Univerzitet u Beogradu
Matematički fakultet

Master rad

Primena neuronskih mreža na predviđanje funkcije enzima određivanjem njihovog EC broja

Autor:
Nevena Ćirić

Mentor:
dr Jovana Kovačević

Članovi komisije:

dr Jovana Kovačević

dr Mladen Nikolić

prof. dr Gordana Pavlović-Lažetić

Beograd, 2020.

Univerzitet u Beogradu
Matematički fakultet
Katedra za računarstvo i informatiku

Sažetak

Enzimi su vrsta biološki aktivnih proteina koji katalizuju hemijske reakcije neophodne za odvijanje metaboličkih funkcija u ćelijama. Poznavanje funkcija enzima je stoga esencijalno u istraživanju biohemijskih procesa u organizmima. Anotacija funkcije enzima ima širok spektar primena, kao što su metagenomika, industrijska biotehnologija i dijagnoza bolesti izazvanih nedostatkom ili funkcionalnim mutacijama enzima. S obzirom da eksperimentalno određivanje funkcije enzima zahteva dosta vremena i skupe resurse, a broj novosekvencioniranih enzima eksponencijalno raste, pribegava se računarskim metodama za određivanje funkcije enzima.

EC broj (*Enzyme Commision number*) je numerička klasifikaciona šema za enzime koja pomoću 4 broja opisuje funkcije enzima na osnovu njihove strukturne uloge - uloge katalizatora hemijskih reakcija. U ovom sistemu enzimske nomenklature svaki EC broj odgovara familiji enzima specifične strukture, pri čemu su te familije hijerarhijski organizovane na 4 nivoa. Enzimi iz iste (pod)familije katalizuju isti (pod)tip hemijskih reakcija.

U ovom radu izgrađen je i implementiran model mašinskog učenja koji predviđa funkciju enzima određivanjem njegovog EC broja na osnovu sekvence i funkcionalnih domena enzima. Sekvence enzima zajedno sa njihovim EC brojevima na kojima će model biti obučavan i testiran biće preuzeti iz UniProtKB baze proteina, a funkcionalni domeni dobijeni pomoću programa HMMER na osnovu sekvence enzima. U radu je korišćen programski jezik Python sa svojim bibliotekama za mašinsko učenje, numerička izračunavanja i slično.

Sadržaj

Spisak slika

Glava 1

1 Uvod

Funkcionalna anotacija enzima predstavlja jedan od najznačajnijih problema u oblasti bioinformatike. Precizna detekcija funkcionalnog ponašanja enzima je važan korak u personalnoj dijagnostici i razvijanju lekova za bolesti izazvanih nedostatkom ili funkcionalnim mutacijama enzima. Takođe, kod bolesti izazvanih patogenim mikroorganizmima blokiranje enzimske aktivnosti može da uzrokuje prestanak rada patogena, pa su osnov mnogih lekova inhibitori enzima sa određenim funkcijama. Aktuelne eksperimentalne metode za funkcionalnu anotaciju enzima zahtevaju dosta vremena i skupe resurse, i ne mogu da isprate priliv novosekvencioniranih enzima čiji broj raste sa svakim sekvencioniranim genomom. Zbog toga se radi na razvijanju i unapređenju računarskih metoda za određivanje funkcije enzima.

Problemu određivanja funkcije enzima se može pristupiti kao problemu hijerarhijske klasifikacije. EC broj (*Enzyme Commision number*) je numerička klasifikaciona šema za enzime koja pomoću 4 broja opisuje funkcije enzima na osnovu njihove strukturne uloge - uloge katalizatora hemijskih reakcija. U ovom sistemu enzimske nomenklature svaki EC broj odgovara familiji enzima specifične strukture, pri čemu su te familije hijerarhijski organizovane na 4 nivoa. Enzimi iz iste (pod)familije katalizuju isti (pod)tip hemijskih reakcija.

Predviđanje funkcije enzima se može vršiti na osnovu hemijskih, fizičkih i strukturnih svojstva enzima. Struktura proteina zavisi od rasporeda aminokiselina i direktno utiče na njegovu funkciju. U zavisnosti sa kog nivoa se posmatra struktura, to može biti primarna, sekundarna, tercijarna ili kvaternerna struktura enzima. Pri tome, sve navedene osobine enzima se mogu posmatrati lokalno ili globalno.

Klasičan pristup problemu predikcije funkcije enzima podrazumeva izbor nekog podskupa navedenih svojstava enzima na osnovu kojih će se vršiti takozvano prenošenje funkcionalne anotacije sa sličnih enzima, za koje su funkcije eksperimentalno utvrđene, na enzim kome treba pridružiti funkciju. Napredniji pristup se sastoji u primeni neuronskih mreža - modela mašinskog učenja koji je u stanju da sam konstruiše nove atribute na osnovu odabranog skupa podataka, koji mogu biti informativniji za predviđanje funkcije enzima.

U ovom radu prikazan je pristup koji kombinuje dve vrste lokalnih strukturnih svojstva enzima i dve vrste neuronskih mreža koje na osnovu njih konstruišu nove atribute i vrše predikciju funkciju enzima pridruživanjem odgovarajućeg EC broja. Korišćene su sekvence enzima, kao lokalne informacije o strukturi enzima, i funkcionalni domeni (delovi sekvenci koji imaju određenu funkciju) kao lokalne informacije o funkciji enzima. Sekvence enzima zajedno sa njihovim EC brojevima na kojima je model obučavan i testiran su preuzete iz UniProtKB baze proteina, a funkcionalni domeni dobijeni pomoću programa HMMER na osnovu sekvence enzima. Model je razvijan u programskom jeziku *Python* sa njegovim bibliotekama za mašinsko

učenje, numerička izračunavanja i slično.

Motivaciju za ovo istraživanje pružio je rad [1], u daljem tekstu referentni rad. Iz njega su preuzete osnovne ideje za izbor skupa podataka i strukturu modela. Kako skup podataka sa kojim se radilo nije u potpunosti isti kao u referentnom radu, poređenje rezultata nije bilo moguće. Takođe, rezultati u referentnom radu nisu dati tabelarno, već samo grafički, pa stoga ne bi ni bilo moguće precizno poređenje sa rezultatima dobijenim u ovom radu. Vizuelnim poređenjem je utvrđeno da su rezultati sličnih redova veličine.

U poglavlju 2 uvedeni su biološki pojmovi neophodni za razumevanje rada, dok su u okviru poglavlja 3 predstavljene metode i tehnike mašinskog učenja koje su korišćene u radu. Zatim je u poglavlju 4 prikazan način predstavljanja bioloških podataka u računaru zajedno sa tehnikama za efikasno čuvanje i baratanje velikom količinom podataka. U poglavlju 5 su opisani detalji izgradnje modela i implementacije. U poglavlju 6 sumirani su rezultati koje model daje na izdvojenom skupu enzima za evaluaciju. Na kraju, u poslednjem poglavlju dat je kratak osvrt na celokupan rad, kao i planovi za dalja unapređenja.

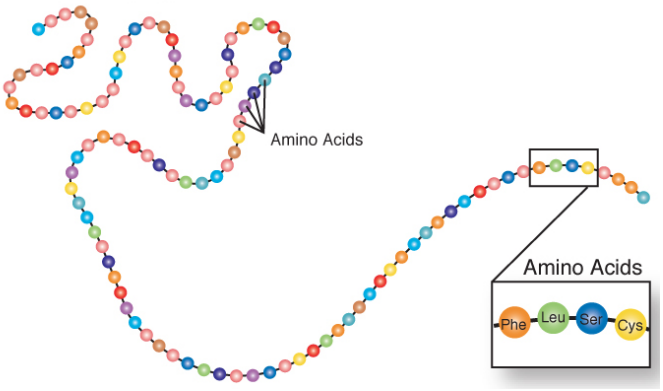
Glava 2

2 Predviđanje funkcije enzima

Enzimi su vrsta biološki aktivnih proteina koji omogućavaju nastanak i ubrzavaju i olakšavaju tok velikog broja hemijskih reakcija u ćelijskoj i vanćelijskoj sredini. Skoro svim metaboličkim procesima u ćeliji neophodan je odgovarajući enzim kako bi se odvijali dovoljno brzo da bi se održao život. Funkcionalna mutacija, nedovoljna ili uvećana proizvodnja jednog jedinog enzima može da ima kao posledicu teške metaboličke poremećaje. Na primer, nedostatak enzima fenilalnin hidroksilaze, koji je neophodan za sintezu tirozina - neurotransmitera koji podstiče moždanu aktivnosti, dovodi do moždanog oštećenja i progresivne mentalne retardecije.

2.1 Struktura enzima

Po hemijskom sastavu enzimi predstavljaju linearne lance aminokiselina koje su povezane peptidnim vezama (slika 2.1). Aminokiseline se sastoje iz amino-grupe $-NH_2$, karboksilne grupe $-COOH$ i bočnog lanca koji je jedinstven za svaku aminokiselinu. Zahvaljujući svom hemijskom sastavu, aminokiseline mogu da se nadovezuju u lance formiranjem peptidnih veza, gde karboksilna grupa jedne aminokiseline reaguje sa amino grupom druge aminokiseline. Lanci aminokiselina enzima mogu biti dužine od nekoliko desetina do nekoliko hiljada. Dužina lanca ne utiče na funkciju enzima, već isključivo redosled povezivanja aminokiselina.



Slika 2.1: Sekvenca enzima [3]

U prirodi postoji veliki broj aminokiselina, ali samo 20 učestvuje u izgradnji enzima (i opštije, proteina) i one se nazivaju *osnovnim aminokiselinama*. Spisak osnovnih aminokiselina sa njihovim skraćenicama koje se koriste u predstavljanju sekvenci enzima prikazane su na slici 2.2.

Amino Acid	Three Letter Code	One Letter Code
Alanine	Ala	A
Arginine	Arg	R
Aspartic Acid	Asp	D
Asparagine	Asn	N
Cysteine	Cys	C
Glutamic Acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V

Slika 2.2: Osnovne aminokiseline [4]

Broj, vrsta i redosled aminokiselina jednog enzima predstavlja njegovu *primarnu strukturu*. Ona je primarni izvor informacija o enzimu i njegovoj funkciji. Pored primarne, lanci aminokiselina enzima imaju i različite prostorne strukture. Da bi bili u stanju da obavljaju svoju biološku funkciju, enzimi se savijaju u jednu ili više specifičnih prostornih konformacija.[1] U zavisnosti od dimenzije prostora u kojoj se posmatra konformacija enzima, može se govoriti o *sekundarnoj*, *tercijarnoj* i *kvaternernoj strukturi*. Slika 2.3 ilustruje različite nivoe strukture proteina:

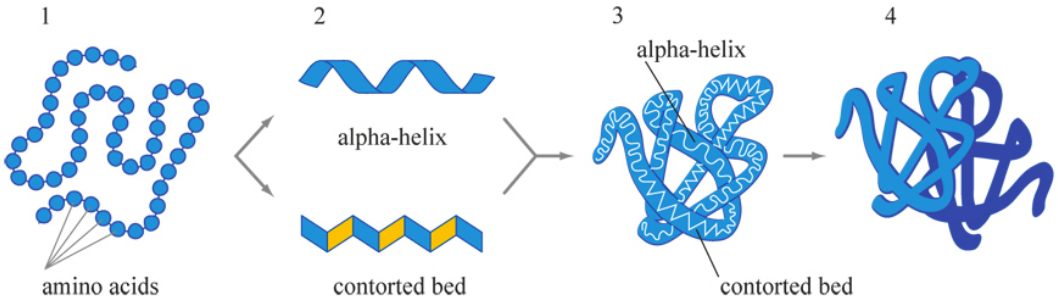
* **Primarna struktura** predstavlja redosled povezivanja aminokiselina peptidnom vezom kako bi formirale enzim. Primarna struktura enzima je određena genom na osnovu kog se vrsi sinteza tog enzima. Poznato je da proteini sa sličnim primarnim strukturama teže da obavljaju iste funkcije. Mutacije na primarnoj strukturi enzima dovode do nepravilnog savijanja enzima, odnosno do mutacije na sekundarnoj, tercijarnoj i kvaternernoj strukturi.

* **Sekundarna struktura** je lokalna prostorna organizacija molekula lanca. Opisuje odnos i prostorni raspored susednih aminokiselina u lancu. Najčešće sekundarne strukture su α heliksi i β ravni. Ona, ipak, ne opisuje specifične položaje atoma u 3D prostoru, već njih opisuje tercijarna struktura.

* **Tercijarna struktura** opisuje globalnu konformaciju enzima, odnosno trodimenzioni raspored molekula u jednom lancu aminokiselina. Tercijarna struktura je zavisna i od primarne i od sekundarne strukture, jer je definisana interakcijama između aminokiselina koje mogu biti veoma udaljene u lancu, ali se usled savijanja lanca one nađu jedna blizu druge. Pod tercijarnom strukturom podrazumevamo prostorne koordinate svih molekula lanca. Često, slične sekvence enzima imaju i slične trodimenzione strukture.

[1] Prostorni oblici molekula se nazivaju konformacije.

* Mnogi enzimi su sastavljeni od više lanaca aminokiselina, takozvanih podjedinica enzima koje čine enzimski kompleks. Podjedinice mogu biti međusobno različite ili potpuno iste. **Kvaternarna struktura** proteina podrazumeva trodimenzionalni oblik podjedinica enzimskog kompleksa, odnosno način na koji su podjedinice smeštene unutar kompleksa. Postoje enzimi čija kompleksnost nije dovoljno velika da bi sadržali i kvartenarnu strukturu, što znači da ona nije definisana za sve enzime.



