike), dok funkcija g označava filter, koji je uglavnom manjih dimenzija u odnosu na dimenzije odgovarajućeg signala. Vrednosti filtera g predstavljaju parametre neuronske mreže, pa trening mreže predstavlja ništa drugo do učenje odgovarajućih filtera. Izlazi iz konvolutivnih slojeva transformišu se nelinearnim aktivacionim funkcijama, koje su diskutovane u prethodnim poglavljima. Konvolutivni slojevi imaju ulogu konstrukcije novih atributa. U praksi se implementiraju višekanalne operacije konvolucije koje omogućavaju filterima da deluju na više, a ne samo na jedan kanal prethodnog sloja.

Kao što njihovo ime govori, slojevi agregacije koriste se u svrhu agregiranja informacija odnosno njihovog ukrupnjavanja. Obično se radi o nekoj jednostavnoj funkciji koja se primenjuje na susedne jedinice prethodnog sloja poput uprosečavanja ili pronalaženja maksimuma. U mrežama koje obrađuju slike, ukoliko se agregiraju

polja veličine $k \times k$, dimenzije rezultata su k^2 puta manje od dimenzija ulaza. Uloga agregacije pre svega je u tome da se smanji broj potrebnih računskih operacija u višim slojevima mreže, kao i smanjenje broja parametara modela. Agregacijom se smanjuje zahtevnost procesa optimizacije neuronske mreže, ali se istovremeno smanjuje i njena fleksifilnost. Potrebno je naći kompromis između prethodna dva. Kao što je pokazano u relevantnim radovima, iz arhitekture konvolutivne neuronske mreže moguće je izbacivanje slojeva agregacije, pri čemu njihovu ulogu preuzimaju slojevi konvolucije (sa odgovarajućom veličinom koraka (eng. *stride*)) [36]. Ovo je u novije vreme sve češća praksa.

Na izlaze poslednjeg od ovih slojeva obično se nadovezuje potpuno povezana neuronska mreža. Njena uloga jeste učenje nad atributima koje su ekstrahovali prethodni slojevi mreže. Ove mreže se mogu koristiti kako za regresiju, tako i za klasifikaciju. U slučaju regresije, neuroni poslednjeg sloja ove mreže ne koriste aktivacionu funkciju, dok se u slučaju klasifikacije na linearne kombinacije ulaza pretposlednjeg nivoa primenjuje takozvana funkcija mekog maksimuma (*softmax*) [16]:

$$softmax(x) = (\frac{e^{x_1}}{\sum_{i=1}^{C} e^{x_i}}, ..., \frac{e^{x_C}}{\sum_{i=1}^{C} e^{x_i}})$$

koja vektor x preslikava u vektor iste dimenzije a koji se može tumačiti kao predviđanje verovatnoća ciljanih klasa.

Na samom kraju ovog poglavlja valjalo bi istaći osnovne prednosti konvolutivnih mreža u odnosu na druge tipove modela mašinskog učenja. Na prvom mestu ističe se deljenje parametara koje se odnosi na to da svi neuroni jednog kanala dele parametre, čime se umanjuje njihov ukupan broj, što opet vodi jednostavnijim modelima koje je lakše optimizovati. Svaki neuron povezan je sa malim brojem neurona prethodnog sloja, otuda dolaze proređene interakcije kao važno svojstvo ovih modela. Nadalje, konvolutivne mreže ispoljavaju neosetljivost na translacije, što se ističe kao velika prednost u obradi slika. Dodatno posebnu pažnju treba obratiti na delimičnu interpretabilnost konvolutivnih mreža, iako je ona daleko od pune interpretabilnosti nekih drugih modela mašinskog učenja, koji su po pravilu jednostavniji.

Konvolutivne mreža, iako predstavlja jako moćnu tehniku, ipak ima određene mane i nedostatke. Mreža je jako osetljiva na određene afine transformacije ravni kao što su rotacija i homotetija [16]. Dodatno, kako konvolutivna mreža na svom kraju ima manju potpuno povezanu neuronsku mrežu, s obzirom da ta mreža prihvata fiksan broj ulaza, to ulazi u konvolutivnu mrežu takođe moraju biti fiksnih

dimenzija. U novije vreme postoje tehnike koje ovo prevazilaze. Naravno, nedostaci diskutovani za potpuno povezane neuronske mreže prisutni su i ovde. Obučavanje konvolutivnih mreža zahteva veliku količinu informacija. Pristutni su takođe i problemi optimizacije.

2.6 Stratifikacija i ponovno uzorkovanje

Prilikom podele podataka neophodno je voditi računa da podaci u svim podskupovima imaju približno sličnu raspodelu kao i polazni skup podataka, s obzirom da se od pretreniranog modela mašinskog učenja ne može očekivati dobro ponašanje na podacima koji ne dolaze iz (eventualno jako slične) raspodele kao i skup podataka na kojima je on treniran. U slučaju velike količine podataka slučajna podela daje dobre rezultate. Kada skup podataka koji se deli nije tako veliki, o prethodnom je potrebno voditi računa. U tu svrhu koristi se tehnika stratifikacije (eng. *stratification*).

Stratifikacija predstavlja vid preprocesiranja podataka kojim se prilikom podele datog skupa podataka u dva ili više podskupova obezbeđuje da svi podskupovi imaju istu raspodelu atributa i ciljne promenljive kao i polazni skup podataka. U slučaju labelovanog skupa podataka, stratifikacija se najčešće sreće u pojednostavljenom obliku koji se fokusira samo na raspodelu ciljnih promenljivih i ona se naziva stratifikacija po ciljnoj promenljivoj. Podela na M skupova stratifikovana u odnosu na ciljnu promenljivu može se postići sortiranjem polaznog skupa podataka D u odnosu na vrednosti ciljne promenljive, a nakon toga, i-ti skup sačinjavaju instance sa indeksima (u sortiranom skupu) $i + j * M$, $i \in 1, .., M$, $j \in |D|/M$.

Ponovno uzorkovanje predstavlja jednostavnu tehniku evaluacije pri kojoj se iz datog skupa podataka veći broj puta nezavisno biraju uzorci određene veličine (izbor sa vraćanjem). Cilj ponovnog uzorkovanja jeste smanjenje varijanse ocene greške. Kao i tehnika stratifikacije, do izražaja dolazi u slučaju skupova podataka relativno male veličine.

Glava 3

Matematičke osnove

U ovom poglavlju biće izložena matematička pozadina neophodna za razumevanje u radu predloženog metoda poboljšanja. Pretpostavlja se da je čitalac upoznat sa osnovnim pojmovima teorije verovatnoće kao što su pojmovi slučajne veličine, gustine raspodele, nezavisnosti događaja i slično.

3.1 Bernulijeva raspodela

Bernulijeva raspodela predstavlja jednu od fundamentalnih raspodela teorije verovatnoće. Sa izrazito intuitivnom motivacijom i prilično opštom postavkom, nalazi svoju primenu u mnogim problemima. Često je upravo ona prvi izbor prilikom modelovanja procesa sa binarnim ishodima kao što su bacanje novčića (postoje dva moguća ishoda, pala je glava ili je palo pismo), broj kandidata koji su položili test (mogući ishod za svakog od kandidata je da je pao ili položio test) ili broj pacijenata koji imaju određenu bolest (mogući ishod za svakog pacijenta jeste da ima ili nema bolest).

Za diskretnu slučajnu veličinu X kažemo da ima Binomnu raspodelu [2] ako je njen zakon raspodele verovatnoća dat sa:

$$f_X(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x}, & x = 0, 1, 2, ..., n \\ 0, & inače \end{cases}$$

gde je $n \geq 1$ prirodan broj, $0 \leq p \leq 1$ parametar binomne raspodele.

U prethodnoj formuli x predstavlja broj uspešnih a $n - x$ broj neuspešnih realizacija događaja u n pokušaja. Broj $\binom{n}{x} = \frac{n!}{x!(n-x)!}$ naziva se binomni koeficijent. Ako slučajna promenljiva X ima binomnu raspodelu sa parametrima n i p to zapisujemo

kao $X : B(n, p)$. Matematičko očekivanje i varijansa ovakve slučajne veličine dati su formulama: $EX = np$, $DX = np(1-p)$. Do prethodnih formula dolazi se relativno jednostavno, recimo u slučaju matematičkog očekivanja:

$$\begin{aligned} EX &= \sum_{j=0}^{n} j \binom{n}{j} p^j (1-p)^{n-j} \\ &= \sum_{j=1}^{n} n \binom{n-1}{j-1} p^j (1-p)^{n-j} \\ &= n \sum_{j=1}^{n} \binom{n-1}{j-1} p^j (1-p)^{n-j} \\ &= np \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-i-1} \\ &= np[p + (1-p)]^{n-1} \\ &= np \end{aligned} \tag{3.1}$$

3.2 Beta funkcija i beta raspodela

Bernuli i Goldbah još u osamnaestom veku baveći se interpolacijom redova susreli su se sa problemom određivanja faktorijela velikih pozitivnih realnih brojeva. Obratili su se Ojleru za pomoć. Ojler pristupa rešavanju problema, i u toku 1729. godine dolazi do nečega što danas nosi naziv gama funkcija. Nešto kasnije biva definisana i beta funkcija.

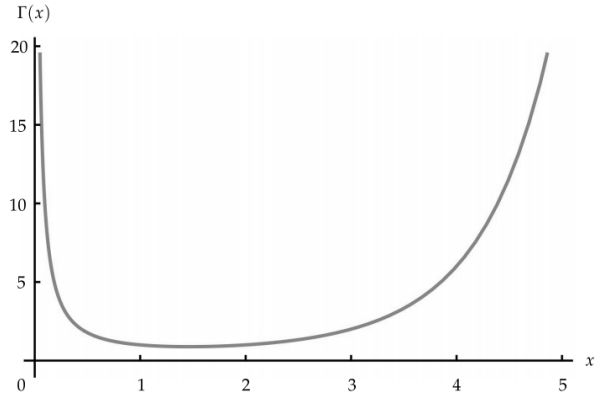
Funkciju $\Gamma : R^+ \to R^+$ definisanu sa:

$$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$$

nazivamo gama funkcijom [27]. Nju smo definisali samo za pozitivne realne brojeve, s obzirom da samo tada prethodni definicioni integral konvergira. Dodatno, prethodna definicija može se proširiti, a gama funkcija definisati za sve kompleksne brojeve kojima je realni deo pozitivan.

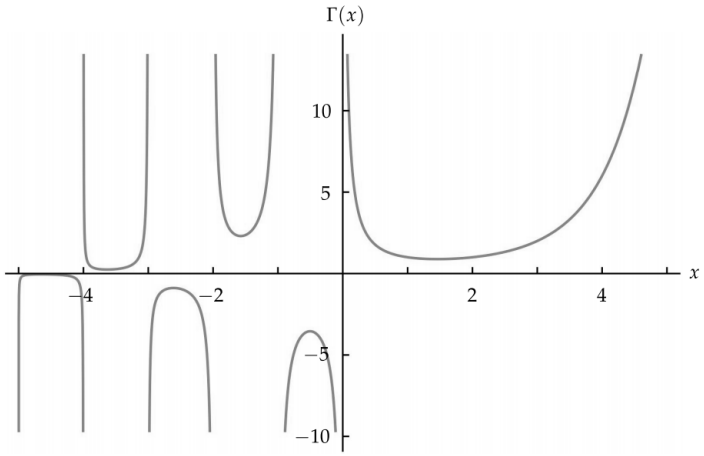
Jednostavnim računom može se doći do rezultata $\Gamma(1) = 1$. Štaviše, koristeći se osnovnim tehnikama matematičke analize može se skicirati grafik prethodno uvedene gama funkcije. Isti je dat na slici 3.1.

Koristeći se tehnikom parcijalne integracije kao i Lopitalovim pravilima, dolazimo do rezultata da za sve pozitivne realne brojeve x važi: $\Gamma(x+1) = x\Gamma(x)$. Dalje,



Slika 3.1: Gama funkcija, osnovna varijanta

na osnovu toga dobijamo da, za sve prirodne brojeve n važi: $\Gamma(n) = n!$. Koristeći prethodne formule, možemo proširiti domen gama funkcije i na negativne prirodne odnosno realne brojeve [14]. Grafik proširene gama funkcije dat je na slici 3.2.