Slika 2.3: Četiri nivoa strukture proteina [5]

2.2 Funkcionalni domeni

Funkcionalni domen enzima je stabilna, fundamentalna jedinica tercijarne strukture enzima koja može da ostvaruje svoju funkciju nezavisno od ostatka enzimskog lanca. Funkcionalni domen je deo sekvence enzima i strukture koji može autonomno da se savija[2], funkcioniše i evoluira. Mnogi enzimi se sastoje od nekoliko funkcionalnih domena i oni se nazivaju multifunkcionalnim domenima. U multidomenskom proteinu svaki domen može da izvršava svoju funkciju nezavisno ili na organizovan način sa svojim susedima. Takođe, jedan domen se može javiti u više različitih enzima.

Molekulska evolucija koristi domene kao strukturne komponente koje mogu da se kombinuju na različite načine da bi se formirali enzimi sa različitim funkcijama. Domeni variraju u dužini od oko 25 aminokiselina do 500 aminokiselina. Funkcionalni domeni mogu sadržati prekide (biti diskontinualni), što znači da je više od jednog segmenta lanca neophodno da bi se formirao domen.[3] Pretpostavlja se da je to posledica umetanja jednog domena u drugi tokom evolucije. Približno jedna četvrtina poznatih funkcionalnih domena je diskontinualno.

Međutim, poznavanje funkcionalnih domena proteina ne daje dovoljnu informaciju o njegovoj funkciji. Mnogi funkcionalni domeni su pronađeni kod enzima iz različitih familija enzima koji katalizuju potpuno različite hemijske reakcije.[4]

[2]Formira kompaktnu trodimenzionalnu strukturu

[3]Iz ovog razloga su pored sekvenci enzima uzeti i funkcionalni domeni za atribute nad kojima će model biti obučavan - rekurentni sloj neuronske mreže ne bi mogao da dovede u vezu segmente lanca koji su međusobno na velikom rastojanju.

[4]Ovo je razlog zašto za atribute nad kojima će model biti obučavan nisu uzeti samo funkcionalni domeni, već kao dopuna informacija koje ne mogu biti dobijene samo na osnovu sekvenci.

2.3 Klasifikacija enzima

Na osnovu podataka iz UniProtKB baze podataka, od 563 082 eksperimentalno anotiranih proteina njih 274 249 su enzimi. Tako veliki broj enzima bilo je potrebno na neki način klasifikovati, pa je po preporuci Komisije za enzimologiju 1961. godine na Internacionalnom kongresu biohemije u Briselu definisan sistem enzimske nomenklature.

EC broj (*Enzyme Commision number*) je numerička klasifikaciona šema za enzime koja pomoću 4 broja opisuje funkcije enzima na osnovu njihove strukturne uloge - uloge katalizatora hemijskih reakcija. U ovom sistemu enzimske nomenklature svaki EC broj odgovara familiji enzima specifične strukture, pri čemu su te familije hijerarhijski organizovane na 4 nivoa. Enzimi iz iste (pod)familije katalizuju isti (pod)tip hemijskih reakcija.

Ovaj klasifikacioni sistem ima strukturu drveta (slika 2.4). Najpre se familija enzima deli u klase prema tipu katalisane reakcije. Klase enzima se dele na podklase prema vrsti hemijske veze koja se transformise. Podklase enzima su dalje podeljene u grupe prema tipu supstrata[5] koji učestvuje u reakciji. Prema tome, klasifikacioni broj enzima je sastavljen iz 4 cifre, gde:

– prva cifra označava klasu enzima
– druga cifra označava podklasu
– treća cifra označava grupu u podklasi
– četvrta cifra je serijski broj enzima u okviru grupe

Prema vrsti reakcija na koje deluju enzimi su grupisani u 7 osnovnih klasa:[6]

1. Oksidoreduktaze
2. Transferaze
3. Hidrolaze
4. Liaze
5. Izomeraze
6. Ligaze
7. Translokaze

Oksidoreduktaze U ovu klasu se ubrajaju enzimi koji katalizuju procese oksidacije i redukcije u ćelijama. Oksidoredukcioni procesi koji su katalisani ovim enzimima zasnivaju se na transferu elektrona sa jednog molekula na drugi. Reakcija oksidoredukcije sastoji se iz: reakcije oksidacije koja predstavlja proces otpuštanja elektrona sa nekog molekula i reakcije redukcije koja predstavlja proces primanja elektrona od strane nekog molekula. [7]

[5]Supstrat je molekul na koji deluje enzim. Enzimi katalizuju hemijske reakcije nad supstratima.

[6]Prvih 6 klasa enzima je prepoznato još 1961. u prvoj verziji sistema enzimske nomenklature, dok je 7. klasa izdvojena u avgustu 2018.

[7]Može biti zbunjujuće da proces redukcije podrazumeva primanje (povećanje broja) elektrona, međutim termin 'redukcija' se u ovom kontekstu odnosi na promenu oksidacionog broja. Za molekul koji otpušta elektrone kaže se da je oksidovao - povećao svoj oksidacioni broj, dok se molekul koji prima elektrone redukovao - smanjio svoj oksidacioni broj.

Slika 2.4: Hijerarhijska klasifikacija enzima [7]

Transferaze Enzimi ove klase katalizuju transfer funkcionalnih grupa između molekula. Transferaze omogućavaju prenošenje grupe atoma (delova molekula) u organskim jedinjenjima koje daju karakterističnе osobine tim jedinjenjima i njihovim reakcijama sa drugim jedinjenjima.

Hidrolaze Hidrolaze su enzimi koji katalizuju hidrolitičko razlaganje supstrata uz učešće molekula vode. Ovi enzimi mogu da razlažu složena organska jedinjenja na manje fragmente pod uticajem kontakta sa molekulima vode. Proces hidrolitičkog

razlaganja se može odvijati u intracelularnim i ekstracelularnim uslovima. U ovu grupu enzima ubrajaju se svi probavni enzimi koji katalizuju razlaganje složenih molekula do jednostavnih jedinjenja. Na primer, enzim lipaza doprinosi razgradnji masti, lipoproteina i drugih većih molekula na manje molekule poput masnih kiselina i glicerola. Masne kiseline i drugi mali molekuli koriste se za sintezu i kao izvor energije.

Liaze i ligaze Liaze katalizuju razlaganje supstrata, ali na način koji se razlikuje od oksidacije i hidrolize, često formirajući novu dvostruku vezu ili nov prsten. Takođe, omogućavaju pripajanje neke hemijske grupe na dvostruku vezu supstrata. Ligaze su enzimi koji mogu da katalizuju spajanje dva velika molekula putem formiranja nove hemijske veze, uz utrošak energije. Najznačajniji enzimi ovih klasa katalizuju procese karboksilacije i dekarboksilacije karboksilnih kiselina.[8] Procesima dekarboksilacije i karboksilacije se obezbeđuje kruženje ugljen-dioksida u prirodi.

Izomeraze Izomeraze su enzimi koji katalizuju intramolekulska preuređivanja, odnosno stvaranje izomera, jedinjenja koja imaju isti hemijski sastav i molekulsku masu, a razlikuju se po položaju hemijskih grupa i zbog toga imaju različite fizičke i hemijske osobine.

Translokaze Prethodnih 6 klasa enzima je identifikovano pri definisanju prve verzije klasifikacione šeme. Međutim, kasnije je postalo očigledno da nijedna od tih klasa ne može da okarakteriše veoma važnu klasu enzima koji katalizuju prenos molekula kroz membranu ili njihovo odvajanje unutar membrana. To su većinom enzimi koji su se prethodno svrstavali u klasu hidrolaza, i oni zaista katalizuju hidrolitičke reakcije ali to im nije primarna funkcija, već premeštanje molekula kroz i unutar membrane.

Svaka od ovih klasa se da dalje deli na podklase prema vrsti hemijske veze koja se transformiše. Ukupan broj podklasa na drugom nivou hijerarhije je 74, stoga neće biti pojedinačno nabrajane.[9] Nazivi nekih od podklasa su prikazani na slici 2.4.

[8] Karboksilne kiseline su organska jedinjenja koja u sebi sadrže karboksilnu grupu ($-COOH$). Opšta formula karboksilnih kiselina je $R-COOH$, gde R predstavlja alkil grupu C_nH_{2n}. Proces dekarboksilacije je hemijska reakcija kojom se uklanja karboksilna grupa ($-COOH$) i oslobađa ugljen-dioksid (CO_2). Karboksilacija je suprotna reakcija u kojoj se uvodi karboksilna grupa u supstrat i sintetise karboksilna kiselina.

[9] Spisak svih podklasa može se pronaći na adresi `https://en.wikipedia.org/wiki/List_of_enzymes`

Glava 3

3 Neuronske mreže

Neuronske mreže predstavljaju trenutno najpopularniju i jednu od najprimenjenijih metoda mašinskog učenja. Njihove primene su dale snažan doprinos mnogim bioinformatičkim problemima, kao što su genetska analiza, predviđanje specifičnosti vezivanja sekvenci, obrada slika raznih dijagnostičkih merenja kao što su magnetna rezonanca, skener i slično. Ono što ih izdvaja od ostalih metoda mašinskog učenja jeste mogućnost konstruisanja novih atributa na osnovu odabranog skupa podataka (postojećih atributa ili sirove reprezentacije podataka). Naime, iako domenski eksperti mogu odrediti koji su atributi najinformativniji za konkretan problem, njihovi izbori mogu biti pogrešni, a neretko lošiji od onoga što bi algoritam učenja mogao da detektuje. Taj proces se naziva *ekstrakcijom atributa* i smatra se da je to jedan od najbitnijih razloga za delotvornost neuronskih mreža.

Postoje različite vrste neuronskih mreža, često specijalizovanih za konkretne primene. Ono što im je zajedničko je da se sastoje od određenog broja elementarnih jedinica – neurona, koji nalik biološkim neuronima u mozgu, jedni drugima prosleđuju signale i izračunavaju nove signale na osnovu onih koji su im prosleđeni. Tačna struktura povezanosti neurona i način na koji oni vrše izračunavanja definiše o kakvoj mreži se radi. Izbori između različitih strukura i načina izračunavanja nazivaju se arhitekturom mreže. U nastavku će najpre biti opisana struktura pojedinačnih neurona, a potom kako se oni organizuju u dve različite arhitekture - *potpuno povezanu* i *rekurentnu*.

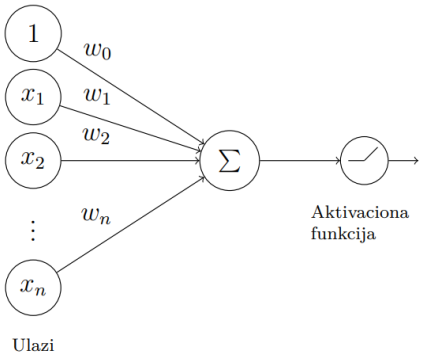
3.1 Neuron kao osnovna gradivna jedinica modela

Neuroni su osnovne jedinice izračunavanja čijim povezivanjem se gradi složenije izračunavanje, odnosno neuronska mreža. To su jednostavne parametrizovane funkcije, pa se obučavanje neuronske mreže svodi upravo na određivanje parametara neurona tako da greška predviđanja u odnosu na stvarnu vrednost bude što manja.

Svaki neuron računa linearnu kombinaciju svojih argumenata (ulaznih signala) nad kojom se zatim primenjuje neka nelinearna transformacija, takozvana *aktivaciona funkcija*. Formalno, ako neuron kao ulaz uzima n signala x_1, x_2, ..., x_n, on vrši sledeće izračunavanje:

$$g\left(w_0 + \sum_{i=1}^{n} w_i x_i\right) \tag{3.1}$$

gde su w_0, w_1, ..., w_n parametri ili težine neurona kojima se vrši linearno kombinovanje ulaza, a g nelinearna aktivaciona funkcija kojom se ta linearna kombinacija transformiše. Grafički prikaz neurona dat je slikom 3.5.



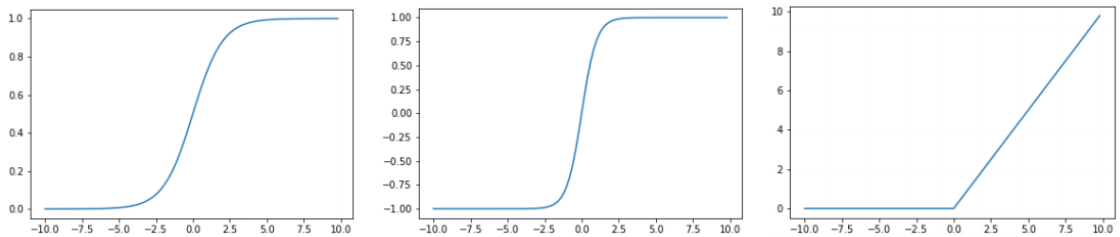
Slika 3.5: Struktura neurona

Za aktivacionu funkciju g se uvek bira nelinearna funkcija jer bi u suprotnom celokupna transformacija koju neuron vrši, a samim tim i cela neuronska mreža, bila linearna funkcija, pa ne bi bilo moguće ovim modelom aproksimirati nelinearne funkcije dovoljno dobro. Vrednost w_0 naziva se slobodnim članom. Nekada se vektor ulaza x proširuje tako da bude oblika $x = (1, x_1, ..., x_n)$ kako bi izraz (3.1) imao kraći zapis $f_w(x) = g(w \cdot x)$, gde $\cdot$ označava skalarni proizvod.

Aktivaciona funkcija nije jednoznačno definisana i postoji više mogućih izbora. Najčešće su monotone, neprekidne i skoro svuda diferencijabilne. Zahvaljujući monotonosti, što je vrednost linearne kombinacije (na koju deluje aktivaciona funkcija) veća, to je izlazni signal neurona jači. Visoke težine pridružene ulazima vode jakom uticaju tih ulaza na izlazni signal, a vrednosti bliske nuli eliminišu uticaj tih ulaza. Znak pridružene težine definiše da li neki ulaz podstiče ili sprečava ispoljavanje izlaznog signala. Zatim, kako se optimizacija[10] najčešće vrši metodima koji koriste gradijent funkcije, potrebno je da aktivaciona funkcija bude neprekidna i skoro svuda diferencijabilna. Neke od često korišćenih aktivacionih funkcija su:

- sigmoidna funkcija $\sigma(x) = \frac{1}{1+e^{-x}}$
- tangens hiperbolički $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- ispravljena linearna jedinica (eng. *rectified linear unit*) $rlu(x) = max(0, x)$

Grafici sve tri funkcije su prikazani na slici 3.6.



Slika 3.6: Sigmoidna funkcija, tangens hiperbolički i ispravljena linearna jedinica

[10]Postupak minimizacije greške predviđanja modela u odnosu na stvarnu vrednost.

3.2 Opšta formulacija klasifikacionog modela

Gradivni elementi ovakvog modela - neuroni, organizuju se u slojeve koji se nadovezuju i time formiraju neuronsku mrežu. Organizacija neurona u okviru slojeva zajedno sa načinom njihovog međusobnog povezivanja definiše arhitekturu mreže.

Prvi sloj mreže naziva se ulaznim slojem, dok se poslednji sloj naziva izlaznim slojem. Neuroni prvog sloja kao argumente primaju ulaze mreže dok neuroni svakog od preostalih slojeva kao svoje ulaze prihvataju (neke) izlaze prethodnog sloja. Svi slojevi koji svoje izlaze prosleđuju narednom sloju nazivaju se skrivenim slojevima.[11] Broj slojeva mreže određuje njenu dubinu.

Ukoliko mreža ima L slojeva, ona se može formalno definisati na sledeći način:

$$\begin{aligned} h_0 &= x \\ h_i &= g_i(W_i h_{i-1} + w_{i0}) \\ f_w(x) &= g'(h_L) \end{aligned}$$

gde h_i za $i = 1, ..., L$ predstavlja vektor izlaza i-tog sloja neurona, W_i je matrica parametara i-tog sloja čije vrste predstavljaju vektore parametara pojedinačnih neurona tog sloja koji množe ulazne vrednosti, w_{i0} vektor slobodnih koeficijenata tih neurona, a W predstavlja skup svih tih parametara W_i i w_{i0}. Pod aktivacionom funkcijom g_i podrazumeva se vektorska funkcija određena skalarnim aktivacionim funkcijama koje deluju na svaku koordinatu vektora pojedinačno, pri čemu se one mogu razlikovati.

Funkcija g' može predstavljati drugačiju funkciju od aktivacionih funkcija g_i, što je uobičajeno za poslednji sloj mreže. Ovaj poslednji izbor zavisi od toga u koju svrhu se mreža koristi. Neuronske mreže se koriste kod problema *regresije*, određivanja neprekidne funkcije koja opisuje vezu između ulaza i izlaza, i problema *klasifikacije*, svrstavanja ulaznih vektora u jednu od konačno mnogo kategorija. U slučaju regresije, neuroni poslednjeg sloja ne koriste aktivacionu funkciju, odnosno $g' = id$.[12] Kod klasifikacije se podrazumeva da poslednji sloj mreže ima onoliko neurona koliko ima klasa, a za funkciju g' se koristi takozvana funkcija *mekog maksimuma* (eng. *softmax*) koja preslikava vektor dimenzije C (broj neurona na poslednjem sloju) u vektor iste dimenzije na sledeći način:

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

Vrednosti izlaznog vektora su nenegativne i sumiraju se na 1, pa se mogu interpretirati kao raspodela verovatnoća diskretne slučajne veličine koja uzima C različitih vrednosti. Za vrednost predviđanja uzima se kategorija koja odgovara izlazu sa najvišom vrednošću (verovatnoćom). Ono što je još karakteristično za ovu funkciju je to što dodatno naglašava razlike među koordinatama polaznog vektora. Zbog dejstva eksponencijalne funkcije najveća pozitivna vrednost će biti preslikana u novu vrednost koja još više odskače od drugih. Na taj način se postiže veća sigurnost

[11] Nazivaju se skrivenim slojevima jer se njegove vrednosti obično koriste samo unutar modela.
[12] Funkcija identiteta, $id(x) = x$.

klasifikatora i fokusira učenje na parametre koji potkrepljuju taj izlaz, u pozitivnom ili negativnom smeru[9].

S obzirom na to da postoji verovatnosna interpretacija, smisleno je vrednosti parametara birati tako da verovatnoća raspoloživog skupa podataka bude maksimalna pri izabranim vrednostima parametara[10]. To se postiže maksimizacijom *funkcije verodostojnosti* (eng. *likelihood function*)

$$\mathcal{L}(w) = P_w(y_1, ..., y_N | x_1, ..., x_N)$$

gde je $\mathcal{D} = \{(x_i, y_i) \mid i = 1, ..., N\}$ skup raspoloživih podataka, x_i ulazni podaci, a y_i vektori dimenzije C koji se sastoje od C-1 nula i jedne jedinice, pri čemu je jedinica na poziciji j ukoliko je odgovarajućem podatku pridružena kategorija j. Uz pretpostavku nezavisnosti pojedinačnih podataka, tj. uslovne nezavisnosti vrednosti y_i za date vrednosti x_i, dobija se jednostavnija forma funkcije verodostojnosti:

$$\mathcal{L}(w) = \prod_{i=1}^{N} P_w(y_i | x_i)$$

Za verovatnoću jednog podatka važi:

$$P_w(y_i | x_i) = \prod_{j=1}^{C} \left(\frac{e^{x_{ij}}}{\sum_{k=1}^{C} e^{x_{ik}}} \right)^{y_{ij}}$$

gde su x_{ij} i y_{ij} j-te koordinata vektora x_i i y_i.[13] Funkcija verodostojnosti je onda data sledećim izrazom:

$$\mathcal{L}(w) = \prod_{i=1}^{N} \prod_{j=1}^{C} \left(\frac{e^{x_{ij}}}{\sum_{k=1}^{C} e^{x_{ik}}} \right)^{y_{ij}}$$

Potrebno je rešiti problem

$$\max_{w} \mathcal{L}(w)$$

Za gradijentne metode optimizacije je reprezentacija funkcije u vidu proizvoda problematična iz dva razloga:

- množenjem velikog broja malih ili velikih vrednosti može doći do potkoračenja ili prekoračenja
- komplikovano izračunavanje parcijalnih izvoda proizvoda, što implicira da će izračunavanje gradijenata i ceo proces optimizacije biti vremenski veoma zahtevan

Iz ovih razloga se umesto funkcije verodostojnosti koristi njen logaritam, kojim se proizvod prevodi u sumu. Kako je logaritam monotono rastuća funkcija, tačke u kojima se dostiže maksimum funkcije verodostojnosti i njenog logaritma se podudaraju. Takođe, kako su optimizacioni metodi najčešće definisani za problem minimizacije, umesto maksimizacije logaritma funkcije verodostojnosti minimizuje se njegova ne-

[13] Primetiti da y_{ij} ima vrednost 0 za sve indekse j osim za j koje odgovara kategoriji i-tog podatka, kada ima vrednost 1.

gativna vrednost, pa je problem koji se rešava sledeći:

$$\min_w -\log \mathcal{L}(w) = \min_w -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log \frac{e^{x_{ij}}}{\sum_{k=1}^{C} e^{x_{ik}}}$$

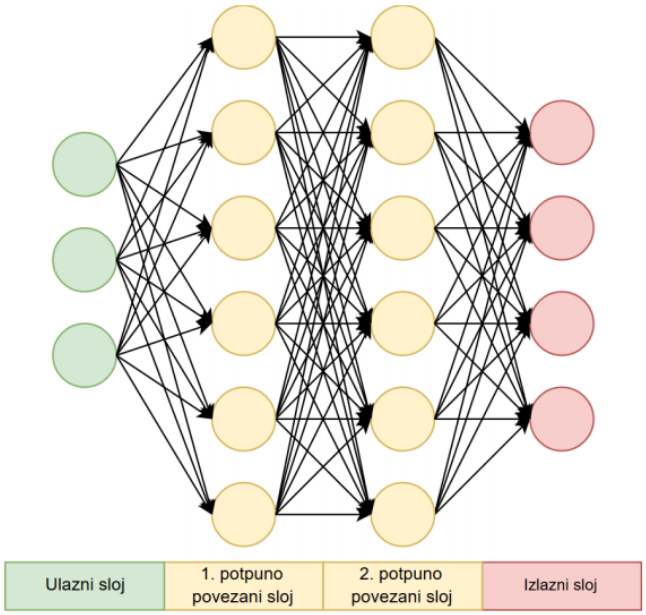
Ovime je ujedno definisana funkcija greške (eng. *loss*)[14] za problem klasifikacije kod neuronskih mreža

$$L(x_i, y_i) = -\sum_{j=1}^{C} y_{ij} \log \frac{e^{x_{ij}}}{\sum_{k=1}^{C} e^{x_{ik}}} \tag{3.2}$$

Funkcija L se naziva *kategoričkom unakrsnom entropijom* (eng. *categorical crossentropy*) i nju je potrebno minimizovati na datom skupu podataka.

3.3 Potpuno povezani sloj mreže

Neuronske mreže koje se sastoje od potpuno povezanih slojeva predstavljaju najstariji od trenutno korišćenih modela neuronskih mreža. Naziv su dobile po strukturi povezanosti neurona - neuroni se u okviru potpuno povezanog sloja povezuju tako što svi neuroni jednog sloja kao ulaze uzimaju vrednosti svih neurona iz prethodnog, a svoje vrednosti prosleđuju svim neuronima narednog sloja. Grafički prikaz jedne potpuno povezane neuronske mreže dat je na slici 3.7. Na njoj su prikazani ulazni sloj, dva skrivena potpuno povezana sloja i izlazni sloj. Jedinice ulaznog sloja ne predstavljaju neurone, već jedinice koje prihvataju ulaze, i njihov broj odgovara dimenziji ulaznih podataka. Dimenzije skrivenih slojeva mogu biti različite, dok broj neurona izlaznog sloja odgovara broju vrednosti koje neuronska mreža predviđa.



Slika 3.7: Potpuno povezana neuronska mreža

[14] Funkcija greške meri odstupanje (jednog) predviđanja modela u odnosu na stvarnu vrednost.

Potpuno povezane neuronske mreže se uobičajeno koriste za modelovanje tabelarnih podataka, odnosno podataka predstavljenih u vidu vektora svojstava fiksne dužine, dok se pojedinačni potpuno povezani slojevi mogu koristiti kao deo drugih vrsta neuronskih mreža, najčešće u završnim slojevima mreže.

3.4 Rekurentni sloj mreže

Za obradu sekvencijalnih podataka koriste se razne vrste rekurentnih slojeva neurona koji imaju strukturu specijalizovanu za modelovanje zavisnosti među elementima sekvence. Sekvencijalni podaci su najčešće promenljive dužine i sastoje se od jednostavnijih elemenata koji imaju određeni redosled. Primeri takvih podataka su rečenice prirodnog jezika, biološke sekvence i vremenske serije.

Osnovni princip rekurentnih slojeva je da se elementi ulazne sekvence obrađuju u koracima, element po element, pri čemu se informacija o obrađenim elementima sekvence akumulira u vidu vektora koji predstavlja skriveno stanje. Da bi ovaj sistem funkcionisao, potrebno je definisati kako naredno stanje zavisi od prethodnog stanja i od tekućeg ulaza (elementa sekvence). Takođe, potrebno je definisati i na koji način izlaz iz rekurentnog sloja u svakom koraku zavisi od tekućeg stanja. Ove zavisnosti se izražavaju linearnim transformacijama sa nelinearnom aktivacionom funkcijom, pri čemu se matrice koje definišu linearne transformacije uče.

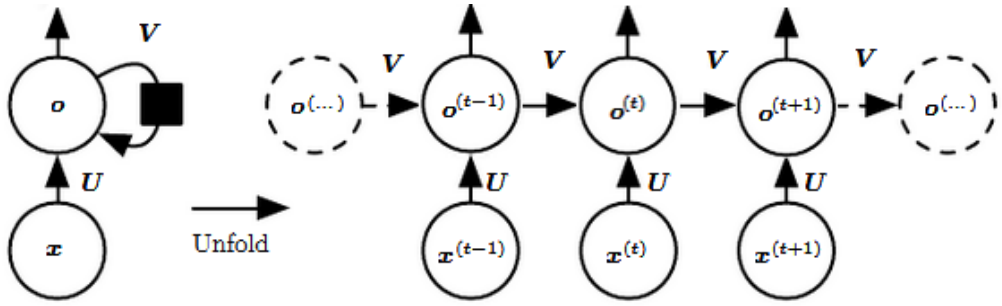
Postoji više vrsta rekurentnih slojeva koji se razlikuju u tome na koji način se definiše i ažurira skriveno stanje sloja i kako se određuje izlaz u svakom koraku. Ovde će biti opisane 3 vrste - *jednostavni rekurentni sloj*, *osnovni rekurentni sloj sa skrivenim stanjem* i *rekurentni sloj sa dugom kratkoročnom memorijom*.

Jednostavni rekurentni sloj

Jednostavni rekurentni sloj (eng. *Simple RNN Layer*) suštinski nema skriveno stanje, odnosno može se reći da se za skriveno stanje uzima izlaz sloja za prethodni element sekvence. Pri obradi sekvence u trenutku t se na osnovu ulaznog elementa sekvence $x^{(t)}$ i prethodnog izlaza sloja $o^{(t)}$ odlučuje o novom izlazu $o^{(t)}$. Ovaj rekurentni sloj karakterišu dve matrice - matrica U koja povezuje trenutni ulaz $x^{(t)}$ sa trenutnim izlazom $o^{(t)}$ i matrica V koja povezuje trenutni izlaz $o^{(t)}$ sa prethodnim izlazom $o^{(t-1)}$. Formalno, jednostavni rekurentni sloj se definiše sledećom rekurentnom jednačinom:

$$\begin{aligned} o^{(0)} &= 0 \\ o^{(t)} &= g(Ux^{(t)} + Vo^{(t-1)} + b) \end{aligned}$$

Matrice U i V, kao i vektor slobodnih članova b su parametri modela koji se uče. Kao izlaz rekurentnog sloja za datu sekvencu $[x^{(1)}, x^{(2)}, ..., x^{(n)}]$ može se uzeti odgovarajuća sekvenca izlaza $[o^{(1)}, o^{(2)}, ..., o^{(n)}]$ ili samo izlaz sloja za poslednji element sekvence $o^{(n)}$. Na slici 3.8. je prikazana grafička ilustracija jednostavnog rekurentnog sloja.