Slika 3.2: Gama funkcija, proširena definicija

Beta funkciju definišemo koristeći prethodno uvedenu gama funkciju. Funkciju $B : R^+ \times R^+ \to R^+$ definisanu sa:

$$B(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x + y)}$$

nazivamo beta funkcijom. Moguće je dokazati da se i beta funkcija može predstaviti u obliku jednostrukog integrala. Naime, važi da je funkcija uvedena prethodnom definicijom jednaka:

$$B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt$$

Beta funkcija ima određena jako povoljna svojstva kao što su svojstvo simetričnosti $B(x,y) = B(y,x)$ i određene vrste aditivnosti po prvom argumentu $B(x+1,y) = \frac{x}{x+y}B(x,y)$. Predstavlja jako bitnu funkciju matematičke analize, koja ima mnogobrojne primene.

Jedna od važnijih primena svakako jeste i primena u definisanju beta raspodele.

U teoriji verovatnoće odnosno statistici beta raspodela predstavlja familiju neprekidnih raspodela definisanih na segmentu $[0,1]$. Raspodela je parametrizovana sa dva pozitivna parametra α i β. Ovi parametri se nazivaju parametrima oblika, s obzirom da kontrolišu oblik same raspodele. Beta raspodela jeste specijalni slučaj opštije, višedimenzione Dirihleove raspodele o kojoj na ovom mestu neće biti reči.

Beta raspodela jeste raspodela definisana gustinom:

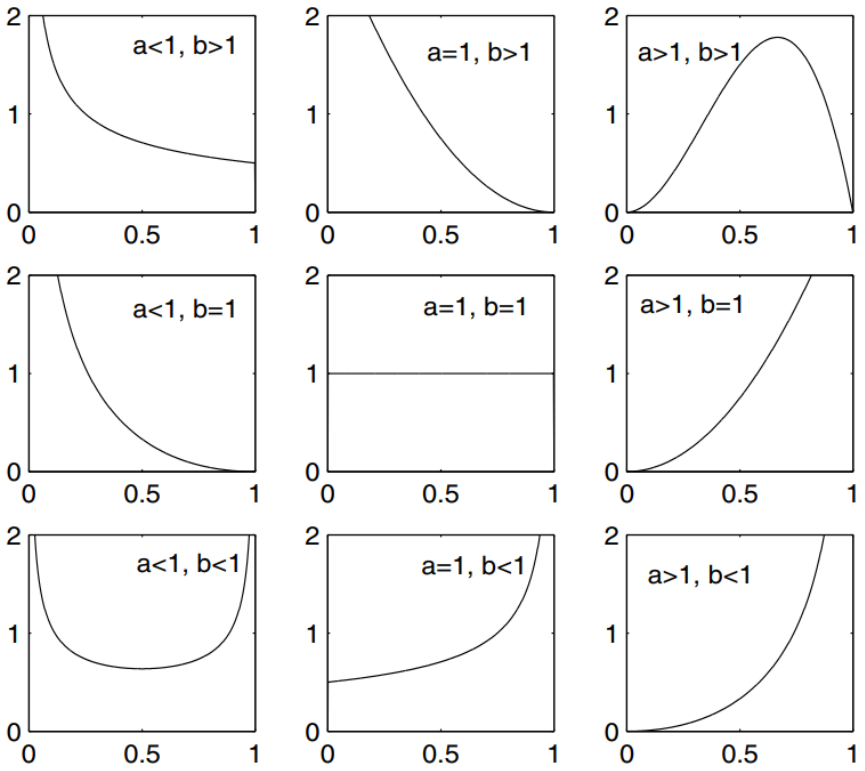
$$\begin{aligned} f_B(x;\alpha,\beta) &= \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt} \\ &= \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \end{aligned} \tag{3.2}$$

gde za argument x važi $0 \leq x \leq 1$, a za parametre oblika $\alpha \geq 0$, $\beta \geq 0$. Recipročna vrednost beta funkcije jasno predstavlja normalizujuću konstantu koja služi tome da funkcija gustine bude normirana u smislu integrala.

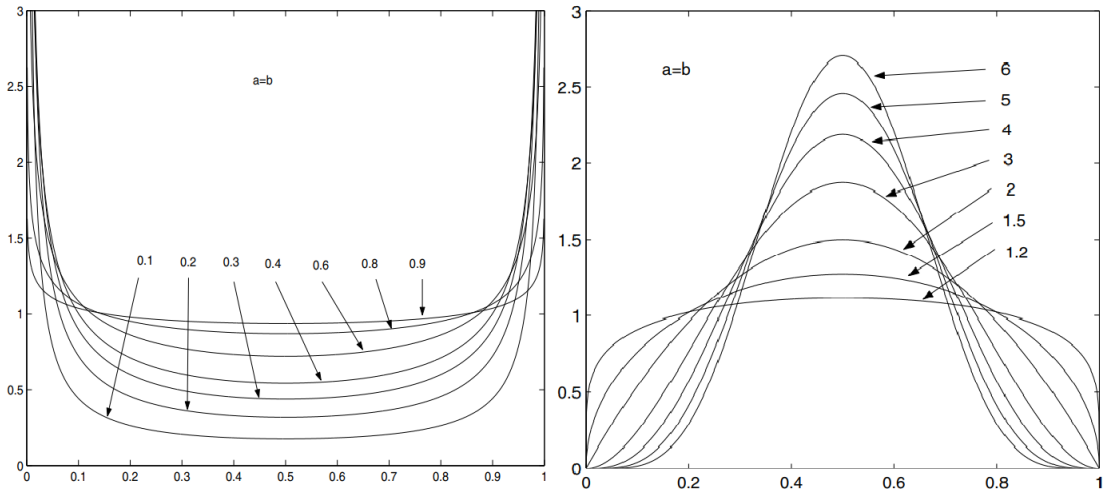
U zavisnosti od vrednosti parametara α i β postoji više karakterističnih slučajeva koji su ilustrovani na slici 3.3.

Ako je barem jedan od parametara manji od 1, odgovarajući grafik ima vertikalne asimptote.

U slučaju $\alpha = \beta$ funkcija gustine je simetrična funkcija kojoj se maksimum menja zavisno od veličine koeficijenata. Ovo je ilustrovano na slici 3.4.

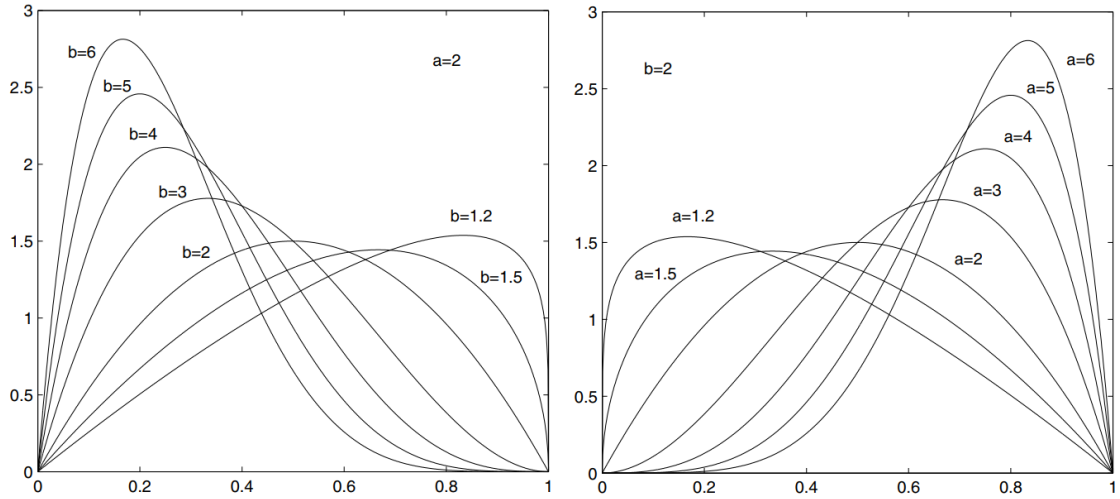


Slika 3.3: Karakteristični izgled funkcije gustine zavisno od vrednosti parametara oblika



Slika 3.4: Karakteristični izgled funkcije gustine zavisno od vrednosti parametara oblika; slučaj jednakih vrednosti parametara

Promena izgleda funkcije gustine prilikom promene jednog od parametra data je na slici 3.5.



Slika 3.5: Promena gustine prilikom menjanja jednog od parametara

Na osnovu prethodno uvedene gustine, funkcija raspodele data je sa:

$$F(x) = \frac{B_x(\alpha,\beta)}{B(\alpha,\beta)} = I_x(\alpha,\beta)$$

gde su $B_x(\alpha,\beta)$ i $I_x(\alpha,\beta)$ nepotpune beta funkcije su definisane sa:

$$B_x(\alpha,\beta) = \int_0^x t^{\alpha-1}(1-t)^{\beta-1}dt$$
$$I_x(\alpha,\beta) = \frac{B_x(\alpha,\beta)}{B(\alpha,\beta)}$$

Matematičko očekivanje μ slučajne veličine X koja ima beta raspodelu sa parametrima α i β može se predstaviti bilo kao funkcija tih parametara, bilo kao funkcija njihovog količnika [2]:

$$\begin{aligned}\mu = E[X] &= \int_0^1 x f(x;\alpha,\beta)dx \\ &= \int_0^1 x\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}dx \\ &= \frac{\alpha}{\alpha+\beta} \\ &= \frac{1}{1+\frac{\beta}{\alpha}}\end{aligned} \tag{3.3}$$

Ako su vrednosti parametara jednake, tada važi $\mu = \frac{1}{2}$ i funkcija gustine predstavljena je simetričnom funkcijom, čemu svedoči i slika 3.4.

Na osnovu prethodnih izraza, dolazimo do ponašanja matematičkog očekivanja u graničnim slučajevima [12]:

$$\lim_{\frac{\beta}{\alpha}\to 0} \mu = 1$$
$$\lim_{\frac{\beta}{\alpha}\to \infty} \mu = 0$$

Dakle, ukoliko $\frac{\beta}{\alpha} \to \infty$ ili $\frac{\alpha}{\beta} \to 0$ matematičko očekivanje ovakve slučajne veličine se degeneriše, i predstavljeno je Dirakovom delta funkcijom koja je koncentrisana na desnom kraju segmenta $[0, 1]$, odnosno koja tački $x = 1$ daje verovatnoću 1, dok ostale tačke segmenta imaju verovatnoću 0.

Analogno, ukoliko $\frac{\beta}{\alpha} \to 0$ ili $\frac{\alpha}{\beta} \to \infty$ matematičko očekivanje ovakve slučajne veličine se degeneriše, i predstavljeno je Dirakovom delta funkcijom koja je koncentrisana na levom kraju segmenta $[0, 1]$, odnosno koja tački $x = 0$ daje verovatnoću 1, dok ostale tačke segmenta imaju verovatnoću 0.

Varijansa slučajne veličine X koja ima beta raspodelu sa parametrima α i β predstavlja funkciju tih parametara [2]:

$$var(X) = E[(X - \mu)^2] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Ukoliko su vrednosti parametara α i β jednake, prethodna jednakost postaje:

$$var(X) = \frac{1}{4(2\beta+1)}$$

odakle se jasno zaključuje da varijansa monotono opada kako vrednosti parametara $\alpha = \beta$ rastu. U graničnom slučaju, kada važi $\alpha = \beta = 0$, važi $var(X) = \frac{1}{4}$. Dodatno, ponašanje varijanse u graničnim slučajevima dato je sa:

$$\lim_{\beta\to 0} var(X) = \lim_{\alpha\to 0} var(X) = \lim_{\beta\to\infty} var(X) = \lim_{\alpha\to\infty} var(X) = 0$$

Neka od svojstava beta raspodele, koja se ističu kao posebno zanimljiva data su sa [12]:

- Beta raspodela sa parametrima $\alpha = \beta = 1$ predstavlja uniformnu raspodelu na segmentu $[0, 1]$, odnosno $B(1, 1) = U([0, 1])$
- Beta raspodela sa parametrima $\alpha = \beta = 2$ predstavlja paraboličku raspodelu

- Parametri oblika α i β mogu biti izraženi u terminima matematičkog očekivanja μ i standardne devijacije, za koju na ovom mestu uvodimo oznaku $\sigma = \sqrt{var(X)}$:

$$\alpha = (\frac{1-\mu}{\sigma^2} - \frac{1}{\mu})\mu^2$$
$$\beta = \alpha(\frac{1}{\mu} - 1)$$

- Beta raspodela kao takva, nailazi na primene u Bajesovskoj statistici, meteorologiji, hidrologiji, komunikacijama i sl [12].

Beta raspodelu moguće je definisati i kao funkciju više od dva parametra o čemu neće biti diskutovano na ovom mestu, čitalac za više informacija može pogledati odgovarajuću literaturu [12].