Slika 3.8: Ilustracija jednostavnog rekurentnog sloja. Levo je data kompaktna reprezentacija, a desno takozvano razmotavanje sloja kroz vreme. Svaki krug predstavlja jedan neuron ili jedinicu ulaznog sloja. Kvadrat na strelici označava rekurentnu vezu izlaza sloja sa izlazom u prethodnom trenutku. [16]

Osnovni rekurentni sloj sa skrivenim stanjem

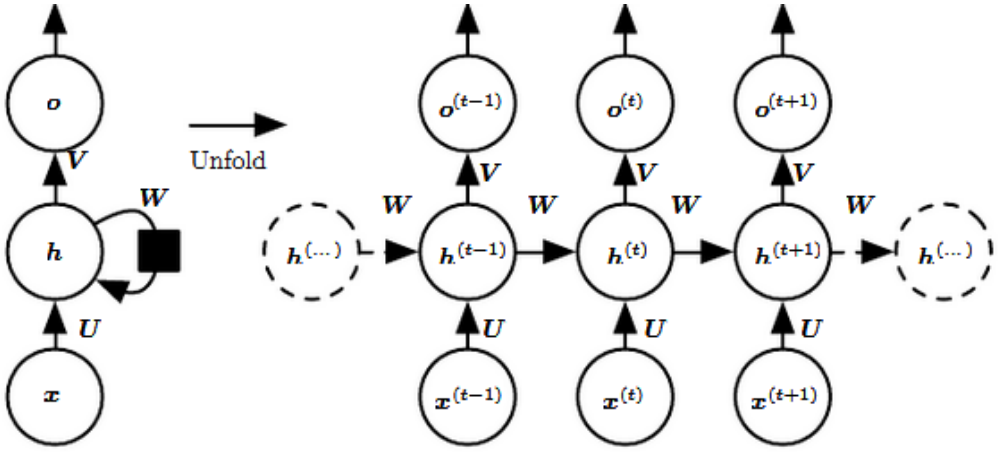
Osnovna varijanta rekurentnog sloja sa skrivenim stanjem (eng. *Base RNN Layer*) u posebnom neuronu akumulira informaciju o prethodno obrađenim elementima sekvence. Skriveno stanje h u trenutku t se ažurira na osnovu ulaznog elementa sekvence $x^{(t)}$ i prethodnog skrivenog stanja $h^{(t-1)}$, a zatim se na osnovu trenutnog skrivenog stanja $h^{(t)}$ odlučuje o novom izlazu $o^{(t)}$. Ovaj rekurentni sloj karakterišu tri matrice - matrica U koja povezuje trenutni ulaz $x^{(t)}$ sa trenutnom vrednošću skrivenog stanja $h^{(t)}$, matrica V koja povezuje skriveno stanje iz prethodnog trenutka $h^{(t-1)}$ i tekuće stanje $h^{(t)}$, i matrica W kojom se povezuje skriveno stanje $h^{(t)}$ sa izlazom $o^{(t)}$. Formalno, osnovni rekurentni sloj sa skrivenim stanjem se definiše sledećim rekurentnim jednačinama:

$$\begin{aligned} h^{(0)} &= 0 \\ h^{(t)} &= g(Ux^{(t)} + Vh^{(t-1)} + b_h) \\ o^{(t)} &= g(Wh^{(t)} + b_o) \end{aligned}$$

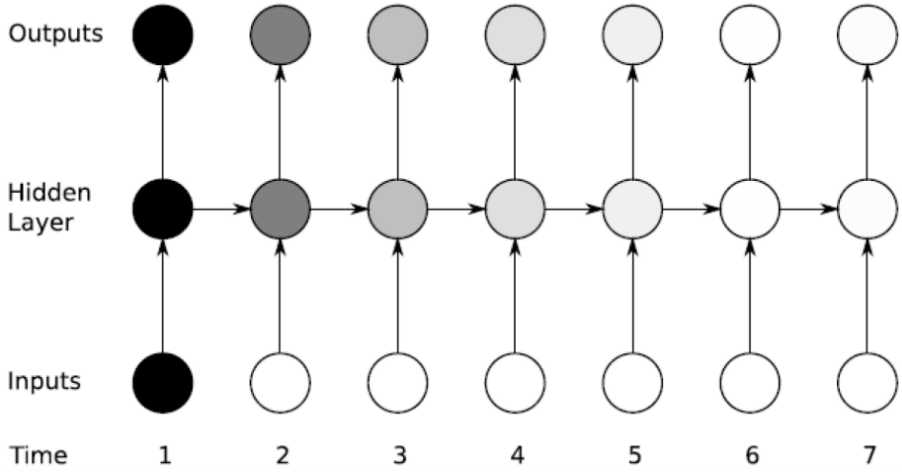
Matrice U, V i W, kao i vektori slobodnih članova b_h i b_o su parametri modela koji se uče. Kao izlaz rekurentnog sloja za datu sekvencu $[x^{(1)}, x^{(2)}, ..., x^{(n)}]$ može se uzeti odgovarajuća sekvenca izlaza $[o^{(1)}, o^{(2)}, ..., o^{(n)}]$ ili samo izlaz sloja za poslednji element sekvence $o^{(n)}$. Na slici 3.9. je prikazana grafička ilustracija osnovnog rekurentnog sloja sa skrivenim stanjem.

Rekurentni sloj sa dugom kratkoročnom memorijom

Prethodno opisan osnovni rekurentni sloj sa skrivenim stanjem ne omogućava dugoročno čuvanje informacije iz prethodno obrađenih elemenata sekvence. Kako se skriveno stanje ažurira linearnom kombinacijom prethodnog stanja i trenutnog ulaza, najveći uticaj na novo stanje pridaje se novom ulazu, dok doprinosi ranijih ulaza eksponencijalno opadaju sa svakim korakom. To znači da tim modelom nije moguće modelovati dugoročne zavisnosti u podacima. Na slici 3.10. je ilustrovano kako uticaj informacija iz ranijih elemenata sekvence bledi tokom vremena.



Slika 3.9: Ilustracija osnovnog rekurentnog sloja sa skrivenim stanjem. Levo je data kompaktna reprezentacija, a desno takozvano razmotavanje sloja kroz vreme. Svaki krug predstavlja jedan neuron ili jedinicu ulaznog sloja. Kvadrat na strelici označava rekurentnu vezu skrivenog stanja sloja sa stanjem u prethodnom trenutku. [16]



Slika 3.10: Ilustracija nemogućnosti dugoročnog čuvanja informacije u osnovnom rekurentnom sloju sa skrivenim stanjem [11]

Ovaj problem prevazilazi rekurentni sloj sa dugom kratkoročnom memorijom (eng. *Long Short Term Memory Layer*, skraćeno *LSTM*). Sloj *LSTM* je složena jedinica mreže sa specifičnom strukturom koja omogućava kontrolu koliko informacija iz prethodnog stanja i tekućeg ulaza treba propustiti dalje u sledeće stanje i na izlaz sloja. Takođe, ova vrsta rekurentnog sloja ublažava problem nestajućih i eksplodirajućih gradijenata koji nastaje pri optimizaciji rekurentne mreže za ulazne sekvence velike dužine. Iz ovih razloga *LSTM* sloj predstavlja standardni izbor prilikom izgradnje modela rekurentne mreže.

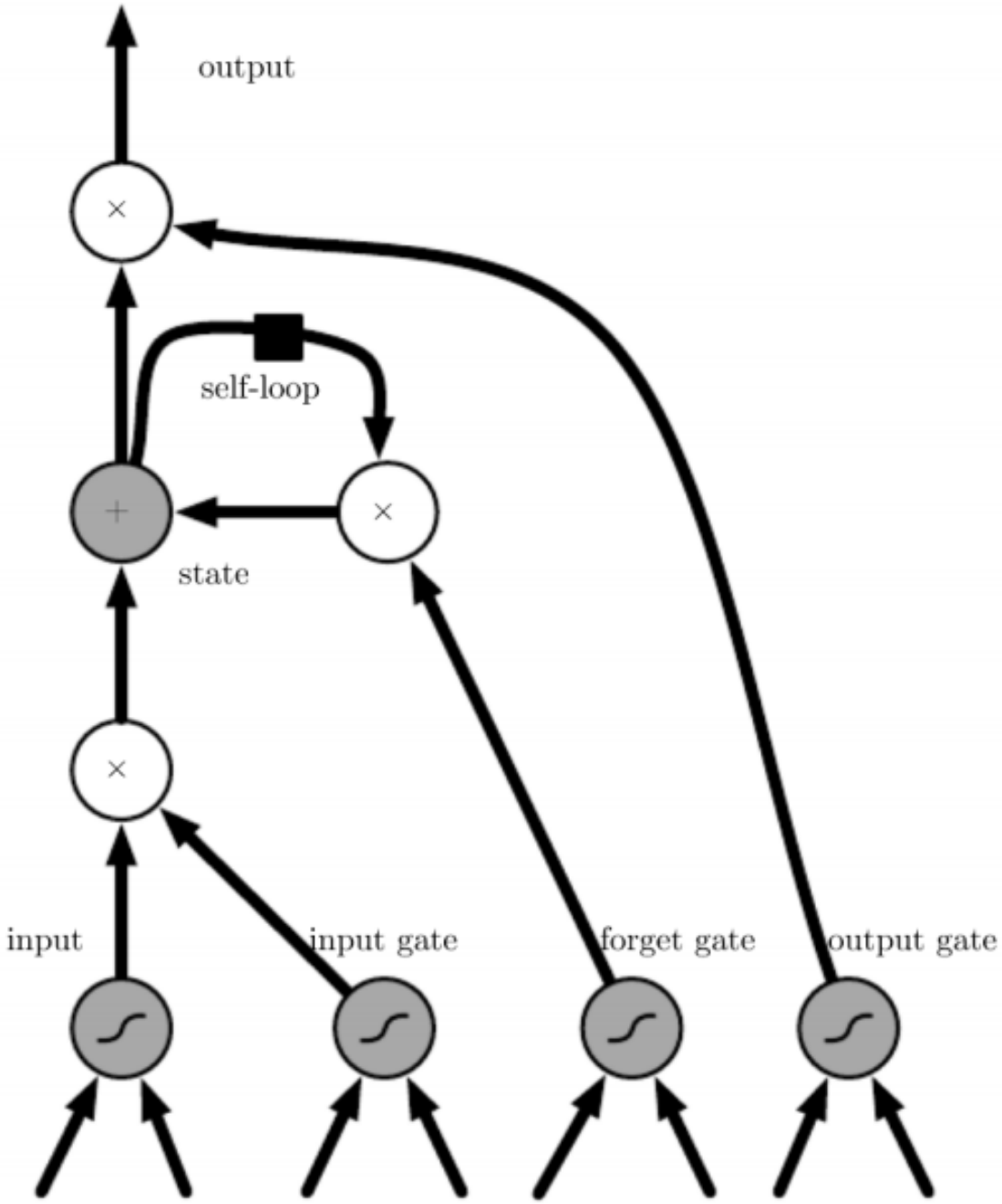
Osnovni princip *LSTM* sloja je postojanje takozvane ćelije koja čuva skriveno stanje i kontroliše ulazak i izlazak informacija. Uz skriveno stanje ove ćelije imaju svoje interno stanje (eng. *cell state*), ulaznu kapiju (eng. *input gate*), kapiju za

zaboravljanje (eng. *forget gate*) i izlaznu kapiju (eng. *output gate*) kojima, redom, kontrolišu prisutnost informacija iz prethodnog stanja i selekciju informacija iz trenutnog ulaza koje utiču na interno stanje ćelije i izlaz iz ćelije. Kapije koje kontrolišu navedene aspekte imaju strukturu osnovnog rekurentnog sloja sa skrivenim stanjem i određene su parametrima koji se uče.

U trenutku t na osnovu ulaza $x^{(t)}$, vrednosti skrivenog stanja $h^{(t-1)}$ i internog stanja ćelije $c^{(t-1)}$ prethodnog koraka obrade, treba odlučiti o novoj vrednosti skrivenog sloja $h^{(t)}$ i stanja $c^{(t)}$. Kapijom za zaboravljanje $f^{(t)}$ se kontroliše koliko informacija iz prethodnog stanja treba dalje propustiti. Pokoordinatnim množenjem sa vrednošću $c^{(t-1)}$ dobijaju se filtrirane informacije stanja. Dalje se, na sličan način, izračunavaju vrednosti ulazne kapije $i^{(t)}$ i vrednosti izlazne kapije $o^{(t)}$. Ulaznom kapijom se filtriraju informacije međustanja ćelije $\tilde{c}^{(t)}$ koje treba dodati informacijama koje se filtriraju kapijom zaboravljanja kako bi se dobilo novo stanje $c^{(t)}$. Nova vrednost skrivenog stanja $h^{(t)}$ se dobija na osnovu vrednosti izlazne kapije $o^{(t)}$ i novog stanja ćelije $c^{(t)}$. Formalno, $LSTM$ sloj se definiše sledećim rekurentnim jednačinama:

$$
\begin{aligned}
o^{(0)} &= 0 \\
h^{(0)} &= 0 \\
f^{(t)} &= \sigma(W_f x^{(t)} + U_f h^{(t-1)} + b_f) && \text{Kapija zaboravljanja} \\
i^{(t)} &= \sigma(W_i x^{(t)} + U_i h^{(t-1)} + b_i) && \text{Ulazna kapija} \\
o^{(t)} &= \sigma(W_o x^{(t)} + U_o h^{(t-1)} + b_o) && \text{Izlazna kapija} \\
\tilde{c}^{(t)} &= g(W_{\tilde{c}} x^{(t)} + U_{\tilde{c}} h^{(t-1)} + b_{\tilde{c}}) && \text{Međustanje ćelije} \\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Stanje ćelije} \\
h^{(t)} &= o^{(t)} \circ g(c^{(t)}) && \text{Skriveno stanje}
\end{aligned}
$$

Matrice W_f, W_i, W_o, $W_{\tilde{c}}$, U_f, U_i, U_o, $U_{\tilde{c}}$, kao i vektori slobodnih članova b_f, b_i, b_o, $b_{\tilde{c}}$ su parametri modela koji se uče. Pored navednog skupa parametara, ovaj rekurentni sloj karakterišu još sigmoidne funkcije koje se primenjuju u svojstvu aktivacionih funkcija za kapiju zaboravljanja, ulaznu i izlaznu kapiju, kako bi kapija davale vrednosti iz opsega $[0, 1]$ i odražavale udeo informacije koja se propušta. Slično kao kod jednostavnog rekurentnog sloja, skriveno stanje sloja je ujedno i izlaz iz sloja. Kao izlaz rekurentnog sloja za datu sekvencu $[x^{(1)}, x^{(2)}, ..., x^{(n)}]$ može se uzeti odgovarajuća sekvenca izlaza $[h^{(1)}, h^{(2)}, ..., h^{(n)}]$ ili samo izlaz sloja za poslednji element sekvence $h^{(n)}$. Na slici 3.11. je prikazana grafička ilustracija $LSTM$ sloja.



Slika 3.11: Struktura LSTM jedinice [11]

Glava 4

4 Rad sa podacima

U ovom poglavlju biće opisano na koji način su prikupljeni i odabrani podaci nad kojima će model biti obučavan. Zatim će biti objašnjen način predstavljanja podataka kako bi se na njima mogle primeniti metode mašinskog učenja.

4.1 Prikupljanje i generisanje podataka

Podaci o enzimima se mogu pronaći u raznim biomedicinskim bazama podataka koje pored sekvenci i EC brojeva enzima, koji su korišćeni u ovom radu, sadrže i podatke o raznim biohemijskim svojstvima. Neke od njih su prikazane su u tabeli 4.1.

Baza podataka	URL	Opis
UniProtKB	www.uniprot.org	Proteinske sekvence i funkcije proteina
PDB	www.rcsb.org/pdb	Eksperimentalno utvrđene strukture
BRENDA	www.brenda-enzymes.org	Sadrži opsežne funkcionalne podatke
ExPASy	enzyme.expasy.org	Informacije o tipu katalisane reakcije i potrebnih kofaktora (ako postoje)
KEGG	www.genome.ad.jp/kegg	Pruža mapiranje sa molekulskim interakcijama

Tabela 4.1: Prikaz nekih javno dostupnih biomedicinskih baza podataka

Podaci o enzimima korišćeni u ovom radu preuzeti su iz UniProtKB baze podataka, i to u dva formata:

* FASTA formatu
* *Tab-separated* formatu koji sadrži ID i EC brojeve enzima

FASTA format je standardni format za zapisivanje nukleotidnih i proteinskih sekvenci. Sastoji se od specifičnog FASTA zaglavlja i same sekvence. U upotrebi je više različitih vrsta FASTA zaglavlja, neretko svaka od baza podataka koristi svoj poseban oblik zaglavlja. Bez obzira na specifičan oblik zaglavlja koje se koristi, za zapis u FASTA formatu važe sledeća opšta pravila:

– linija zaglavlja počinje simbolom '>' za kojim slede identifikacione i neke dodatne informacije
– u narednim redovima je zapis sekvence

Kod UniProtKB baze podataka format FASTA zaglavlja je `>sp|ID|entryName desc`, gde je `sp` - oznaka da se radi o UniProtKB/Swiss-Prot bazi podataka, `ID` – identifikator, `entryName` - naziv i `desc` – opis sekvence. Primer zapisa enzima u FASTA formatu:

```
>sp|A6SSW9|BOA17_BOTFB Oxidoreductase BOA17 OS=Botryotinia fuckeliana
(strain B05.10) OX=332648 GN=BOA17 PE=2 SV=1
MTKIWFITGSSRGLGLAIAEAALNNGDSVIATARKPEQLTNLVNKFGKERVFPVALDVTD
NNQVLQAVKSGHEKFGRIDVVINNAGYANTAAVEDIDVDDFCAQVEANLMGVVYVSKAVL
PILRQQKSGHIFQVSSLGGRIGAPGLSAYQSAKWAVGGFSTVLAQEVASFGIKITVLEPG
GIRTDWAGSSMQVPTVSEPYQATVGAFAESLRKSSGSEVSIPSKIANIVLKVLGEKEPPL
RLLVGPDAVEYAGKAAEVLSASDEKWRELSLASA
```

Pored toga što su iz podataka u FASTA formatu uzete sekvence enzima, ovaj format je potreban i kao ulaz programa za nalaženje funkcionalnih domena enzima. HMMER je paket programa za analizu bioloških sekvenci skrivenim Markovljevim modelima[15](eng. *Hidden Markov Model*, skraćeno *HMM*). Programi pronalaze homologne sekvence na osnovu jednostrukih ili višestrukih poravnanja sa sekvencama iz različitih baza podataka. Za dobijanje funkcionalnih domena korišćen je program *hmmscan* koji vrši poravnanje delova sekvenci sa domenima iz date baze podataka. HMMER paket se može preuzeti na adresi http://hmmer.org.

Funkcionalni domeni u odnosu na koje su vršena poravnanja sekvenci enzima su iz *Pfam* baze podataka (https://pfam.xfam.org). Svaki od funkcionalnih domena u ovoj bazi podataka predstavljen je jedinm skrivenim Markovljevim modelom. Program *hmmpress* priprema bazu podataka za korišćenje sa programom *hmmscan*.

```
#                --- full sequence ---- --- best 1 domain ---- --- domain number estimation ----
# target name    E-value  score  bias   E-value  score  bias   exp reg clu  ov env dom rep inc description of target
#------------   -------- ------ -----  -------- ------ -----   --- --- --- --- --- --- --- --- ---------------------
SH2              5.7e-25   87.1   0.3     1e-24   86.3   0.2   1.5   2   0   0   2   2   2   1 SH2 domain
SH3_1            1.1e-13   50.5   0.0   1.9e-13   49.7   0.0   1.4   1   0   0   1   1   1   1 SH3 domain
SH3_9            8.5e-10   38.2   0.0   1.4e-09   37.6   0.0   1.4   1   0   0   1   1   1   1 Variant SH3 domain
SH3_2            1.1e-08   34.6   0.0     2e-08   33.7   0.0   1.5   1   0   0   1   1   1   1 Variant SH3 domain
SH3_3            3.8e-06   27.0   1.0   1.2e-05   25.4   0.3   2.1   2   0   0   2   2   2   1 Bacterial SH3 domain
SH3_6              0.015   14.8   0.0     0.031   13.7   0.0   1.5   1   0   0   1   1   1   0 SH3 domain (SH3b1 type)
#
# Program:         hmmscan
# Version:         3.2.1 (June 2018)
# Pipeline mode:   SCAN
# Query file:      primer.fasta
# Target file:     Pfam-A.hmm
# Option settings: src/hmmscan --tblout primer.hmmer --notextw Pfam-A.hmm primer.fasta
# Current dir:     /home/nevena/Downloads/hmmer-3.2.1
# Date:            Tue Sep 15 21:17:49 2020
```

Slika 4.12: Primer izlaza programa *hmmscan*

Na slici 4.12 je prikazan izlaz programa *hmmscan* za jednu sekvencu u sažetoj verziji. Kompletan izlaz sadrži detaljne informacije o poziciji i strukturi pronađenog poravnanja sekvence enzima sa svakim od pronađenih funkcionalnih domena. Za potrebe ovog rada iz sažete verzije izlaza ekstrahovani su ID-evi funkcionalnih domena (kolona `target name`).

[15]Skriveni Markovljevi modeli su statistički modeli pomoću kojih je moguće predviđanje verovatnoća ostvarenja budućih događaja na osnovu poznatih verovatnoća viđenih događaja. U kontekstu poravnanja sekvenci verovatnoća koju daje skriveni Markovljev model koristi se za određivanje sličnosti dve ili više sekvenci.

4.2 Selekcija podataka

UniProtKB baza podataka ima ukupno 266 897 enzima. Po ugledu na referentni rad [1], za skup podataka na kome je model obučavan uzet je podskup ovog skupa enzima konstruisan na sledeći način:

* zbog jednoznačnosti labela za klasifikaciju (klasa prema kojima se vrši razvrstavanje ulaznih podataka) izbačeni su enzimi koji imaju dodeljeno više od jednog EC broja
* enzimi koji imaju nekompletan EC broj su izbačeni kako bi labele za klasifikaciju bile potpune
* kako je rađena klasifikacija samo na dva nivoa hijerarhije, za svaki enzim su iz EC brojeva izdvojena prva dva broja koja će predstavljati labele za klasifikaciju
* da bi se mogla izvršiti stratifikacija podataka na trening, validacionom i test skupu, izbačene su klase (tj. enzimi tih klasa) koje imaju manje od 3 instance
* na kraju su izbačeni enzimi koji imaju sekvencu kraću od 50 ili dužu od 5000 aminokiselina, što vodi boljim performansima algoritama učenja

Nakon navedenih izbacivanja u skupu podataka je ostalo 212 370 enzima.

4.3 Predstavljanje podataka

Prikupljeni i generisani podaci su objedinjeni na osnovu ID-a enzima, tj. napravljeno je mapiranje koje svakom ID-u enzima pridružuje sekvencu, EC brojeve za prva dva nivoa klasifikacije i listu funkcionalnih domena. Na slici 4.13 je prikazano prvih nekoliko instanci skupa podataka.

Id enzima	Sekvenca	EC 1	EC 2	Funkcionalni domeni
Q6LLK1	MDKYVVFGNPIAQSKSPFIHTLFARQTAQKMEYTAELAPADGFKLA...	1	1	['Shikimate_dh_N', 'Shikimate_DH', 'Sacchrp_dh...
Q9JPA3	MQLTLWTYEGPPHVGAMRVATALDDVHYVLHAPQGDTYADLLFTMI...	1	3	['Oxidored_nitro', 'PCP_red']
Q12XL7	MSELTTGRFSISDLDNVQITINNIVGAIEKQSDDIDVEMGPTVKPG...	1	2	['Prismane', 'Fer4_7', 'Fer4_9', 'Fer4_17', 'F...
B3QQZ4	MSLVLAVYGKGGIGKSTTSANISAALALKGAKVLQIGCDPKHDSTF...	1	3	['Fer4_NifH', 'CbiA', 'AAA_31', 'ParA', 'ArsA_...
A1VXU6	MKIKVGILGASGYAGNELVRILLNHPKVEISYLGSSSSVGQNYQDL...	1	2	['Semialdhyde_dh', 'Semialdhyde_dhC', 'DapB_N']

Slika 4.13: Izgled skupa podataka

Sekvence su predstavljene u obliku niski karaktera (oznaka aminokiselina), EC brojevi rednim brojem klase, a funkcionalni domeni listom ID-eva domena. Podaci u ovom obliku nisu pogodni za primenu metoda mašinskog učenja zbog čega je neophodno njihovo pretprocesiranje.

Kategoričke promenljive

Kategoričkim se smatraju promenljive koje uzimaju vrednosti iz konačnog skupa na kome nije definisan poredak. Njima se mogu pridružiti, ukoliko već nisu, numeričke oznake ali se one ne mogu koristiti kao numerički atributi. Za potrebe metoda

mašinskog učenja kategoričke promenljive se *binarno kodiraju* - pridružuje im se niz binarnih promenljivih koje predstavljaju indikatore o originalnoj vrednosti kategoričke promenljive. Ukoliko kategorička promenljiva x uzima C različitih vrednosti, zamenjuje se sa C binarnih promenljivih x_1, ..., x_C. Vrednost i-te kategorije predstavlja se vrednostima (x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_C) = (0, ..., 0, 1, 0, ..., 0), dok se NaN vrednost, odnosno vrednost za koju kategorička promenljiva nije definisana, predstavlja vrednostima (x_1, ..., x_C) = (0, ..., 0). Biblioteka `sklearn`[16] pruža ovu funkcionalnost kroz klasu `OneHotEncoder` paketa `preprocessing`.

Pretprocesiranje sekvenci

Kao što je već rečeno, proteini su izgrađeni od 20 esencijalnih aminokiselina, a svaka aminokiselina ima jedinstveni simbol (slika 2.2). Svaku od esencijalnih aminokiselina predstavljamo vektorom dimenzije 20 koji se sastoji od svih nula i jedne jedinice na odgovarajućoj poziciji, dok aminokiseline koje nisu esencijalne predstavljamo nula vektorom. To znači da sekvencu dužine L ($50 \leq L \leq 5000$) predstavljamo matricom dimenzije $L \times 20$.

Za dobijanje tzv. *one-hot* reprezentacije sekvenci korišćen je `OneHotEncoder` iz `sklearn` biblioteke. Pre primene metoda klase `OneHotEncoder`, sekvencu je potrebno transformisati u vektor kolona karaktera kako bi enkoder pojedinačne karaktere (simbole aminokiselina) prepoznao kao vrednosti kategoričke promenljive i svakom karakteru pridružio odgovarajući vektor vrste dimenzije 20, odnosno celoj sekvenci pridružio matricu na prethodno opisani način.

Pretprocesiranje funkcionalnih domena

U $Pfam$ bazi postoji 16 451 funkcionalnih domena koje *hmmscan* program pretražuje u svakoj od sekvenci enzima. Izlaz iz *hmmscan* programa je lista ID-eva funkcionalnih domena pronađenih u sekvenci datog enzima. Funkcionalne domene predstavljamo vektorom $[I_1, I_2, ..., I_{16451}]$, pri čemu je $I_k = 1$ na pozicijama koje odgovaraju ID-evima domena iz liste, a inače $I_k = 0$.