3.3 Bajesovska statistika, konjugovane raspodele, apriorne i aposteriorne verovatnoće

Bajesovska statistika (eng. *Bayesian statistic*) predstavlja jedan vid tumačenja pojma verovatnoće, različit od klasičnog, frekventističkog pristupa verovatnoći. U klasičnom pristupu koncept verovatnoće uključuje niz ponavljanja iste situacije pri kojem se verovatnoća tumači kao frekvencija događaja. Bajesovski pristup verovatnoću tumači kao stepen uverenja u određene događaje.

Bajesova formula:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

predstavlja jednu od najvažnijih formula klasične teorije verovatnoće. Ona predstavlja fundamentalnu teoremu i u okviru bajesovskog pristupa. Koristi se u svrhu ažuriranja ocene verovatnoće koja pak predstavlja stepen verovanja u konkretne ishode određenog događaja.

Bajesovska statistika ovu teoremu interpetira na intuitivan način. A obično predstavlja hipotezu o familiji raspodela kojoj pripada slučajna promenljiva koja se razmatra. Raspodela ove familije naziva se apriornom raspodelom, dok se verovatnoća $P(A)$ naziva apriornom verovatnoćom hipoteze A. Ova raspodela uključuje prethodna znanja o fenomenu koji se razmatra. $P(A|B)$ predstavlja verovatnoću hipoteze A nakon što se u obzir uzmu rezultati eksperimenata označeni sa

B. Tako ažurirana ocena verovatnoće naziva se aposteriornom verovatnoćom hipoteze A. Aposteriornu verovatnoću dobijamo kombinujući već diskutovanu apriornu verovatnoću sa $P(B|A)$ i $P(B)$. $P(B|A)$ predstavlja verovatnoću podataka B pri hipotezi A. $P(B)$ predstavlja zbirnu verovatnoću podataka B dobijenu težinskim sumiranjem (ili integracijom) po svim hipotezama A u skladu sa njihovim apriornim verovatnoćama.

Ako apriornu raspodelu predstavimo kao raspodelu slučajne promenljive θ, njenu funkciju gustine obeležavamo sa $f(\theta)$. Naravno, pri tome parametar θ može biti bilo vektor bilo skalar. Funkciju gustine aposteriorne raspodele označavamo sa $f(\theta|D)$, gde su D dati podaci.

U teoriji Bajesovske statistike za dve raspodele kažemo da su konjugovane ukoliko aposteriorna raspodela pripada istoj faimiliji raspodela kao i apriorna raspodela, nakon što se novi podaci uzmu u obzir.

3.4 Konjugovanost binomne i beta raspodele

Posmatrajmo eksperiment sa dva moguća ishoda (označena sa 0 i 1) u kome važi da je verovatnoća jednog θ a drugog $1-\theta$. Ovakav eksperiment se očigledno može modelovati Bernulijevom raspodelom. Činjenicu da je $P(1) = \theta$ i $P(0) = 1-\theta$, pri čemu nam nije poznata tačna vrednost parametra θ zapisujemo kao $P(k|\theta) = \theta^k(1-\theta)^{1-k}$, gde $k \in \{0,1\}$, $\theta \in [0,1]$.

Ukoliko imamo situaciju izvršavanja N nezavisnih eksperimenata koji imaju istu, Bernulijevu raspodelu, verovatnoća z uspeha data je sa: $P(z|\theta) = \theta^z(1-\theta)^{N-z}$.

Aposteriorna ocenu parametra θ dobija se na osnovu apriorne ocene ovog parametra, korišćenjem Bajesove formule:

$$P(\theta|z) = \frac{P(z|\theta)P(\theta)}{P(z)} = \frac{P(z;\theta)}{\int_0^1 P(z;\theta)d\theta}$$

gde je sa $P(z;\theta)$ označena zajednička raspodela.

Koristeći se formulama:

$$P(z|\theta) = \binom{N}{z}\theta^z(1-\theta)^{N-z}$$
$$P_B(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

i činjenicom da je:

$$\binom{N}{z} = \frac{N!}{(N-z)!z!} = \frac{\Gamma(N+1)}{\Gamma(N-z+1)\Gamma(z+1)}$$

jer je $\Gamma(n+1) = n!$, dolazimo do niza jednakosti:

$$\begin{aligned} P(z;\theta) &= P(z|\theta)P(\theta) \\ &= \frac{\Gamma(\alpha+\beta)\Gamma(N+1)}{\Gamma(\alpha)\Gamma(\beta)\Gamma(N-z+1)\Gamma(z+1)}\theta^{z+\alpha-1}(1-\theta)^{N-z+\beta-1} \\ &= \gamma\theta^{z+\alpha-1}(1-\theta)^{N-z+\beta-1} \\ &= \frac{\Gamma(\alpha+\beta+N)}{\Gamma(z+\alpha)\Gamma(N+\beta-z)}\theta^{z+\alpha-1}(1-\theta)^{N+\beta-z-1}\gamma\frac{\Gamma(z+\alpha)\Gamma(N+\beta-z)}{\Gamma(\alpha+\beta+N)} \end{aligned} \tag{3.4}$$

gde je sa γ označen odgovarajući količnik proizvoda gama funkcija.

Dalje imamo:

$$\begin{aligned} P(z) &= \int_0^1 P(z;\theta)d\theta \\ &= \int_0^1 \frac{\Gamma(\alpha+\beta+N)}{\Gamma(\alpha+z)\Gamma(N+\beta-z)}\theta^{z+\alpha-1}(1-\theta)^{N+\beta-z-1}\gamma\frac{\Gamma(z+\alpha)\Gamma(N+\beta-z)}{\Gamma(\alpha+\beta+N)}d\theta \\ &= \gamma\frac{\Gamma(z+\alpha)\Gamma(N+\beta-z)}{\Gamma(\alpha+\beta+N)} \end{aligned} \tag{3.5}$$

s obzirom da je $\int_0^1 B(\alpha+z, N+\beta-z)d\theta = 1$.

Odavde dobijamo da je aposteriorna ocena parametra θ takođe pripada familiji beta raspodela:

$$\begin{aligned} P(\theta|z) &= \frac{P(z;\theta)}{P(z)} \\ &= \frac{\gamma\theta^{z+\alpha-1}(1-\theta)^{N-z+\beta-1}}{\gamma\frac{\Gamma(\alpha+\beta)\Gamma(N+\beta-z)}{\Gamma(\alpha+\beta+N)}} \\ &= \frac{\Gamma(\alpha+\beta+N)}{\Gamma(\alpha+z)\Gamma(N+\beta-z)}\theta^{z+\alpha-1}(1-\theta)^{N-z+\beta-1} \\ &= B(\alpha+z, N+\beta-z) \end{aligned} \tag{3.6}$$

Glava 4

Metod prilagođavanja

Pionirski pokušaj automatskog prepoznavanja rukom pisanog teksta datira još iz pedesetih godina prošlog veka [25]. Za to vreme, ono je predstavljalo jedan izuzetno zahtevan zadatak. Nakon ovog početnog pokušaja, nekoliko istraživača radilo je na ovom problemu pokušavši različite pristupe. Tokom poslednje decenije istraživanje ovog problema doseže svoj vrhunac, a ubedljivo najbolji rezultati postignuti su korišćenjem neuronskih mreža.

Neformalna definicija automatskog prepoznavanja rukom pisanog teksta mogla bi se formulisati na sledeći način: ono predstavlja sposobnost računara da prepozna i tumači rukom pisani tekst iz izvora kao što su papirni dokumenti, fotografije, ekrani osetljivi na dodir i drugi. Pisani tekst računaru može biti predstavljen na dva načina. Oflajn pisani tekst podrazumeva da su računaru na raspolaganju pikseli slike dobijeni na neki način ekstrahovanjem iz prethodno pomenutih izvora. Oflajn pisani tekst fokusiran je na statički prikaz rukopisa. Onlajn pisani tekst, sa druge strane, rukopis tumači kao dinamički objekat, noseći veću količinu informacija. Pamti se, i računaru stavlja na raspolaganje kompletna informacija o kretanju ruke odnosno olovke tokom pisanja dokumenta.

U ovom odeljku biće opisan predloženi model poboljšanja neuralnog klasifikatora oflajn rukom pisanog teksta. Poboljšanje se zasniva na ideji da se klasifikator prilagodi svakom pojedinačnom korisniku, odnosno njegovom načinu pisanja praćenjem grešaka baznog klasifikatora. U te svrhe koriste se tehnike klasterovanja k-sredina i metoda k najbližih suseda. Više detalja dato je u nastavku, u odgovarajućim poglavljima.

4.1 Pregled predloženog metoda povećanja preciznosti klasifikatora učenjem specifičnosti pojedinačnih korisnika

Osnovna ideja jeste da se za svaki karakter izdvoje referentni načini njegovog pisanja. Ovo je motivisano intuicijom da se svaki karakter može napisati na konačno mnogo značajno različitih načina. Izdvajanje različitih stilova pisanja svakog od karaktera postignuto je klasterovanjem. Time je, za svaki od karaktera, dobijeno po nekoliko klastera koji predstavljaju agregirane različite stilove pisanja tog karaktera. Ovim se teži raščlanjavanju različitih stilova pisanja svakog pojedinačnog karaktera.

Prilikom upotrebe, na skupu za primenu, za svakog korisnika skup karaktera čiji je on autor stratifikovano se deli na dva skupa: skup za prilagođavanje i skup za testiranje. Na skupu za prilagođavanje detektuje se korisnikov stil pisanja tako što se zapamte njegovi načini pisanja pojedinačnih karaktera. Na skupu za testiranje predloženi metod daje predviđanja uzimajući u obzir poznavanje korisnikovog stila pisanja.

Skica modela prilagođavanja, koji će biti detaljnije objašnjen u naredim odeljcima, data je sa:

- Slike iz trening i validacionog skupa za bazni klasifikator grupišu se po labelama (karakterima) i u okviru svake grupe vrši se klasterovanje. Ti klasteri predstavljaju glavne stilove pisanja za svaki od karaktera.
- Za svakog autora skupa za primenu:
  - Skup njegovih slika stratifikovano se deli na skup za prilagođavanje i skup za testiranje.
  - Na skupu za prilagođavanje, za svaki karakter identifikuju se najsličniji od već definisanih stilova pisanja, što čini autorovu istoriju pisanja.
  - Na skupu za prilagođavanje kreiraju se alternativni klasifikatori (k najbližih suseda) i vektori poverenja za sve klasifikatore.
  - Prilikom upotrebe, za svaku sliku test skupa:
    - Pored predviđanja bazne mreže izračunavaju se i predviđanja alternativnih klasifikatora.

* Na osnovu vektora poverenja svih klasifikatora, bira se predviđanje najpouzdanijeg od njih.
* Za svaki klasifikator, korišćenjem ispravne i njegove predviđene labele, ažurira se njegov vektor poverenja.

Detaljniji opis svakog od koraka, kao i objašnjenje pojmova kao što su istorija pisanja i vektor poverenja klasifikatora detaljnije su dati u narednim poglavljima.

Klasterovanje stilova pisanja pojedinačnih karaktera

Prvi korak predloženog metoda jeste klasterovanje slika u okviru skupova sa istim labelama. Klasteruju se slike trening i validacionog skupa za bazni klasifikator.

Osnovna ideja koja je motivisala ovaj korak metoda poboljšanja jeste pretpostavka da iako postoje varijacije u načinima pisanja određenog karaktera od strane pojedinačnih autora, ipak je taj skup varijacija konačan. Iako je skup svih slika rukom pisanih karaktera skup velike kardinalnosti, ipak, čak i vizuelno, deluje da možemo identifikovati manji broj varijacija rukopisa svakog pojedinačnog karaktera.

Još jedan motiv za korišćenje klasterovanja po instancama jeste činjenica da pojedinačni autor, prilikom raznih pisanja određenog karaktera nikada neće napisati dva potpuno identična. Svaki autor ima svoj stil pisanja, međutim svaki novonapisani karakter biće mala modifikacija njegovog načina pisanja tog karaktera. Uprosečavanjem ovakvih karaktera očekujemo da se varijacije u pisanju ponište, a to opet vodi kristalisanju stila odnosno načina na koji autor piše taj karakter.