Ovakva reprezentacija je takođe dobijena pomoću `OneHotEncoder`-a iz `sklearn` biblioteke. Lista ID-eva funkcionalnih domena dužine L se najpre transformiše u vektor kolona kojem enkoder zatim pridružuje binarnu matricu dimenzije $L \times 16451$ sa jedinicama na pozicijama koje odgovaraju ID-evima domena. Prethodno opisana reprezentacija dobija se sumiranjem elemenata matrice po kolonama.

Pretprocesiranje EC brojeva

Pošto su konstruisana dva modela, za klasifikaciju na prvom i drugom nivou hijerarhije, EC brojevi su predstavljani na dva načina. Na prvom nivou hijerarhije ima 7 klasa, pa su instancama kao labele pridruženi vektori dimenzije 7 koji imaju sve nule i jednu jedinicu na poziciji koja odgovara prvom EC broju enzima. Ukupan broj klasa na drugom nivou hijerarhije je 68[17]. Broj podklasa za svaku od osnovih klasa prikazan je u tabeli 4.2.

[16] Biblioteka SciKit-Learn je biblioteka za programski jezik Python koja pruža podršku za metode mašinskog učenja i preprocesiranje podataka.

[17] Ovaj broj se razlikuje od ukupnog broja postojećih klasa na drugom nivou hijerarhije (74) pošto su prilikom selekcije podataka izbačene klase kardinalnosti < 3

Osnovna klasa	Broj podklasa
EC 1	22
EC 2	10
EC 3	11
EC 4	6
EC 5	7
EC 6	6
EC 7	6

Tabela 4.2: Broj klasa na drugom nivou hijerarhije

Na osnovu prvog i drugog EC broja instancama su kao labele pridruženi vektori dimenzije 68 koji imaju sve nule i jednu jedinicu na indeksu klase sa drugog nivoa hijerarhije. Indeksi iz opsega $[0, 21]$ odgovaraju podklasama osnovne klase EC1, sledećih 10 indeksa odgovara podklasama osnovne klase EC2, itd.

Glava 5

5 Izgradnja modela i detalji implementacije

Klasifikacija enzima određivanjem njihovog EC broja na osnovu sekvence i funkcionalnih domena vrši se neuronskom mrežom koja se sastoji iz dva dela - potpuno povezanog i rekurentnog. Ulaz rekurentnog dela mreže su *one-hot* reprezentacije sekvenci enzima, dok su funkcionalni domeni, odnosno njihove vektorske reprezentacije, ulaz potpuno povezanog dela mreže. Pre razdvajanja skupa podataka na ulaz za rekurentni i potpuno povezani deo mreže, podaci su podeljeni na skupove za trening, validaciju i testiranje. Takođe, za potrebe hijerarhijske klasifikacije definisana je funkcija greške koja uzima u obzir pogrešnu klasifikaciju na prvom i na drugom nivou hijerarhije.

5.1 Podela podataka na trening, validacioni i test skup

Kako bi se ocenila sposobnost predviđanja modela i kako bi ta ocena bila nepristrasna, iz polaznog skupa podataka se izdvaja poseban skup na kome se model testira i formira ocena odabranom metrikom. Preostali deo skupa podataka se koristi za obučavanje modela. Podaci na kojima se model obučava nazivaju se *podacima za trening*, a njihov skup *trening skup*. Podaci za evaluaciju modela čine *test skup*. Ova dva skupa podataka moraju biti disjunktna.

Algoritam učenja može biti konfigurabilan po različitim aspektima. U slučaju neuronskih mreža osnovni metaparametar koji treba odabrati je broj epoha treniranja mreže. Epoha predstavlja jedan prolazak kroz ceo skup podataka za trening. Na kraju svake epohe izračunava se vrednost funkcije greške i na osnovu nje se ažuriraju parametri mreže. Potrebno je odrediti nakon koliko epoha se treba zaustaviti sa treniranjem mreže, odnosno za koju vrednost ovog metaparametra se dobija najbolji model.

Isti princip sa odvajanjem posebnog skupa na kome se finalni model ocenjuje primenjuje se i u slučaju ocenjivanja modela za različite vrednosti metaparamara. Iz skupa za treniranje se izdvaja manji skup podataka koji se naziva *validacioni skup*. Na njemu se odabranom metrikom ocenjuje model za svaku od vrednosti metaparametara i bira najbolji model, odnosno vrednosti metaparametra za koju model ima najbolju ocenu u smislu te metrike. Na kraju, model sa najboljim vrednostima metaparametara se obučava na uniji skupova za trening i validaciju, a ocenjuje na test skupu.

Slika 5.14: Podela podataka na trening, validacioni i test skup

Prilikom podele skupa podataka na skupove za treniranje, validaciju i testiranje potrebno je voditi računa da raspodela podataka na svakom od skupova bude ista kao na celom skupu podataka. Kod problema klasifikacije sa nebalansiranim skupom podataka je ovo posebno važno kako bi instance manjinskih klasa bile podjednako zastupljene u svakom od skupova. Tehnike podele podataka koje omogućavaju čuvanje raspodele nazivaju se tehnikama *stratifikacije*.

Metod `train_test_split` iz paketa `model_selection` biblioteke `sklearn` vrši podelu skupa podataka na skup za treniranje i skup za testiranje. Srazmera ovih skupova određuje se parametrima `train_size` ili `test_size`. Ovi parametri se izražavaju vrednostima od 0 do 1 i predstavljaju procentualni udeo naznačenog skupa. Kroz parametar `stratify` se prosleđuju labele pridružene podacima na osnovu kojih treba izvršiti stratifikaciju.

Generisanje skupova za trening, validaciju i testiranje

`OneHotEncoder` kojim smo dobili odgovarajuće reprezentacije sekvenci i funkcionalnih domena vraća `scipy.sparse.csr_matrix` - retku matricu (eng. *sparse matrix*) u kompresovanom CSR (eng. *Compressed Sparse Row*) formatu. Retkim ili proređenim matricama se nazivaju matrice koje imaju veliki broj nula elemenata. Umesto čuvanja cele matrice pamte se samo indeksi i vrednosti ne-nula elemenata matrice. U realnim primenama kada se radi sa matricama velikih dimenzija ovo svojstvo je izuzetno poželjno kako po pitanju zauzeća memorije tako i vremena izvršavanja računskih operacija. Metodom `scipy.sparse.csr_matrix.toarray` matrice se mogu razviti iz kompresovanog formata u pun oblik. Povratna vrednost ovog metoda je `numpy.array`, što je standardni tip podataka za čuvanje (višedimenzionih) nizova.

Generator objekat je objekat koji na zahtev vraća deo po deo nekog skupa podataka. On se kreira generator funkcijom koja se vrti u beskonačnoj petlji i iterira po skupu podataka iz kojeg se uzimaju delovi. Generatori se koriste kada je skup podataka previše veliki da se učita u RAM memoriju, a kako se neuronskoj mreži podaci ionako ne prosleđuju odjednom, već u paketima (`batch`-evima), tako se mogu i podaci učitavati u RAM memoriju deo po deo.

Generator podataka za rekurentni deo mreže

Definišemo generator funkciju koja vraća iterator po skupu podataka koji iz kojeg uzima delove na zahtev. Taj iterator kasnije prosleđujemo modelu neuronske mreže, koji posredstvom tog iteratora uzima iz skupa podataka paket po paket. U ovom slučaju mreži je potrebno proslediti `numpy` nizove, a ne `sparce` matrice, pa ćemo u okviru generatora uzimati po paket podataka i prebaciti ga iz kompresovanog u pun oblik. Takođe, potrebno je da dužine sekvenci u okviru istog paketa budu iste[18] pa ćemo izvrsiti takozvani *zero padding*. Funkcijom `pad_sequence` iz paketa `preprocessing.sequence` biblioteke `keras`, sekvence se dopunjuju do potrebne dužine nula-vektorima dimenzije 20. Naredni segment koda definiše opisanu generator funkciju.

[18]Kao što je već rečeno, kroz neuronsku mrežu se podaci propuštaju paket po paket. Kod rekurentnih mreža razmotavanje mreže se vrši za svaki paket posebno, pa iz tog razloga dužine sekvenci u okviru istog paketa moraju biti iste, dok se na različitim paketima mogu razlikovati.

```
def generator_za_rnn(x, y, batch_size, shuffle=False):
  i = 0

  while True:
    if shuffle==True:
      row_indexes = np.random.randint(low=0, high=len(x), size=batch_size)
    else:
      i = i % len(x)

      if i+batch_size < len(x):
        row_indexes = np.arange(i, i+batch_size)
      else:
        row_indexes = np.hstack(np.arange(i, len(x)),
                                np.arange(0, batch_size - (len(x)-i)))

    x_batch = []
    y_batch = []

    for j, row_index in enumerate(row_indexes):
      x_batch.append(x[row_index].toarray())
      y_batch.append(y[row_index])

    x_batch = pad_sequences(x_batch)

    i += batch_size

    yield np.array(x_batch), np.array(y_batch)
```

Generator podataka za potpuno povezani deo mreže

Analogno, za rekurentni deo mreže definišemo funkciju kojom se kreira generator skupa funkcionalnih domena. Kako je neuronskoj mreži potrebno proslediti `numpy` nizove, a ne `sparce` matrice, u okviru generatora se uzima sledeći paket podataka i prebacuje iz kompresovanog u pun oblik. Naredni segment koda definiše opisanu generator funkciju.

```
def generator_za_dense(x, y, batch_size, shuffle=False):
  i = 0

  while True:
    if shuffle==True:
      row_indexes = np.random.randint(low=0, high=len(x), size=batch_size)
    else:
      i = i % len(x)

      if i+batch_size < len(x):
        row_indexes = np.arange(i, i+batch_size)
      else:
        row_indexes = np.hstack(np.arange(i, len(x)),
                                np.arange(0, batch_size - (len(x)-i)))

    x_batch = np.zeros((batch_size, x[0].shape[1]))
    y_batch = np.zeros((batch_size, y[0].shape[0]))

    for j, row_index in enumerate(row_indexes):
      x_batch[j] = x[row_index].toarray()
      y_batch[j] = y[row_index]

    i += batch_size

    yield x_batch, y_batch
```

Pomoću ovih funkcija prave se generatori odvojeno za trening, validacioni i test skup. Generatore skupova za trening i validaciju prosleđujemo `fit_generator` metodu modela neuronske mreže, a generator test skupa metodu `evaluate_generator`.

5.2 Funkcija greške za hijerarhijsku klasifikaciju

Kao što je već rečeno, prilikom obučavanja neuronske mreže na kraju svake epohe izračunava se vrednost funkcije greške i na osnovu nje se ažuriraju parametri mreže. Kako su klase enzima hijerarhijski organizovane, neke klase su srodnije od drugih, pa je potrebno definisati funkciju greške koja će manje penalizovati pogrešnu klasifikaciju između tih klasa nego pogrešnu klasifikaciju između klasa koje nisu srodne.

Keras biblioteka omogućava definisanje sopstvene funkcije greške koja se zatim na standardni način pridružuje modelu kroz metod `compile`. Potrebno je da funkcija greške zadovoljava sledeće uslove:

- Funkcija greške treba da uzima dva argumenta - stvarne vrednosti `y_true` i predviđanja modela `y_pred`; takođe, potrebno je da funkcija ima mogućnost prihvatanja eventualnih dodatnih imenovanih argumenata, što se postiže argumentom `**kwargs`.
- Kako se neuronskoj mreži u toku obučavanja podaci prosleđuju u paketima, tako će i funkciji greške podaci biti prosleđivani u paketima, što znači da će promenljive `y_true` i `y_pred` koje funkcija greške prihvata biti dimenzija `batch_size` $\times$ `dimenzija_izlaza_mreže` i da je potrebno da povratna vrednost funkcije greške bude niz grešaka na pojedinačnim instancama, odnosno promenljiva dimenzije `batch_size` $\times$ `1`.
- Promenljive `y_true` i `y_pred` su Tensorflow tenzori i treba u skladu sa time baratati sa njima.
- Prilikom izračunavanja greške potrebno je da funkcija uzima u obzir predviđanja modela, odnosno koristi vrednosti iz promenljive `y_pred`, jer će u suprotnom gradijent funkcije greške biti nedefinisan i doći će do greške prilikom koraka optimizacije.

Kako su za klasifikaciju na drugom nivou hijerarhije instancama kao labele pridruženi vektori dimenzije 68 (podsećanja radi, to je broj klasa na drugom nivou hijerarhije), dimenzije promenljivih `y_true` i `y_pred` će biti `batch_size` $\times$ `68`. Stvarne vrednosti labela su vektori koji imaju sve nule i jednu jedinicu na indeksu klase sa drugog nivoa hijerarhije, dok su kod labela koja dobijamo kao predviđanja neuronske mreže to vektori verovatnoća pripadnosti svakoj od klasa sa drugog nivoa hijerarhije. Ovim vektorima su implicitno određene i stvarne vrednosti, kao i predviđanja modela na prvom nivou hijerarhije. Preslikavanje `EC1_n_classes` svakoj klasi sa prvog nivoa hijerarhije pridružuje broj njenih podklasa i na osnovu njega se lako određuju intervali indeksa vektora labela koji odgovaraju klasama sa prvog nivoa hijerarhije. Stvarne i predviđene klase na prvom nivou hijerarhije se izračunavaju sumiranjem elemenata vektora labela na indeksima u odgovarajućim intervalima. Zatim se na osnovu stvarnih i predviđenih labela za prvi i na drugi nivo hijerarhije izračunava zbir vrednosti funkcije kategoričke unakrsne entropije na svakom od nivoa. Ovako definisana funkcija greške će davati manju vrednost ukoliko je instanca dobro klasifikovana na prvom nivou hijerarhije ali joj je pridružena pogrešna podklasa na drugom nivou hijerarhije, što je upravo željeno ponašanje koje manje penalizuje pogrešnu klasifikaciju između klasa koje su srodne. Naredni segment koda definiše opisanu funkciju greške.