Dalje se postavlja pitanje dobre reprezentacije karaktera, koja će u sebi nositi informaciju o stilu njegovog pisanja, odnosno rukopisu autora. Naivni pristup bi sliku posmatrao kao niz sirovih podataka odgovarajućih dimenzija. Napredniji pristup kao skup atributa kojima se karakteriše slika rukom pisanog karaktera koristi pretposlednji sloj neuronske mreže odnosno baznog klasifikatora [13]. Naime, pretposlednji sloj neuronske mreže daje reprezentacije ulaznih slika u nekom novom prostoru atributa u kome očekujemo da su različiti karakteri dobro razdvojeni, dok poslednji sloj neuronske mreže ima ulogu labelovanja nad tim reprezentacijama. Stoga se poslednji sloj mreže odseca, i za skup atributa koji opisuju sliku sa ulaza uzimaju se vrednosti neurona pretposlednjeg sloja potpuno povezane neuronske mreže. Reprezentacije koje daje pretposlednji sloj mreže već su korišćene u literaturi [15][20].

Pomenuta ideja uprosečavanja, zajedno sa kompletnom postavkom problema vodi jednostavnom izboru metode klasterovanja. Podaci se klasteruju metodom K

sredina. Težišta klastera predstavljaće stilove pisanja karaktera koji se klasteruju na osnovu pretposlednjeg sloja bazne mreže. Broj klastera za svaku labelu, dat je formulom:

$$k = \min(30, 1 + \max(n/1000, 4)) \tag{4.1}$$

gde n predstavlja broj slika koje se klasteruju odnosno odgovaraju konkretnoj labeli. Prethodna heuristika određena je eksperimentalno, u odnosu na posmatrane skupove podataka. Pri tome, prilikom evaluacije kvaliteta klasterovanja nisu korišćene uobičajene tehnike procene kvaliteta klasterovanja. Korišćena je metoda najbližih suseda koja je trenirana na dobijenim težištima. I za klasterovanje kao i za evaluaciju metodom najbližih suseda korišćena je euklidska metrika. Broj suseda prilikom ove evaluacije pripadao je skupu $\{1, 2, 3, 4, 5, 7, 8, 9, 10, 15\}$. Kvalitet klasterovanja ocenjivan je na osnovu najveće od preciznosti metode najbližih suseda za razne vrednosti k. Motivacija za ovakav pristup jeste činjenica da će prezentovani metod poboljšanja, u nastavku koristiti metod najbližih suseda sa ciljem pronalaska karaktera najsličnijeg tekućem u okviru istorije pisanja.

Kreiranje istorije pisanja

Nakon klasterovanja, za svaki karakter izabran je određen broj njegovih karakterističnih stilova pisanja (težišta klastera), koji su predstavljeni pomoću vektora odgovarajućih dimenzija (dimenzija vektora jednaka je broju neurona pretposlednjeg sloja baznog neuronskog klasifikatora). Drugi deo metoda poboljšanja odnosi se na fazu upotrebe modela.

Slike rukom pisanih karaktera svakog pojedinačnog korisnika skupa za primenu stratifikovano su podeljene na dva skupa: skup za prilagođavanje i test skup. Na skupu za prilagođavanje metod poboljšanja kreira istoriju pisanja svakog pojedinačnog korisnika. Na osnovu istorije pisanja će se, u fazi upotrebe, vršiti poboljšanje baznog klasifikatora. Na osnovu nje potrebno je zaključiti kada bazni klasifikator za trenutnog korisnika i njegov stil pisanja greši, i kako te greške ispraviti.

Proces kreiranja istorije pisanja korisnika sastoji se od sledećeg. Za ulaznu sliku posmatramo ispravnu labelu kao i labelu koju predviđa bazni klasifikator. Na osnovu izlaza neurona pretposlednjeg sloja baznog klasifikatora pronalazimo najbliži klaster među klasterima koji odgovaraju ispravnoj labeli, odnosno među raznim stilovima pisanja karaktera koji se nalazi na ulaznoj slici. Pretraga se vrši u odnosu na vrednost Euklidske norme. Ovom pretragom pokušava se identifikovanje stila pisanja

konkretnog karaktera od strane konkretnog korisnika sa nekim od predefinisanih stilova dobijenih klasterovanjem.

Istorija pisanja za svaki uređeni par *(predviđena labela, ispravna labela)* pamti prosek težišta klastera najbližih vektoru izlaza pretposlednjeg sloja klasifikatora. Interpretacija ovoga je sledeća: kada bazni klasifikator predvidi *predviđenu labelu* za sliku na kojoj je predstavljena *ispravna labela*, na osnovu pretposlednjeg sloja bazne mreže ovom uređenom paru se dodeli odgovarajući klaster odnosno način pisanja ispravnog karaktera. Klasteri koji se dodeljuju za isti uređeni par ne moraju biti uvek identični. Njihov prosek smatramo stilom pisanja konkretnog karaktera konkretnog korisnika.

Upotreba istorije pisanja

Nakon kreiranja istorije pisanja potrebno je, na osnovu nje, donositi odluke u kojim slučajevima verovati baznom klasifikatoru, a kada modifikovati njegova predviđanja i preinačiti ih na osnovu metode k najbližih suseda trenirane na pomenutoj istoriji pisanja. Predstavljeni metod poboljšanja, na istoriji pisanja ne kreira jedan model najbližih suseda za korekciju, već niz modela k najbližih suseda, pri čemu k uzima vrednosti $2, 4, 6, 8$ i 10.

Na osnovu istorije pisanja korisnika, kreira se *vektor poverenja* baznog klasifikatora i *vektor poverenja* metode k najbližih suseda, za svaku od prethodnih vrednosti parametra k. Pod vektorom poverenja klasifikatora podrazumeva se niz očekivanja beta raspodela, kojima se procenjuje pouzdanost klasifikatora prilikom predviđanja svake od labela. Za svaku labelu l kreira se po jedna beta raspodela. Stepen poverenja klasifikatora za labelu l predstavljen je matematičkim očekivanjem kreirane beta raspodele. Ovo očekivanje predstavlja funkciju parametara α i β. Za svaku sliku skupa za prilagođavanje, za koju pomenuti klasifikator predvidi labelu l, ažuriraju se parametri α i β beta raspodele u zavisnosti od toga da li je predviđanje ispravno ili ne. Ispravna predviđanja uvećavaju vrednost parametra β (time povećavaju matematičko očekivanje beta raspodele, odnosno stepen poverenja u klasifikator kada on predviđa labelu l) dok negativna uvećavaju vrednost parametra α (time smanjujući matematičko očekivanje beta raspodele). Uticaji parametara oblika beta raspodele na vrednost njenog matematičkog očekivanja objašnjeni su u glavi 3.2.

Vektor poverenja svakog od klasifikatora ažurira se na osnovu originalnih labela i njegovih predviđanja za slike istorije pisanja. Pri tome, skup za treniranje metoda k najbližih suseda predstavlja podskup korisnikove istorije pisanja, i čine ga vektori i

njihove originalne labele svih uređenih parova istorije pisanja kod kojih je *predviđena labela* jednaka trenutnom predviđanju baznog klasifikatora.

Time smo na skupu za prilagođavanje kreirali odgovarajuće vektore poverenja kojima ocenjujemo pouzdanost kako baznog klasifikatora tako i metoda k najbližih suseda, za razne vrednosti k. Na osnovu toga, određujemo kome od njih da verujemo tokom upotrebe (verujemo onom klasifikatoru koji za datu sliku ima najveću pouzdanost na karakteru kog predviđa).

Prilikom upotrebe, na skupu za testiranje umesto samo jednog predviđanja koje je u stanju da nam da bazni klasifikator, simultano dobijamo više njih. Za konkretnu sliku, na osnovu predviđanja baznog klasifikatora i autorove istorije pisanja, dobijamo predviđanja metoda najbližih suseda za razne vrednosti k. Ako bazni klasifikator predviđa labelu l, u okviru istorije pisanja posmatramo vektore stila svih uređenih parova *(predviđena labela, ispravna labela)* kod kojih je predviđena labela upravo l. Nad tim vektorima, za svako posebno k, vršimo odlučivanje metodom k najbližih suseda u odnosu na ispravne labele. Dalje, na osnovu vektora poverenja baznog klasifikatora koji se odnosi na predviđenu labelu l, kao i na osnovu vektora metoda k najbližih suseda za razne vrednosti k, koji se pak odnose na njihove predviđene labele, odlučujemo koji karakter predviđamo.

Nakon evaluacije na pojedinačnoj slici ažuriramo parametre odgovarajućih statistika za svaki od klasifikatora. U zavisnosti od toga da li je klasifikator predvideo ispravnu labelu ili ne, ažuriramo parametre beta raspodele kojom se ocenjuje pouzdanost tog klasifikatora na predviđenom karakteru. Ovo odgovara realnoj primeni, u kojoj bi korisnik, u slučaju da aplikacija za prepoznavanje rukopisa nije uspešno klasifikovala njegov upravo napisani karakter isti ispravio, te bi aplikacija imala informaciju o ispravnoj labeli. U slučaju izostanka ispravke, model je siguran da je uspešno klasifikovao upravo obrađeni karakter. Ovakvim načinom izvršavanja prezentovani metod postiže sve bolje rezultate kako se duže izvršava, jer je stepen njegove pouzdanosti direktno proporcionalan veličini istorije pisanja. Pomenuto takođe predstavlja veliku prednost prilikom upotrebe prezentovanog modela poboljšanja u praksi.

4.2 Tehnički detalji implementacije

Implementacija rešenja napisana je u programskom jeziku *Python* [34], verziji 3.7. U pomenutom jeziku, korišćene su sledeće biblioteke:

- numpy [30] - za numerička izračunavanja i vektorsku aritmetiku
- scipy [21], skimage i cv2 [3] - za osnovne operacije nad slikama u fazi pripreme skupa podataka
- matplotlib [18] - za iscrtavanje grafika
- keras [5] - za implementacije metoda i tehnika mašinskog učenja, konkretno neuronskih mreža
- sklearn [32] - za tehnike klasterovanja i algoritma najbližih suseda
- scipy.beta i random - za generisanje statističkih modela

Kompletan izvorni kod projekta javno je dostupan[1], i strukturiran je na sledeći način:

- Sveska *NIST_ Data_ Util.ipynb* sadrži kod kojim se iz originalne NIST baze podataka parsiraju potrebne informacije o slikama, njihovim autorima i labelama.
- Sveska *NIST_ Data_ Util_ Advanced.ipynb* sadrži kod kojim se vrši preprocesiranje i kreiranje odgovarajućeg skupa podataka.
- Sveska *NIST_ Baseline_ Training.ipynb* sadrži kod kojim se kreira bazna arhitektura klasifikatora.
- Sveska *NIST_ Clustering_ Knn_ Evaluation.ipynb* - sadrži implementaciju poboljšanja baznog klasifikatora i evaluaciju poboljšanja.

U okviru sveske *NIST_ Data_ Util_ Advanced.ipynb* nalaze se sledeće funkcije:

- Funkcija `add_gaussian_noise(img)` na sliku prosleđenu joj kao argument primenjuje Gausov filter kod koga je standardna devijacija Gausovog kernela postavljena na 1
- Funkcija `crop_image(path_to_image, pad=1)` kao prvi argument dobija putanju do slike na disku, a kao rezultat vraća isečen karakter sa debljinom okvira od *pad* piksela

[1] Na adresi *github.com/MilanCugur/NNClassifierImprove*

- Funkcija `square_image(img, pad=2)` vrši centriranje prosleđenog karaktera *img* u kvadratnu sliku
- Funkcija `resize_image(img, box_size)` vrši skaliranje date kvadratne slike u dimenzije $box\_size \times box\_size$ korišćenjem bikubne interpolacije
- U okviru pasusa `Create Folder ImgDiscAdvanced` kreira se skup podataka koji će biti korišćen u okviru ovog projekta.

U okviiru sveske *NIST_ Baseline_ Training.ipynb*:

- U okviru pasusa `Set up ImageDisk and Info Disk` izvršena su učitavanja slika i pratećih informacija (autor, labela) u odgovarajuće strukture
- Definisana je klasa `OneHot` koja implementira binarno kodiranje (end. *dummy coding*). Interfejs klase predstavljaju funkcije za enkodiranje i dekodiranje
- U okviru pasusa `Model` kreiran je i natreniran bazni klasifikator predstavljen konvolutivnom neuronskom mrežom.
- U okviru pasusa `Data` izvršena je podela podataka na skupove za trening, validaciju i test. Pri tome, podela je vršena tako da se autori sa najviše napisanih karaktera nalaze u skupu za testiranje, s obzirom da metod preferira što veći broj karaktera pojedinačnog korisnika.
- U okviru pasusa `Training` izvršeno je obučavanje mreže.

U okviru sveske *NIST_ Data_ Util_ Advanced.ipynb*:

- Pasusi `Set up ImageDisk and Info Disk`, `OneHot` i `Data` vrše funkciju kao i u prethodnoj svesci
- U okviru pasusa `Clustering by instances` implementirano je klasterovanje dimenzionih vektora koji predstavljaju izlaze pretposlednjeg sloja bazne neuronske mreže slika trening i validacionog skupa. Korišćen je algoritam K sredina, sa 10 početnih inicijalizacija, maksimalnim brojem iteracija 300, i vrednosti parametra tolerancije od $1e^{-4}$.
- U okviru pasusa `Density class` implementirana je klasa *DensityClass*. Detaljniji opis klase dat je narednim poglavljem.

- U okviru pasusa `With Stratify, multiple k idea` izvršena je evaluacija modela poboljšanja bez ponovnog uzorkovanja.
  - Funkcija `find_minimal(i, vector)` kao argumente prima odgovarajuću labelu i i vektor, i kao svoj rezultat vraća klaster na najmanjem rastojanju u odnosu na euklidsku metriku od prosleđenog vektora *vector* od svih klastera (stilova pisanja) koji odgovaraju labeli i.
  - Funkcija `get_min_label(history, predicted_vector)` kao argumente prima istoriju pisanja korisnika i izlaz pretposlednjeg sloja baznog klasifikatora kada se na ulazu nađe trenutna slika, i vraća predviđanja algoritama k najbližih suseda za svako k od $2, 4, 6, 8, 10$
  - Funkcija `who_to_believe(predicted_label, writer_network_spectar, min_labels, writer_knn_spectars)` kao argumente prima: labelu koju predviđa bazni klasifikator *predicted_ label*, vektor poverenja baznog klasifikatora *writer_ network_ spectar*, niz labela koje predviđaju algoritmi najbližih suseda za razne vrednosti k *min_ labels*, kao i niz vektora poverenja klasifikatora metodom k najbližih suseda za razne vrednosti k *writer_ knn_ spectars*. Kao rezultat, funkcija vraća optimalno k ili *None* što po konvencji predstavlja indikator verovanja baznom klasifikatoru
  - Funkcija `flip_an_unfair_coin(p)` kao argument dobija verovatnoću p i vraća ishod bacanja pristrasnog novčića koji daje glavu sa verovatnoćom p, a pismo sa verovanoćom $1 - p$, $p \in [0, 1]$
  - Funcija `stratify_me(images, labels)` vrši podelu datog labelovanog skupa slika stratifikovanu u odnosu na prosleđene labele
- U okviru pasusa `With Stratify, multiple k idea, RESAMPLE` izvršena je evaluacija modela poboljšanja korišćenjem ponovnog uzorkovanja.

Direktorijum *NIST_ weights* sadrži odgovarajuće koeficijente pretreniranog baznog klasifikatora.

Opis implementacije klase *CharStatistic*

U ovom delu biće opisana implementacija osnovne klase koja se koristi za aproksimaciju gustine raspodele slučajne veličine koja predviđa stepen poverenja jednom od dva klasifikatora, na svakom pojedinačnom karakteru. Drugim rečima za svakog

pojedinačnog korisnika i za svaki pojedinačni karakter nezavisno se posmatra preciznost osnovnog klasifikatora u odnosu na novi, u radu prezentovan klasifikator, koji se zasniva na korisnikovoj istoriji pisanja.

Klasa *CharStatistic* implementirana je kao klasa u programskom jeziku Python. Zasnovana je na sledećim paketima pomenutog jezika:

- Modul *numpy* [30] koristi se za vektorska numerička izračunavanja
- Paket *stats* modula *scipy* [21] koristi se za simulaciju beta raspodele
- Paket *pyplot* modula *matplotlib* [18] koristi se za vizuelizaciju gustine raspodele koja se aproksimira

Klasa od atributa sadrži polja *label* koje predstavlja labelu odnosno identifikaciju karaktera za koji se ocenjuje funkcija raspodele. Polja α i β predstavljaju parametre oblika beta raspodele koja se ocenjuje.

Pored odgovarajćih funkcija kojima se mogu postaviti i pročitati trenutne vrednosti ovih parametara, klasa implementira i funkcije koje, u zavisnosti od trenutne vrednosti parametara oblika, vraćaju očekivanje i standardnu devijaciju funkcije raspodele (prema formulama definisanim u prethodnom poglavlju). Za svakog pojedinačnog korisnika, na svakom pojedinačnom karakteru instancira se po jedna instanca ove klase za osnovni klasifikator i za novi pristup. Kako pristužu novi karakteri, u zavinosti da li je klasifikator koji se ocenjuje (osnovna mreža ili novi pristup) bio u pravu ili ne, ažuriraju se vrednosti parametara oblika. Ovo se postiže putem interfejsa kog implementiraju funkcije *update_one* i *update_all*. U slučaju ispravnog predviđanja parametar α se uvećava, inače uvećavanje se odnosi na parametar β. Promenom ovih parametara, menja se i očekivanje same beta raspodele, koje predstavlja aproksimaciju poverenja klasifikatoru [1].

Funkcija *print_current_density_function* ilustruje trenutnu gustinu procenjuće raspodele. Instanca se podrazumevano kreira sa parametrima $\alpha = \beta = 1$, odnosno na početku se za apriornu raspodelu podrazumeva uniformna raspodela na segmentu $[0, 1]$. Klasa dodatno predefiniše ponašanje operatora $>$.

U okviru Jupyter sveske *NIST_Clustering_Knn_Evaluation.ipynb* u odeljku *DensityClass* implementirana je pomenuta klasa.

Glava 5

Evaluacija rešenja

Na ovom mestu biće diskutovani rezultati predloženog metoda. U radu je korišćen skup podataka NIST Special Database 19 Američkog nacionalnog instituta za standarde i tehnologiju kao i skup rukom pisanih kraktera kreiran od strane istraživača ETH Univerziteta u Cirihu. Na prvom od njih u ovom radu predloženi metod poboljšanja povećava preciznost baznog klasifikatora za oko $2.3 - 2.5\%$, dok je na drugom povećanje preciznosti baznog klasifikatora veće od 2.7%. Ovo predstavlja značajno poboljšanje. Pored povećanja preciznosti predviđanja u obzir treba uzeti i druge kvalitete predloženog metoda. Metod umesto jednog klasifikatora simultano koristi predviđanja niza metoda k najbližih suseda, za razne vrednosti k. Ovi metodi izvršavaju se u vreme predviđanja baznog klasifikatora koristeći izuzetno male resurse. Metod ne zahteva dodatno treniranje neuronske mreže, što ga kvalifikuje kao pogodnog za izvršavanje na manjim uređajima, bez grafičkih karti. Ovo predstavlja izuzetno važnu prednost prezentovanog modela, koja do izražaja dolazi prilikom praktične primene. Dodatno, metod ne samo da ispoljava svojstvo skalabilnosti, već sa povećanjem količine napisanih karaktera korisnika povećava svoju preciznost, s obzirom da ocena parametara odgovarajuće beta raspodele postaje sve pouzdanija sa povećanjem količine podataka.

Hardverska arhitektura na kojoj su vršena izračunavanja jeste javno dostupna aplikacija kompanije Google pod nazivom Google Colaboratory koja predstavlja platformu dostupnu svim istraživačima i entuzijastima. Resursi na kojima je model treniran su dvojezgarni procesor *Intel(R) Xeon(R) CPU @ 2.30GHz* sa 13GB radne memorije i grafička karta sa *11GB* memorije.

5.1 Skupovi oflajn rukom pisanih karaktera

Postoji veći broj skupova podataka ove oblasti. Izdvojićemo najpopularnije od njih:

- *MNIST dataset* [24]
- *NIST dataset* [17]
- *IAM Handwritting Database* [26]
- *CVL Database, TU Wien* [23]
- *Bangala Hadwritten Characters* [11]

Trenutno je aktuelno i nekoliko takmičenja sa istom tematikom.

Kao što je konstatovano u uvodnim poglavljima, tema ovog rada je poboljšanje klasifikatora oflajn rukom pisanog teksta, na nivou pojedinačnih karaktera. Dosadašnji rezultati ove oblasti izrazito variraju od problema do problema, i zavise od raznih faktora. Broj karaktera, kvalitet i rezolucija slika, debljina i oštrina traga olovke i slično. Klasifikatori na pojedinim skupovima podataka dostigli su visoku preciznost, dok na drugima i dalje ne postoji rešenje zadovoljavajuće preciznosti.

5.2 Korišćeni skupovi oflajn rukom pisanih karaktera

Prilikom rešavanja problema mašinskog učenja u kojima se koriste zahtevne arhitekture (kojima su stoga potrebne velike količine podataka) jako je važno imati na raspolaganju skupove podataka odgovarajućih osobina i obima. U njihovom nedostatku prosto nije moguće izvršiti optimizaciju na pravi način. Prethodno predstavlja ogromnu barijeru.

Ideja ovog rada zasniva se na prilagođavanju baznog klasifikatora pojedinačnom korisniku, te je stoga potreban skup podataka sa velikom količinom podataka o pojedinačnom korisniku, da bi model na tim podacima naučio specifičnosti koje karakterišu svakog od njih.

Kao što je rečeno u prethodnom odeljku, u radu su korišćeni skup podataka NIST Special Database 19 Američkog nacionalnog instituta za standarde i tehnolo-

giju i skup rukom pisanih karaktera grupe istraživača ETH Univerziteta u Cirihu. Detaljniji prikazi ovih skupova podataka dati su u nastavku.

MNIST

Skup podataka *MNIST* predstavlja zasigurno najpopularniji skup podataka koji se koristi u oblasti mašinskog učenja. Isti je objavljen 1998. godine. Njegovi autori su *Yann LeCun*, *Corinna Cortes* i *Christopher J.C. Burges*.

Trening skup se sastoji od 60000 slika, dok je broj slika u test skupu 10000, nastalih iz baze *NIST Special Database 19* Američkog nacionalnog instituta za standarde. Slike su normalizovane, monohromatske, rezolucije 28×28 piksela.

Javno objavljeni rezultati pojedinih klasifikatora na ovom skupu podataka dostižu skoro stoprocentnu preciznost.

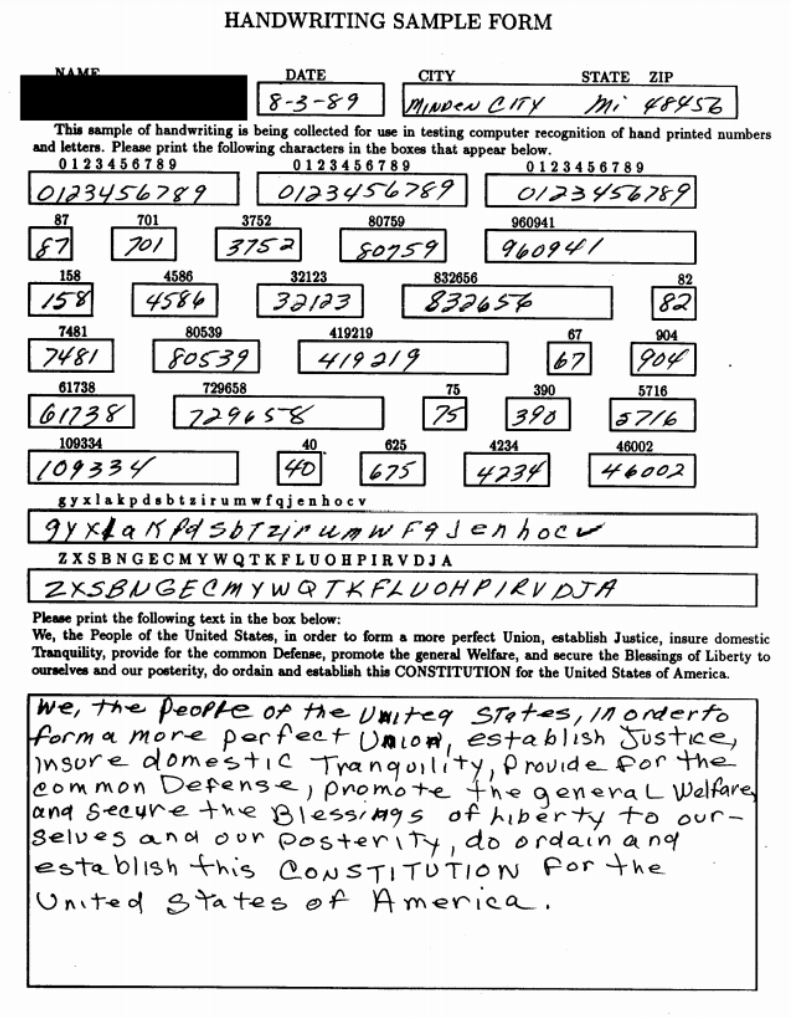
Preprocesiranja u ovom radu motivisana su MNIST skupom podataka (kao i odgovarajučim proširenjem, EMNIST-om[9]).