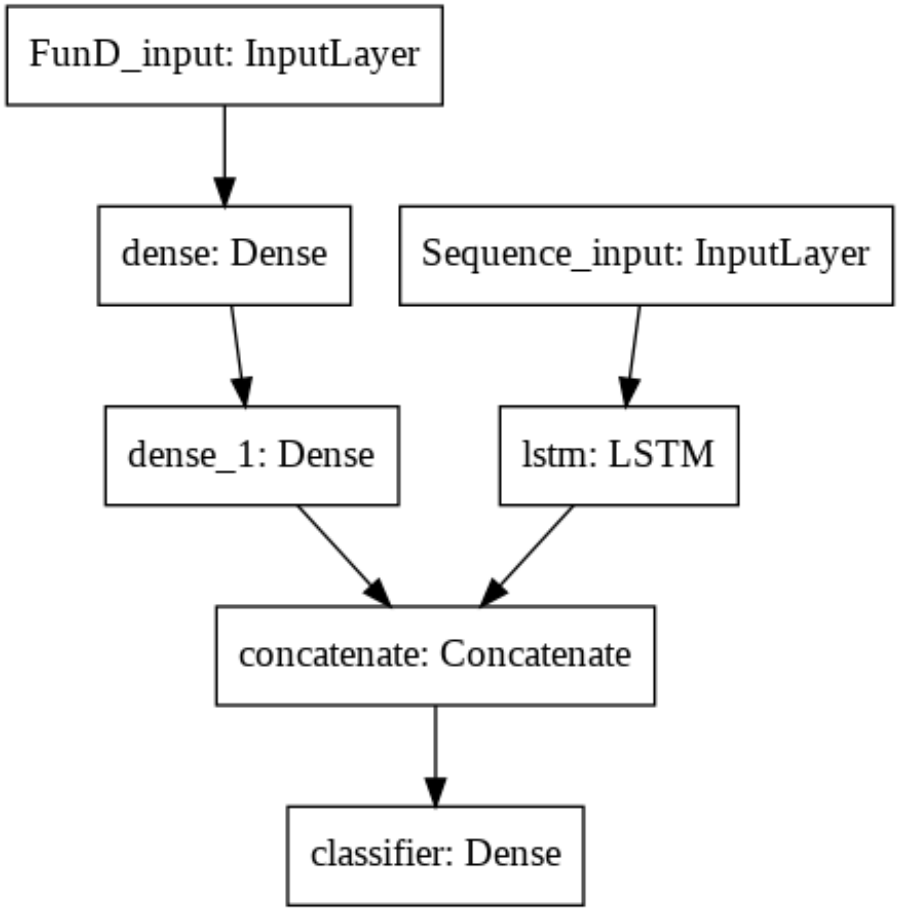
```
def custom_categorical_crossentropy(y_true, y_pred, **kwargs):

    batch_size = y_true.shape[0]

    y_true_level2 = y_true

    y_pred_level2 = y_pred / kb.sum(y_pred, axis=-1, keepdims=True)
    y_pred_level2 = tf.clip_by_value(y_pred_level2, 1e-7, 1. - 1e-7)

    y_true_level1 = kb.sum(y_true_level2[:, 0:0+EC1_n_subclasses[1]],
                           axis=-1, keepdims=True)
    index = EC1_n_subclasses[1]

    for i in range(2, n_classes_level1+1):
        p_i = kb.sum(y_true_level2[:, index:index+EC1_n_subclasses[i]],
                     axis=-1, keepdims=True)
        y_true_level1 = kb.concatenate((y_true_level1, p_i), axis=-1)
        index += EC1_n_subclasses[i]

    y_pred_level1 = kb.sum(y_pred_level2[:, 0:0+EC1_n_subclasses[1]],
                           axis=-1, keepdims=True)
    index = EC1_n_subclasses[1]

    for i in range(2, n_classes_level1+1):
        p_i = kb.sum(y_pred_level2[:, index:index+EC1_n_subclasses[i]],
                     axis=-1, keepdims=True)
        y_pred_level1 = kb.concatenate((y_pred_level1, p_i), axis=-1)
        index += EC1_n_subclasses[i]

    loss_level1 = kb.sum(-y_true_level1 * kb.log(y_pred_level1),
                         axis=-1, keepdims=False)

    loss_level2 = kb.sum(-y_true_level2 * kb.log(y_pred_level2),
                         axis=-1, keepdims=False)

    loss = loss_level1 + loss_level2

    return loss
```

5.3 Arhitektura neuronske mreže

Arhitektura neuronske mreže je konstruisana po ugledu na referentni rad [1]. Neuronska mreža se sastoji iz dva dela - potpuno povezanog i rekurentnog, koji vrše ekstrakciju atributa iz odgovarajućih ulaznih podataka. Ulaz rekurentnog dela mreže su *one-hot* reprezentacije sekvenci enzima, dok su funkcionalni domeni, odnosno njihove vektorske reprezentacije, ulaz potpuno povezanog dela mreže. Arhitektura neuronske mreže je definisana na sledeći način:

* Potpuno povezani deo mreže se sastoji od dva potpuno povezana sloja sa po 1024 ćelije i *rlu* aktivacionom funkcijom.
* Rekurentni deo mreže se sastoji od jednog $LSTM$ sloja sa 256 ćelija i podrazumevanom aktivacionom funkcijom *tanh*.
* Izlazi rekurentnog i potpuno povezanog dela mreže se konkateniraju (256+1024 atributa).
* Nad ekstrahovanim atributima vrši se klasifikacija jednim potpuno povezanim slojem sa *softmax* aktivacionom funkcijom i 7 izlaza u slučaju klasifikacije na prvom nivou hijerarhije, odnosno 68 izlaza za klasifikaciju na drugom nivou hijerarhije.

Na slici 5.15 je dat vizuelni prikaz arhitekture mreže. Za konstrukciju neuronske mreže korišćen je *funkcionalni* API biblioteke Keras. Ova vrsta API-ja se koristi u slučajevima kada se konstruišu nesekvencijalni modeli. `Input` i `Output` slojevi predstavljaju Tensorflow tenzore, dok svi drugi slojevi predstavljaju funkcije koje kao ulaz primaju Tensorflow tenzore i kao rezultat generišu takođe Tensorflow tenzore. Izlazi slojeva su funkcije, tj. *callable* objekti.



Slika 5.15: Arhitektura neuronske mreže

Glava 6

6 Eksperimentalna evaluacija

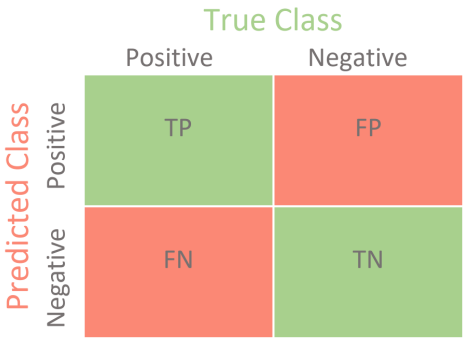
U ovom poglavlju biće opisane mere kvaliteta koje su korišćene za evaluaciju modela, kao i još neke tehnike korišćene prilikom konstrukcije nekoliko varijanti prethodno opisane osnovne arhitekture neuronske mreže. Uz svaku od isprobanih varijanti biće dati rezultati evaluacije modela i njihova međusobna poređenja. Poređenje dobijenih rezultata sa rezultatima iz referentnog rada [1] nije bilo moguće zato što se razlikuju korišćeni skupovi podataka. Pored toga, rezultati nisu predstavljeni tabelarno već samo grafički. Ono što se može reći na osnovu dostupnih grafika jeste da su rezultati veoma sličnog reda veličine.

6.1 Mere kvaliteta modela za problem klasifikacije

Evaluacija modela predstavlja kvantifikaciju njegove sposobnosti predviđanja. Ona počiva na *merama kvaliteta* i *tehnikama evaluacije modela*. O tehnici evaluacije pomoću skupova za validaciju i testiranje koja je korišćena u ovom radu bilo je reči u odeljku 5.1, a na ovom mestu će biti navedene i opisane korišćene mere kvaliteta modela.

Mere koje se najčešće koriste za klasifikaciju su *tačnost* (eng. *accuracy*), *preciznost* (eng. *precission*), *odziv* (eng. *recall*) i *F1 mera*. Ove mere će zbog jednostavnosti biti definisane u terminima binarne klasifikacije. Za slučaj višeklasne klasifikacije ove mere se izračunavaju za svaku od klasa pojedinačno po principu *one-vs-rest*.

Sve četiri mere kvaliteta se zasnivaju na *matrici konfuzije* (eng. *confusion matrix*) u kojoj se sumiraju rezultati testiranja klasifikatora. To je matrica C čiji element c_{ij} predstavlja broj instanci klase i koje su klasifikovane u klasu j. Elementi na glavnoj dijagonali predstavljaju broj instanci koje su ispravno klasifikovane, dok nedijagonalni elementi označavaju greške. To znači da je klasifikacija najbolja kada je matrica konfuzije dijagonalna. U slučaju binarne klasifikacije uobičajeno je da se jedna klasa naziva *pozitivnom*, a druga *negativnom*. Tada matrica konfuzije ima karakterističan oblik prikazan na slici 5.16.



Slika 6.16: Matrica konfuzije

Stvarno pozitivne (eng. *true positive*) instance su instance iz pozitivne klase za koje je predviđena klasa takođe pozitivna. Broj takvih instanci označen je sa *TP*. *Stvarno negativne* (eng. *true negative*) instance su negativne instance za koje model predviđa negativnu klasu. Broj takvih instanci označen je sa *TN*. *Lažno pozitivne* (eng. *false positive*) instance su instance negativne klase a koje je model prepoznao kao pozitivne. Broj takvih instanci označen je sa *FP*. *Lažno negativne* (eng. *false negative*) instance su pozitivne instance za koje model predviđa negativnu klasu. Broj takvih instanci je označen sa *FN*.

Tačnost predstavlja ocenu ukupnog broja uspešno klasifikovanih instanci i izračunava se po sledećoj formuli:

$$Acc = \frac{TP + FN}{TP + TN + FP + FN}$$

Ova mera je obično prvi izbor kod problema klasifikacije, međutim ona ne daje uvek dobru ocenu modela. U slučaju nebalansiranih klasa[19] model koji uvek (ili skoro uvek) predviđa većinsku klasu imaće visoku tačnost na datom skupu podataka, a da pritom nije naučio ništa iz podataka (osim koja je klasa većinska). Takav model ima izuzetno lošu sposobnost predviđanja i iz tog razloga se pri radu sa nebalansiranim skupom podataka ova mera nikada ne koristi ili se koristi zajedno sa još nekom merom koja nije osetljiva na nebalansiranost klasa.

Preciznost ocenjuje koliko je pozitivnih instanci ispravno klasifikovano u odnosu na ukupan broj instanci koje model klasifikuje kao pozitivne. Sa druge strane, **odziv** predstavlja udeo ispravno klasifikovanih pozitivnih instanci u odnosu na ukupan broj pozitivih instanci u skupu. Ove dve mere definisane su sledećim formulama:

$$Prec = \frac{TP}{TP + FP} \qquad Rec = \frac{TP}{TP + FN}$$

Preciznost i odziv pojedinačno posmatrane nisu korisne mere [18]:

- visoka preciznost a nizak odziv - model retko klasifikuje instance kao pozitivne, i samim tim još manje pravi grešku da neku instancu koja je negativna proglasi za pozitivnu
- visok odziv a niska preciznost - model često klasifikuje instance kao pozitivne, i time retko za neku instancu koja je pozitivna predviđa negativnu klasu

Iz navedenih razloga se ove dve mere uvek posmatraju zajedno. Način na koji se to najčešće radi je tako što se izračuna **F1 mera** - njihova harmonijska sredina:

$$F1 = 2\frac{Prec \cdot Rec}{Prec + Rec}$$

Treba imati u vidu da su preciznost, odziv i F1 mera definisane u odnosu na pozitivnu klasu, odnosno da nisu simetrične u odnosu na izbor neke klase kao pozitivne. Sumarni izvestaj vrednosti ovih mera po klasama (uzimanjem ponaosob svake klase za pozitivnu klasu) može se dobiti funkcijom `classification_report` iz biblioteke

[19] Kada u skupu podataka postoji mnogo više instanci jedne klase u odnosu na broj instanci koje pripadaju drugoj klasi.

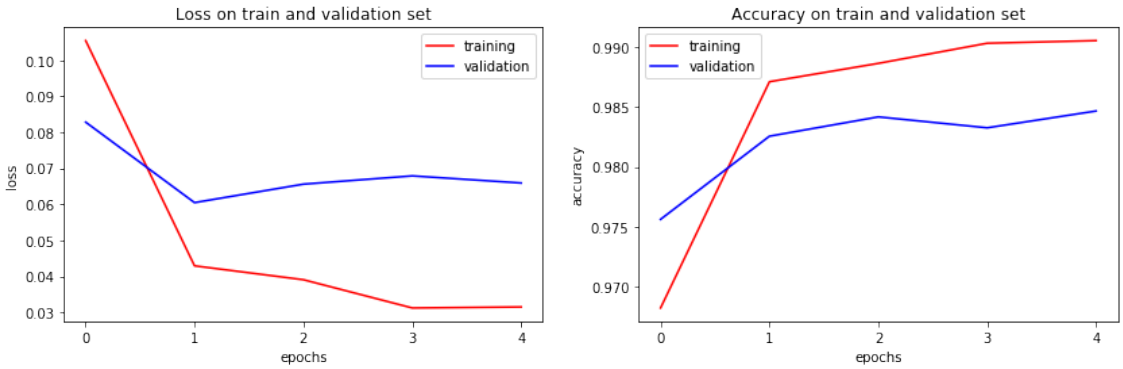
`metrics`. Takođe, u biblioteci `metrics` se nalaze funkcije koje računaju ove četiri mere, kao i razne druge mere kvaliteta, kako za problem klasifikacije, tako i za problem regresije. Matrica konfuzije se može dobiti pozivom funkcije `confusion_matrix` biblioteke `metrics`.