NIST Special Database 19

Kompletan korpus za trening i prepoznavanje rukopisa Američkog nacionalnog instituta za standarde i tehnologiju nalazi se u bazi *NIST Special Database 19*. Ovaj skup podataka sadrži približno rukopise 3600 autora u vidu približno 810000 slika proparsiranih iz odgovarajućih formi, zajedno sa odgovarajućim labelama. Dodatno, baza sadrži referentne obrasce za eventualno dalje prikupljanje podatka i softverske alate za rad sa istim.

Prva verzija ove baza podataka objavljena je (kao CD-ROM) još 1995. godine. U radu je korišćena verzija dva ove baze, objavljena u avgustu 2016. godine. Originalni CD sadrži binarizovane slike nastale na osnovu 3669 formi i sadrži 814255 segmentiranih, ručno klasifikovanih, rukom pisanih slova i brojeva. Ovi karakteri predstavljeni su monohromatski u rezoluciji 128×128 dok su labelovani jednom od 62 ASCII klase koje odgovaraju karakterima engleskog alfabeta „A”-„Z”, „a”-„z” i ciframa od „0” do „9”. Baza podataka strukturirana je na pet različitih načina, hijerarhijski organizovanim po raznim kriterijumima. U ovom radu, korišćeni su podaci strukturirani po odgovarajućim autorima odnosno klasama.

Primer jedne od formi na osnovu kojih su prikupljani karakteri dat je na slici 5.1.



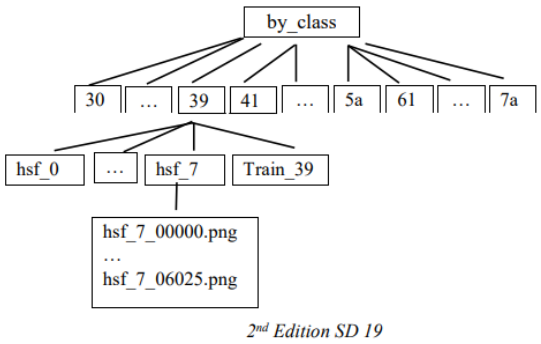
Slika 5.1: Forma za prikupljanje podataka, baza NIST

Odgovarajuće hijerarhije koje odgovaraju strukturiranju na osnovu labela i autora date su na slikama 5.2 i 5.3. Na osnovu njih bilo je moguće proparsirati sve potrebne informacije o slikama.

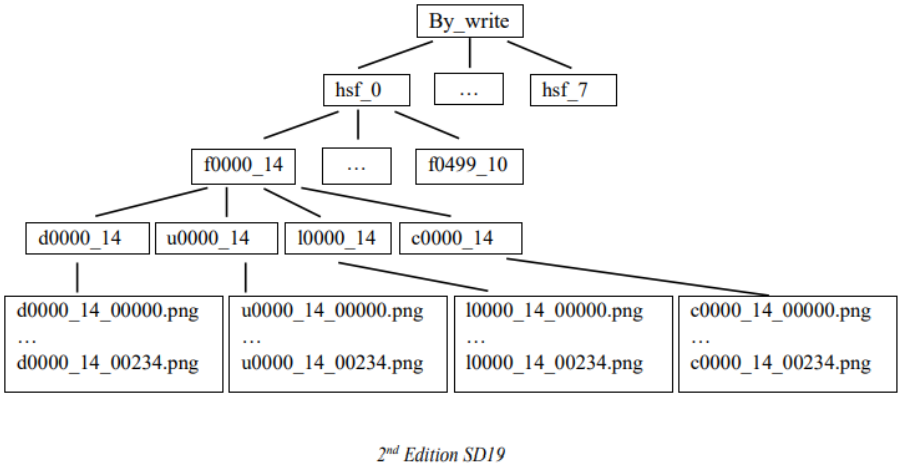
Broj slika svake od klasa, za pomenute dve hijerarhije dat je na slici 5.4

Skup *Deepwriting* rukom pisanih karaktera ETH Univerziteta u Cirihu

Ovaj skup podataka u sebi sadrži rukom pisani tekst anotiran na nivou rečenice, reči i pojedinačnih karaktera. Nastao je kao nadogradnja postojeće baze *IAM Handwritting Database*[26] koja je na ovom mestu labelovana na nivou pojedinač-



Slika 5.2: Hijerarhija baze organizovana u odnosu na labele

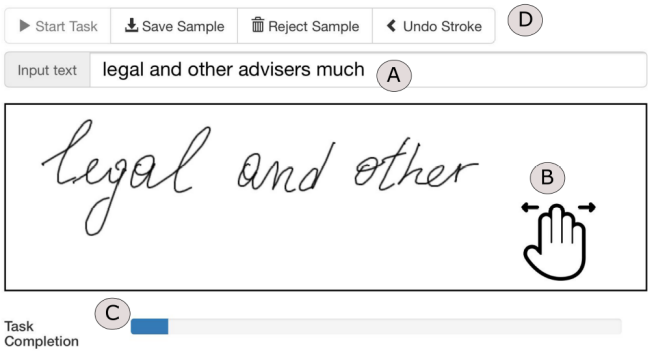


Slika 5.3: Hijerarhija baze organizovana u odnosu na autore

	Type	No. Classes	Training	Testing	Total
By Class	Digits	10	344,307	58,646	402,953
	Uppercase	26	208,363	11,941	220,304
	Lowercase	26	178,998	12,000	190,998
	Total	**62**	**731,668**	**82,587**	**814,255**
By Merge	Digits	10	344,307	58,646	402,953
	Letters	37	387,361	23,941	411,302
	Total	**47**	**731,668**	**82,587**	**814,255**

Slika 5.4: Hijerarhije u brojkama [17]

nih karaktera. Pored podataka ove baze, novi podaci prikupljeni su uz pomoć veb alata korišćenjem uređaja *iPod Pro*. Ilustracija procesa prikupljanja novih podataka putem veb interfejsa data je na slici 5.5. Novi autori, njih 94, pisali su tekst *Lancaster-Oslo-Bergen (LOB)*[37] s obzirom da su taj tekst pisali i autori u IAM bazi podataka.



Slika 5.5: Veb interfejs za prikupljanje novih podataka baze Deepwriting. Korisniku je prezentovan tekst koji je potrebno da napiše (A), deo za unos teksta označen je sa (B), dužina teksta koji se unosi označena je sa (C) dok su moguće komande označene sa (D)

Ovaj skup podataka objavljen je aprilu 2018. godine i sadrži rukopise 294 autora, koji su ukupno napisali 85181 reč odnosno 406956 karaktera. Prosečan broj napisanih reči po korisniku iznosi 292, dok je prosečan broj karaktera po korisniku 1349. Detaljnije informacije o podacima date su na slici 5.6.

	IAM-OnDB	Ours	**Unified**
Avg. Age (SD)	24.84 (± 6.2)	23.55 (± 5.7)	**24.85 (± 6.19)**
Females %	34.00	32.63	**33.55**
Right-handed %	91.50	96.84	**93.22**
# sentences	11242	63182	**17560**
# unique words	11059	6418	**12718**
# word instances	59141	26040	**85181**
# characters	262981	143975	**406956**

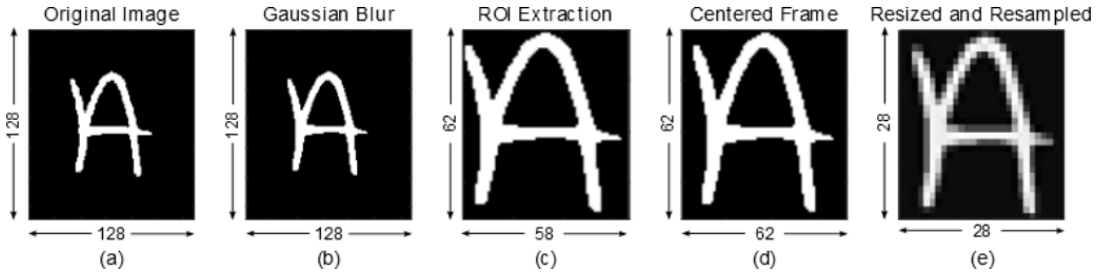
Slika 5.6: Skup rukom pisanih karaktera Deepwriting skupa podataka

Karakteri u bazi inicijalno su dati kao onlajn rukom pisani karakteri. Uz podatke, autori su obezbedili softver kojim je moguće dobiti bazu u oflajn obliku. Na osnovu obezbeđenog softvera, kreirane su oflajn slike karaktera. One su u okviru ovog rada transformisane u monohromatske karaktere rezolucije 28×28. Broj labela u bazi je 70, i to su mala i velika slova engleskog alfabta, arapski brojevi, i dodatno karakteri: ’ . , - ( ) / i razmak.

5.3 Preprocesiranje podataka

U ovom odeljku biće diskutovano preprocesiranje podataka korišćeno u radu. Na oba skupa podataka, Nist Special Database 19 i Deepwriting bazi podataka rukom pisanih karaktera, preprocesiranje je rađeno na isti način a po uzoru na rad koji opisuje kreiranje EMNIST baze podataka [9].

Ilustracija procesa konverzije, preuzeta iz pomenute publikacije, data je na slici 5.7. U pomenutom radu proces konverzije sastoji se od sledećih koraka, redom: dodavanje Gausovog zamućenja, isecanje samog karaktera sa slike, centriranje prethodno isečenog karaktera kao i promena dimenzija slike u 28×28.



Slika 5.7: Proces konverzjie slika NIST baze [9]

Za razliku od pomenute publikacije, u implementaciji se prvo vrši isecanje originalne slike predstavljene u rezoluciji 128×128, pri čemu se veličina margine oko karaktera postavlja na jedan piksel. Nakon toga, na tako isečeni karakter primenjuje se Gausov filter kod koga je standardna devijacija Gausovog kernela postavljena na 1. U toku preprocesiranja podataka testiran je i redosled transformacija originalne slike koji promoviše pomenuti rad (dakle isecanje tek nakon zamućenja slike), čime se dobijaju jako slični rezultati, te je stoga primenjen prvi pristup.

Nakon kropovanja i dodavanja zamućenja vrši se centriranje karaktera u kvadratni okvir. Prilikom centriranja, važno je naglasiti da se ne menja rezolucija karaktera, već se kraća dimenzija slike proširuje praznim prostorom. Ova transformacija od slike dimenzje $w \times h$ kreira sliku dimenzije $max\{w,h\} \times max\{w,h\}$. Originalni odnos visine i širine karaktera predstavlja jednu od specifičnosti rukopisa, stoga bi gubitak ovog odnosa predstavljao gubitak informacije.

Nakon toga kvadratna slika dimenzija $max\{w,h\} \times max\{w,h\}$, gde su w i h originalne dimenzije karaktera, konvertuje se u dimenzije 28×28. Pomenuta konverzija vrši se korišćenjem bikubne interpolacije. Finalna rezolucija izabrana je po uzoru na

skup podataka $MNIST$, odnosno $EMNIST$. Težina problema (uslovljena brojem klasa odnosno slika) ne dozvoljava veću rezoluciju karaktera. Ovo potvrđuje i skup podataka *DoubledMNIST* i rezultati na njemu [10]. Ovaj skup podataka sastoji se od slika (nastalih takođe na osnovu NIST Special Database 19) dimenzija 56×56 na kojima su bez većeg napora natrenirani klasifikatori preciznosti jako bliske 1.

Skupovi podataka, kreirani na opisan način, dostupni su u *Zip* formatu.

5.4 Podela podataka

Podaci iz baze podeljeni su u trening skup za bazni klasifikator, validacioni skup za bazni klasifikator i skup za primenu. Dalje se skup za primenu deli na dva dela: skup za prilagođavanje i test skup. Prilikom podele podataka, sve slike jednog autora pripadaju istom skupu. Obzirom na ideju predloženog pristupa, koja podrazumeva učenje specifičnosti autorovog stila na svakom pojedinačnom karakteru, potreban je značajan broj slika za svaki pojedinačni karakter svakog pojedinačnog korisnika. Kako je originalni skup podataka nebalansiran, podela je vršena tako da su oni korisnici koji su pisali više prebačeni u skup za primenu, dok su korisnici sa manje napisanih karaktera ostali u skupovima za trening i validaciju baznog klasifikatora.

Podela podataka baze NIST

Okvirno deset procenata svih slika sačinjava skup za primenu, dok su preostale slike podeljene u skupove za treniranje i validaciju baznog klasifikatora tako da je broj autora u njima u odnosu $9:1$. Podela je:

- trening skup za bazni klasifikator: 3057 autora
- validacioni skup za bazni klasifikator: 339 autora
- skup za primenu: 200 autora

U terminima broja slika, podela je sledeća:

- trening skup za bazni klasifikator: 659541 slika
- validacioni skup za bazni klasifikator: 73209 slika
- skup za primenu: 81505 slika

Važno je istaći da je skup podataka NIST izrazito nebalansiran, u smislu da za svakog pojedinačnog korisnika postoji dosta veći broj slika koje su labelovane nekim od brojeva nego ostalim karakterima. Uprkos svemu ovome, model poboljšanja koji se ovim radom prezentuje pokazao se jako uspešnim. Na osnovu toga, realno je očekivati još bolje ponašanje sistema u slučaju povoljnijih (u smislu prethodno istaknutih nedostataka) skupova podataka, i u realnoj primeni.

Podela podataka baze Deepwriting

Princip podele podataka u skupu podataka Deepwriting prati podelu podataka baze NIST. Podela je data sa, u terminima broja autora:

- trening skup za bazni klasifikator: 242 autora
- validacioni skup za bazni klasifikator: 27 autora
- skup za primenu[1]: 15 autora

U terminima broja slika, podela je sledeća:

- trening skup za bazni klasifikator: 298313 slika
- validacioni skup za bazni klasifikator: 33484 slika
- skup za primenu: 37658 slika

5.5 Treniranje baznog klasifikatora

Bazni klasifikator čije se poboljšanje u ovom radu prikazuje predstavlja konvolutivnu neuronsku mrežu baziranu na arhitekturi *CaffeNetwork* [19]. Za kreiranje arhitekture i trening mreže korišćena je biblioteka *Keras* programskog jezika *Python*. Bazni klasifikator treniran je na slikama trening skupa a validiran na slikama validacionog skupa.

Arhitektura

Pomenuta neuronska mreža trenirana je na slikama dimenzija 28×28, dok se na njenom izlazu nalazi funkcija mekog maksimuma (eng. *softmax*) odnosno sloj koji

[1] 90% slika svakog autora čini skup za prilagođavanje, dok preostalih 10% slika čini test skup

kao klasu predviđa, u slučaju baze podataka NIST jednu od *62* labele, dok u slučaju skupa podataka Deepwriting jednu od 70 labela. Labele su kodirane ortonormiranim sistemom vektora. Bazni neuronski klasifikatori kreirani za dva skupa podataka koja se u ovom radu razmatraju razlikuju se samo u broju neurona poslednjeg sloja.

Konvolutivni deo neuronske mreže sastoji se od tri uzastopna bloka, od kojih se svaki sastoji od dve uzastopne konvolucije (eng. *convolutional layer*), nakon kojih sledi unutrašnja normalizacija (eng. *batch normalization*) koja je opet praćena slojem agregacije (eng. *pooling layer*). Broj filtera u konvolutivnim slojevima monotono raste po blokovima. Konvolucije koriste *ispravljenu linearnu jedinicu* (eng. $ReLU$) kao aktivacionu funkciju. Agregiranje se vrši korišćenjem maksimuma kao funkcije agregacije, zadržavajući iste dimenzije slike nakon agregiranja. Na konvolutivni deo nadovezuje se potpuno povezana neurnoska mreža, u kojoj se koristi regularizacija izostavljanjem (eng. *dropout*). Kompletnu arhitekturu moguće je videti na slici 5.8. Kompletna mreža sadrži nešto više od *350 000* parametara.

Treniranje i rezultati

Bazni neuronski klasifikator treniran je optimizacionim metodom Adam sa parametrima $\beta_1 = 0.9$, $\beta_2 = 0.999$ sa korakom učenja 0.001.

U slučaju baze podataka NIST bazni kasifikator treniran je 35 epoha u tri serije od po 20, 10 i 5 epoha. Svaka sledeća serija koristi veći podskup za aproksimaciju gradijenta (eng. *batch size*). Ovakav trening inspirisan je aktuelnim radovima ove oblasti [35]. Rezultati koje klasifikator postiže jesu 88.36%, 87.39% i 87.35% redom na trening, validacionom i skupu za primenu.

Bazni klasifikator na Deepwriting skupu podataka treniran je 20 epoha nakon čega je obučavanje zaustavljeno tehnikom ranog zaustavljanja. Rezultati ovako natreniranog klasifikatora su 87.65%, 83.12% i 82.12% redom na trening, validacionom i skupu za primenu.

5.6 Stratifikacija i ponovno uzorkovanje

Metod evaluacije uključuje korake stratifikacije podataka i ponovnog uzorkovanja.

Stratifikovana podela napisanih karaktera prilikom primene modela poboljšanja ima veliki značaj. Stratifikacija se vrši u odnosu na odgovarajuće labele, i time se

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 28, 28, 1)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 32)        9248
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 48)        13872
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 48)          20784
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 48)          192
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 48)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 64)          12352
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 1, 1, 64)          36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 1, 1, 64)          256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 768)               49920
_________________________________________________________________
dropout_1 (Dropout)          (None, 768)               0
_________________________________________________________________
next_to_last (Dense)         (None, 256)               196864
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 62)                15934
=================================================================
Total params: 356,798
Trainable params: 356,510
Non-trainable params: 288
_________________________________________________________________
```

Slika 5.8: Arhitektura baznog klasifikatora (NIST skup podataka)

postiže približno jednaka raspodela labela u skupu za prilagođavanje i u test skupu. Ovim se izbegava situacija pojavljivanja karaktera pri evaluaciji, koji se pak ne nalazi u okviru istorije pisanja. Ova situacija se ipak ne može sasvim eliminisati, međutim može se smanjiti njena verovatnoća. Stratifikovana podela slika test skupa izvršena je tako što je u skup za prilagođavanje poslato 90% slika, dok je 10% slika završilo u test skupu.

U okviru evaluacije ponovno uzorkovanje se koristi tako što se za proizvoljnog korisnika preciznost ne računa samo jedanput na test skupu, već se isti postupak ponovi nezavisno n puta[2], svaki put sa stratifikovanom podelom podataka na deo predviđen za istoriju pisanja i deo predviđen za evaluaciju.

5.7 Rezultati evaluacije bez ponovnog uzorkovanja

Rezultati ostvareni bez ponovnog uzorkovanja dobijeni su prilagođavanjem na 90% skupa za primenu i evaluacijom na preostalih 10% koji čine test skup. Prilikom evaluacije se za svakog korisnika test skupa prati preciznost baznog klasifikatora kao i preciznost metoda poboljšanja. Prate se broj/procenat slika na kojima je bazni klasifikator ispravno predvideo labele, broj/procenat slika autora na kojima je metod poboljšanja dao ispravne labele, kao i broj/procenat slika na kome je barem jedan od njih bio ispravan. Ovo poslednje uključeno je sa ciljem ocene kvaliteta poboljšanja, što predstavlja granični slučaj odnosno scenario kako bi se ponašao model poboljšanja kada bi u svakom trenutku znao kom klasifikatoru da veruje, da li baznom ili nekom od metoda k najbližih suseda koji se simultano izvršavaju sa prethodnim. Prethodno predstavlja gornju granicu kvaliteta prezentovanog poboljšanja.

U slučaju baze podataka NIST, informacije o skupu podataka na kome je metod testiran date su sa:

- Model je testiran na 8089 slika pisanih od strane 200 različitih autora.
- Preciznost baznog klasifikatora na njima iznosi 87.09%, dok je preciznost ostvarena poboljšanjem na istom skupu jednaka 89.62%. Gornja granica preciznosti koju je moguće dostići metodom jeste 91.24%.
- Broj autora na kojima se bazni klasifikator ponaša bolje nego prezentovani metod poboljšanja jeste 18, na 63 autora modeli imaju iste performanse, dok na čak 119 njih novi metod poboljšanja daje bolje rezultate.

Na osnovu prethodne statistike jasno se vidi da pri pomenutoj evaluaciji metod poboljšanja daje povećanje preciznosti baznog klasifikatora veličine 2.53%. Teorijski najveće moguće povećanje preciznosti (pri konkretnoj evaluaciji) jeste povećanje od 4.15% (da se za svaku sliku znalo kom klasifikatoru verovati odnosno čije predviđanje uzeti).

[2]gde n predstavlja parametar ponovnog uzorkovanja

U slučaju Deepwriting skupa podataka, važi:

- Broj slika na kojima je model testiran iznosi 3769, i one su napisane od strane 25 različitih autora.
- Preciznost baznog klasifikatora je 81.16%, dok je preciznost koju ostvaruje poboljšani metod 84.05%. Gornja granica preciznosti koju je moguće dostići metodom iznosi 85.43%.
- Na slikama 22 autora metod poboljšanja bolji je od baznog klasifikatora, na slikama jednog autora ostvaruju istu preciznost, dok je na slikama dva autora bazni model bio bolji od u radu prezentovanog metoda.

Pri prethodnoj evaluaciji, prezentovani metod poboljšanja daje povećanje preciznosti baznog klasifikatora od 2.89%. Teorijski najveće moguće povećanje preciznosti jeste 4.27%.

5.8 Rezultati evaluacije sa ponovnim uzorkovanjem

Pri nezavisnim evaluacijama sistem se pokazuje jako dobro, dajući poboljšanja na bazi podataka NIST od oko 2.3%, dok na Deepwriting skupu podataka povećanje preciznosti iznosi preko 2.7% sa jako malim odstupanjima. Prilikom evaluacije broj ponovnih uzorkovanja postavljen je na 10.

Rezultati prilikom evaluiranja modela poboljšanja prethodnim postupkom na bazi podataka NIST dati su sa:

- Model je testiran na *81 685* karaktera, koje je pisalo 200 različitih autora.
- Preciznost baznog klasifikatora je 87.24%, preciznost metoda poboljšanja 89.60%, a gornja granica preciznosti koju je moguće dostići metodom jeste 91.42%.
- Na slikama 24 autora bazni klasifikator ponaša se bolje od metoda poboljšanja, na 9 autora modeli imaju isto ponašanje, dok na čak 167 njih metod poboljšanja ostvaruje bolje rezultate.

Pri izvedenoj evaluaciji metod poboljšanja daje povećanje preciznosti baznog klasifikatora od 2.36%. Teorijski najveće moguće povećanje preciznosti (pri ovoj evaluaciji) jeste 4.18%.

U slučaju Deepwriting skupa podataka, rezultati su sledeći:

- Model je testiran na 37672 karaktera (napisanih od strane 25 korisnika).
- Bazni klasifikator ostvaruje preciznost82.05%, metod poboljšanja 84.77%, dok je gornja granica poboljšanja 86.38%.
- Na slikama 24 autora metod poboljšanja ostvaruje veću preciznost od bazne mreže, dok je na slikama jednog autora bazna mreža ostvarila veću preciznost.

Model poboljšanja povećava preciznost baznog klasifikatora za 2.72%, dok je teorijski najveće moguće povećanje preciznosti jednako 4.33%.

5.9 Poređenje sa najboljim poznatim rezultatima

Prezentovani metod prevazilazi dosadašnje rezultate objavljene na skupu podataka *NIST Special Database 19* pri čemu se koristi transformacija slika u rezoluciju 28×28 koju promoviše *MNIST* odnosno *EMNIST* skup podataka. Važno je napomenuti da ne postoji zvanična podela podataka na trening i test skup u okviru ove baze podataka, kao i to da objavljeni radovi ne objavljuju svoju tačnu podelu podataka. Otud ne postoji jednostavan način za direktno poređenje sa prethodnim radom i stoga se ovde ne tvrdi da je predloženi model bolji od svih dosadašnjih, već se pokazuje da su njegovi rezultati u rangu do sada najboljih. Neki rezultati objavljeni na ovom skupu podataka, dati su u nastavku:

- 2011. godine pomoću ansambla konvolutivnih neuronskih mreža ostvarena preciznost 88.12% [7].
- 2012. godine korišćenjem mreže sa više stubaca procesiranja (eng. *multi-column deep neural network*) postignuta je preciznost 88.37% [8].
- 2017. godine korišćenjem konvolutivne neuronske mreže i metode potpornih vektora ostvaruje se preciznost 88.32% [31].
- Aprila 2018. godine korišćenjem metoda k najbližih suseda i slučajnih šuma (eng. *random forest*) ostvarena je okvirna preciznost od 75% [6].

Performanse prezentovanog metoda poboljšanja postižu preciznost od 89.60% čime su nadmašeni rezultati pomenutih pristupa. U svakom od navedenih radova

prezentovana je drugačija tehnika, međutim ni jedan od njih ne bavi se poboljšanjem klasifikatora oflajn rukom pisanog teksta, niti uzima u obzir stil pisanja korisnika. Dodatno, prema našim saznanjima na skupu podataka NIST ne postoje objavljeni radovi koji se bave poboljšanjem neuronskih klasifikatora oflajn rukom pisanog teksta.

Na Deepwriting skupu podataka trenutno ne postoje javno objavljeni rezultati problema klasifikacije.

Postoje i radovi koji se bave poboljšanjem oflajn klasifikatora rukom pisanog teksta, međutim oni nisu testirani na skupovima podataka NIST odnosno Deepwriting, te stoga nisu uporedivi sa u radu prezentovanim metodom. Dodatno, ni jedan od njih se ne fokusira na stil pisanja pojedinačnog korisnika, što je osnovna ideja kojom je naš metod poboljšanja motivisan.

Glava 6

Zaključak i budući rad

U okviru ovog rada razmatran je problem klasifikacije oflajn rukom pisanih karaktera. Ovaj problem predstavlja jednu aktuelnu i živu oblast, koja svoju primenu nalazi u svakodnevnom životu. Njena upotrebna vrednost čini je jako agilnom u smislu same zajednice i konkretnih inovacija. Neki od problema ove oblasti delimično su rešeni, međutim veći je deo onih u kojima još nisu postignuti željeni rezultati.

Osnovna motivacija ovog rada bila je činjenica da svaki pojedinac neguje svoj stil pisanja, pri čemu se oni među sobom mogu značajno razlikovati, što se pak može naučiti, i na osnovu toga može se povećati preciznost klasičnog klasifikatora. Osnovni doprinos ovog rada predstavlja razvoj jednog novog metoda koji koristi algoritme i tehnike mašinskog učenja kao što su tehnike klasterovanja algoritmom k sredina i klasifikaciju metodom najbližih suseda, a koji nadmašuje performanse vodećih klasifikatora rukom pisanog teksta.

Predloženi model poboljšanja u vreme izvršavanja baznog klasifikatora daje predviđanje kombinujući predviđanje baznog klasifikatora se predviđanjima niza pomoćnih klasifikatora k najbližih suseda koji rade nad novim reprezentacijama slika, izlazima pretposlednjeg sloja neuronskog klasifikatora. Problem odlučivanja kom klasifikatoru verovati, odnosno čije predviđanje uvažiti prevazilazi se modelovanjem stepena poverenja svakoga od njih korišćenjem beta raspodela. Očekivanjem beta raspodele ocenjuje se pouzdanost klasifikatora na predviđenom karakteru, a njeni parametri određuju se korišćenjem slika istorije pisanja pojedinačnog korisnika.

Buduća istraživanja mogu uključivati tehnike transfera učenja (eng. *meta learning*) ili tehnike učenja na malim skupovima podataka (eng. *few-shot learning*) u cilju pripreme baznog klasifikatora za brzo prilagođavanje svakom pojedinačnom korisniku. Dodatno, sistem može biti razvijan i u pravcu prilagođavanja trenut-

nom korisniku koji bi u obzir uzimao smenjivanje raznih autora na istom uređaju (npr. tabletu) a koji bi značaj davao skorijoj istoriji, odnosno poslednjim napisanim karakterima.

Literatura

[1] *Bayesian Inference of a Binomial Proportion - The Analytical Approach*. https://www.quantstart.com/articles/Bayesian-Inference-of-a-Binomial-Proportion-The-Analytical-Approach. Accessed: 2019-07-16.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] G. Bradski. „The OpenCV Library". U: *Dr. Dobb's Journal of Software Tools* (2000).

[4] Nikhil Buduma i Nicholas Locascio. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. 1st. O'Reilly Media, Inc., 2017. ISBN: 1491925612, 9781491925614.

[5] François Chollet i dr. *Keras*. https://github.com/fchollet/keras. 2015.

[6] Nicole Cilia i dr. „A ranking-based feature selection approach for handwritten character recognition". U: *Pattern Recognition Letters* (Apr. 2018). DOI: 10.1016/j.patrec.2018.04.007.

[7] D. C. Ciresan i dr. „Convolutional Neural Network Committees for Handwritten Character Classification". U: *2011 International Conference on Document Analysis and Recognition*. Sept. 2011, str. 1135–1139. DOI: 10.1109/ICDAR.2011.229.

[8] Dan Cireşan, Ueli Meier i Juergen Schmidhuber. „Multi-column Deep Neural Networks for Image Classification". U: (2012). eprint: arXiv:1202.2745.

[9] Gregory Cohen i dr. *EMNIST: an extension of MNIST to handwritten letters*. 2017. eprint: arXiv:1702.05373.

[10] Milan Čugurović. *DoubledMNIST Database*. https://github.com/MilanCugur/DoubledMNIST. 2019.

[11] Nibaran Das i dr. „A Benchmark Image Database of Isolated Bangla Handwritten Compound Characters". U: *Int. J. Doc. Anal. Recognit.* 17.4 (Dec. 2014), str. 413–431. ISSN: 1433-2833. DOI: 10.1007/s10032-014-0222-y. URL: http://dx.doi.org/10.1007/s10032-014-0222-y.

[12] D. Đorić i dr. *Atlas raspodela*. Građevinski fakultet, 2007. ISBN: 9788675180777. URL: https://books.google.rs/books?id=er9ZGQAACAAJ.

[13] Stefan Fiel i Robert Sablatnig. „Writer Identification and Retrieval Using a Convolutional Neural Network". U: *Computer Analysis of Images and Patterns*. Ur. George Azzopardi i Nicolai Petkov. Cham: Springer International Publishing, 2015, str. 26–37. ISBN: 978-3-319-23117-4.

[14] Brian Fisher i Adem Kilicman. „Some Results on the Gamma Function for Negative Integers". U: *Appl. Math. Inform. Sci.* 6 (Maj 2012), str. 173–176.

[15] Leon A. Gatys, Alexander S. Ecker i Matthias Bethge. *A Neural Algorithm of Artistic Style*. 2015. eprint: arXiv:1508.06576.

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

[17] Patrick Grother i Kayee Hanaoka. *NIST Special Database 19 Handprinted Forms and Characters 2nd Edition*. 2016.

[18] J. D. Hunter. „Matplotlib: A 2D graphics environment". U: *Computing in Science & Engineering* 9.3 (2007), str. 90–95. DOI: 10.1109/MCSE.2007.55.

[19] Yangqing Jia i dr. *Caffe: Convolutional Architecture for Fast Feature Embedding*. 2014. eprint: arXiv:1408.5093.

[20] Justin Johnson, Alexandre Alahi i Li Fei-Fei. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. 2016. eprint: arXiv:1603.08155.

[21] Eric Jones, Travis Oliphant, Pearu Peterson i dr. *SciPy: Open source scientific tools for Python*. Accessed: 2019-07-16. 2001–. URL: http://www.scipy.org/.

[22] Diederik P. Kingma i Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. eprint: arXiv:1412.6980.

[23] F. Kleber i dr. „CVL-DataBase: An Off-Line Database for Writer Retrieval, Writer Identification and Word Spotting". U: *2013 12th International Conference on Document Analysis and Recognition*. Avg. 2013, str. 560–564. DOI: 10.1109/ICDAR.2013.117.

[24] Yann LeCun i Corinna Cortes. „MNIST handwritten digit database". U: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[25] C. G. Leedham. „Historical perspectives of handwriting recognition systems". U: *IEE Colloquium on Handwriting and Pen-Based Input*. Mar. 1994, str. 1/1–1/3.

[26] Urs-Viktor Marti i Horst Bunke. „The IAM-database: an English sentence database for offline handwriting recognition". U: *International Journal on Document Analysis and Recognition* 5 (2002), str. 39–46.

[27] Dino Škrobar Mihaela Ribičić Penava. „Gama i beta funkcije". U: *Osječki matematički list* 15 (2015), str. 93–111.

[28] Mladen Nikolić i Anđelka Zečević. *Mašinsko učenje*. Matematički fakultet, Univerzitet u Beogradu, 2019. URL: http://ml.matf.bg.ac.rs/readings/ml.pdf.

[29] Mladen Nikolić i Anđelka Zečević. *Naučno izračunavanje*. Matematički fakultet, Univerzitet u Beogradu, 2019. URL: http://ni.matf.bg.ac.rs/materijali/ni.pdf.

[30] Travis Oliphant. *NumPy: A guide to NumPy*. USA: Trelgol Publishing. [Online; accessed 2019-07-16]. 2006–. URL: http://www.numpy.org/.

[31] Darmatasia Palehai i Mohamad Ivan Fanany. „Handwriting Recognition on Form Document Using Convolutional Neural Network and Support Vector Machines (CNN-SVM)". U: Maj 2017. DOI: 10.1109/ICoICT.2017.8074699.

[32] F. Pedregosa i dr. „Scikit-learn: Machine Learning in Python". U: *Journal of Machine Learning Research* 12 (2011), str. 2825–2830.

[33] Boris T. Polyak. „Some methods of speeding up the convergence of iteration methods". U: *USSR Computational Mathematics and Mathematical Physics* (1964), str. 1–17.

[34] Guido Rossum. *Python Reference Manual*. Tehn. izv. Amsterdam, The Netherlands, The Netherlands, 1995.

[35] Samuel L. Smith i dr. *Don't Decay the Learning Rate, Increase the Batch Size*. 2017. eprint: arXiv:1711.00489.

[36] Jost Tobias Springenberg i dr. *Striving for Simplicity: The All Convolutional Net*. 2014. eprint: arXiv:1412.6806.

[37] Garside Rodger Stig Johhanson Atwel Eric i Leech Geoffrey. *The tagged LOB Corpus: User's manual*. 1986.

[38] Pang-Ning Tan, Michael Steinbach i Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005. ISBN: 0-321-32136-7.

Biografija autora

Milan Miloje Čugurović rođen je 30.08.1995. godine u Loznici.

Osnovnu školu „14 Oktobar" u Dragincu, kao i prirodno-matematički smer gimnazije „Vuk Karadžić" u Loznici završio je kao nosilac Vukove diplome. Tokom navedenog perioda školovanja isticao se u oblasti matematike. To potvrđuje i veći broj nagrada na Državnim takmičenjima.

Smer Računarstvo i informatika upisuje na Matematičkom fakultetu 2014. godine. Na navedenom smeru diplomirao je 2.7.2018. godine, posle tri godine i devet meseci studiranja, sa prosečnom ocenom 10.

Tokom školovanja dobitnik je velikog broja nagrada i stipendija, od kojih se mogu izdvojiti nagrada za najboljeg studenta generacije Matematičkog fakulteta (2019), nagrada „Nedeljko Parezanović" Katedre za Računarstvo Matematičkog fakulteta (2018), stipendija „Dositeja" Fonda za mlade talente Republike Srbije (2018, 2019), stipendija Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije (2016, 2017, 2018), Nagrada Matematičkog fakulteta za najbolje studente (2017, 2018), kao i Nagrada grada Loznice za najbolje studente (2016, 2017, 2018).

U toku akademske 2018/19 angažovan je kao saradnik u nastavi na Katedri za Računarstvo Matematičkog fakluteta, Univerziteta u Beogradu. Drzi vežbe iz predmeta Relacione baze podataka, Uvod u relacione baze podataka, Programske paradigme, Programiranje 1. U toku letnjeg semestra akademske 2017/18 u svojstvu demonstratora držao je vežbe predmeta Programiranje 2, na Matematičkom faklutetu. Pored toga, tokom treće godie studija jedan mesec proveo je na praksi u industrijskom sektoru (rad na razvoju serverske strane aplikacije u kompaniji *TeleTrader*).

Kao slušalac posetio je dve domaće konferencije, konferenciju *US-Serbia and West Balkan Data Science Workshop*, održanu u avgustu 2018. godine u Beogradu, kao i *Data Science Conference V4.0* koja je održana u septembru 2018. godine, takođe u Beogradu. U okviru iste konferencije, održao je uvodni kurs mašinskog učenja u okviru predkonferencijskih kurseva.

Trenutne oblasti interesovanja uključuju pre svega oblast mašinskog učenja, kao i oblasti istraživanja podataka odnosno nauke o podacima u užem smislu. Takođe, oblast interesovanja uključuje i relacione baze podataka.

U maju 2012. godine kao najbolji mladi matematičar zatvorio je svojim govorom 41. Đački Vukov Sabor u Tršiću, u okviru manifestacije *Maj mesec matematike*.