6.2 Klasifikacija na prvom nivou hijerarhije

Za prvi nivo hijerarhije isprobane su 4 varijante osnovne arhitekture neuronske mreže uz korišćenje tehnika ranog zaustavljanja, pridruživanja težina instancama, unutrašnje standardizacije podataka na nivou paketa i maskiranja određenih izlaza rekurentnog sloja. U nastavku sledi kratak opis svake od navedenih tehnika zajedno sa rezultatima evaluacije odgovarajućih modela.

Rano zaustavljanje

Keras biblioteka nudi mogućnost da se pri treniranju mreže nakon svake epohe pozovu određene funkcije - *callbacks*. Ova funkcionalnost je korišćena za primenu tehnike ranog zaustavljanja (eng. *early stopping*) i čuvanje modela (težina koeficijenata) nakon svake epohe. Tehnika ranog zaustavljanja je vrsta tehnika regularizacije neuronskih mreža i ona podrazumeva da se u slučaju neopadanja funkcije greške kroz nekoliko epoha zaustavi sa treniranjem mreže.



Slika 6.17: Obučavanje mreže - funkcija greške i tačnost

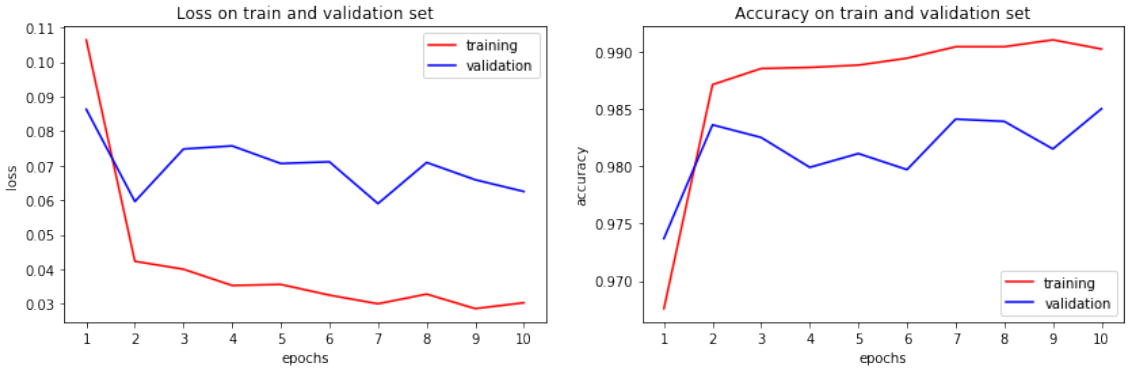
	precision	recall	f1-score	support
1.0	0.99	0.99	0.99	8673
2.0	0.99	0.99	0.99	26275
3.0	0.99	0.97	0.98	12798
4.0	0.98	0.98	0.98	7321
5.0	0.95	0.97	0.96	4417
6.0	0.99	1.00	0.99	8580
7.0	0.99	0.99	0.99	2016
accuracy			0.99	70080
macro avg	0.98	0.98	0.98	70080
weighted avg	0.99	0.99	0.99	70080

Tabela 6.1: Mere kvaliteta na test skupu

Pridruživanje težina instancama

Nebalansirani skupovi podataka podrazumevaju da klase nisu ravnopravne po broju broju instanci, tj. da u skupu postoji mnogo više instanci jedne ili više klasa u odnosu na broj instaci koje pripadaju ostalim klasama. Kako se algoritmi mašinskog učenja zasnivaju na minimizaciji greške na datom skupu podataka, u slučaju velike nebalansiranosti podataka će se prilikom obučavanja zaključiti da je mnogo veća verovatnoća da instanca pripada većinskoj klasi (jednoj ili više njih) što rezultuje modelom koji pri predviđanju favorizuje određenu klasu (ili više njih).

Jedan od pristupa problemu nebalansiranih skupova podataka jeste tehnika pridruživanja težina instancama (eng. *weighting*). Težine se biraju tako da budu obrnuto proporcionalne kardinalnosti klasa i se koriste prilikom računanja funkcije greške kako bi se greška na instancama manjinskih klasa više kažnjavala nego greška na instancama većinske klase, i time ublažio uticaj nebalansiranosti klasa na proces učenja.



Slika 6.18: Obučavanje mreže - funkcija greške i tačnost

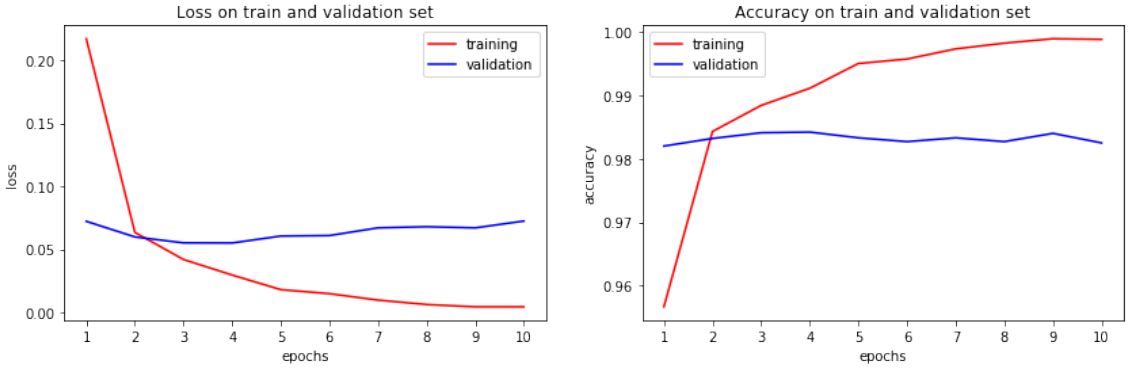
Kao najbolji model uzet je model nakon 7 epoha treninga i on je evaluiran na test skupu.

	precision	recall	f1-score	support
1.0	0.99	0.99	0.99	8673
2.0	0.99	0.99	0.99	26275
3.0	0.99	0.97	0.98	12798
4.0	0.98	0.98	0.98	7321
5.0	0.95	0.97	0.96	4417
6.0	0.99	1.00	0.99	8580
7.0	0.99	0.99	0.99	2016
accuracy			0.99	70080
macro avg	0.98	0.98	0.98	70080
weighted avg	0.99	0.99	0.99	70080

Tabela 6.2: Mere kvaliteta na test skupu

Unutrašnja standardizacija podataka na nivou paketa

Poznato je da algoritmi mašinskog učenja imaju bolja računska svojstva kada se sve promenljive (atributi skupa podataka) nalaze na istoj skali. Standardizacijom datog skupa podataka se lako postiže da ovo svojstvo važi za ulaze neuronske mreže. Kao što je poželjno da ulazi mreže, odnosno ulazi prvog sloja mreže, budu standardizovani, iz istog razloga je poželjno da i ulazi svih skrivenih slojeva budu standardizovani. To se postiže tehnikom unutrašnje standardizacije podataka na nivou paketa (eng. *batch normalization*). Ova tehnika se sastoji u tome da se na nivou pojedinačnih paketa podataka, koji se jedan po jedan propuštaju kroz neuronsku mrežu, vrši standardizacija svake izlazne jedinice sloja.



Slika 6.19: Obučavanje mreže - funkcija greške i tačnost

Kao najbolji model uzet je model nakon 4 epohe treninga i on je evaluiran na test skupu.

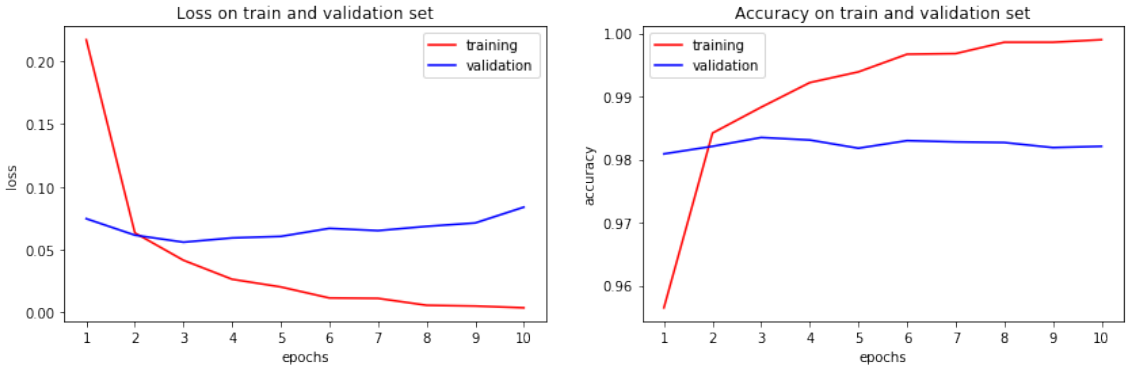
	precision	recall	f1-score	support
1.0	0.99	0.99	0.99	8673
2.0	0.99	0.99	0.99	26275
3.0	0.98	0.98	0.98	12798
4.0	0.97	0.98	0.97	7321
5.0	0.97	0.96	0.96	4417
6.0	0.99	1.00	0.99	8580
7.0	0.99	0.99	0.99	2016
accuracy			0.98	70080
macro avg	0.98	0.98	0.98	70080
weighted avg	0.98	0.98	0.98	70080

Tabela 6.3: Mere kvaliteta na test skupu

Maskiranje određenih izlaza rekurentnog sloja

Kako su sekvence enzima različitih duzina, bilo je potrebno da u okviru istog paketa budu dopunjene nula-vrednostima do potrebne dužine. Prema tome, poželjno je da se kao izlaz rekurentnog sloja uzima izlaz na onom mestu gde je poslednja

ne-nula vrednost sekvence obrađena. To se postiže tehnikom maskiranja izlaza rekurentnog sloja (eng. *masking*). Ova tehnika se sastoji u tome da se prepozna prva nula-vrednost na ulazu rekurentnog sloja i da se izlazi sloja počev od tog ulaza pa do kraja sekvence 'zamaskiraju', odnosno ignorišu, kako bi kao poslednji izlaz rekurentnog sloja bila zapamćena vrednost izlaza za poslednji ne-nula element sekvence.



Slika 6.20: Obučavanje mreže - funkcija greške i tačnost

Kao najbolji model uzet je model nakon 3 epohe treninga i on je evaluiran na test skupu.

	precision	recall	f1-score	support
1.0	0.99	0.99	0.99	8673
2.0	0.99	0.99	0.99	26275
3.0	0.99	0.98	0.98	12798
4.0	0.97	0.98	0.97	7321
5.0	0.95	0.97	0.96	4417
6.0	0.99	1.00	0.99	8580
7.0	0.99	0.99	0.99	2016
accuracy			0.98	70080
macro avg	0.98	0.98	0.98	70080
weighted avg	0.98	0.98	0.98	70080

Tabela 6.4: Mere kvaliteta na test skupu

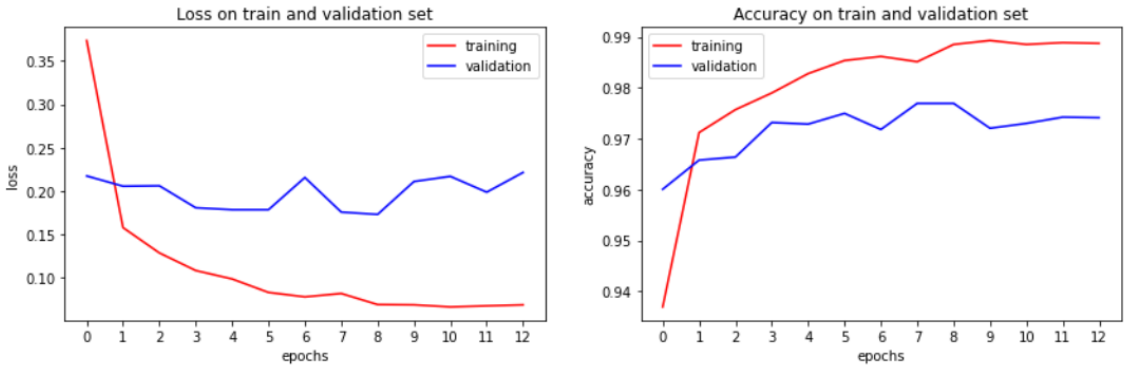
U tabeli 6.5 su prikazane vrednosti funkcije greške i tačnosti na test skupu sa većom preciznosti kako bi se mogle uočiti suptilne razlike između ova 4 modela.

	Early Stopping	Weighting	Batch Normalization	Masking
loss	0.0654	0.0656	0.0567	0.0518
accuracy	0.9841	0.9849	0.9841	0.9854

Tabela 6.5: Vrednosti funkcije greške i tačnosti na test skupu

6.3 Klasifikacija na drugom nivou hijerarhije

Zbog znatnog dužeg vremena obučavanja mreže a zanemarljivog poboljšanja kvaliteta modela, za klasifikaciju na drugom nivou hijerarhije odustalo od svih tehnika koje su isprobavane na prvom nivou hijerarhije.



Slika 6.21: Obučavanje mreže - funkcija greške i tačnost

Vrednost funkcije greške na test skupu je 0.2088, dok je tačnost klasifikacije na test skupu 0.9752.

Glava 7

7 Zaključak

U ovom radu prikazan je pristup koji kombinuje dve vrste lokalnih strukturnih svojstva enzima i dve vrste neuronskih mreža koje na osnovu njih konstruišu nove atribute i vrše predikciju funkcije enzima pridruživanjem odgovarajućeg EC broja. Korišćene su sekvence enzima, kao lokalne informacije o strukturi enzima, i funkcio- nalni domeni kao lokalne informacije o funkciji enzima. Za izgradnju modela korišćene su neuronske mreže koje se sastoje iz dva dela - potpuno povezanog i rekurentnog. Za klasifikaciju na prvom nivou hijerarhije konstruisano je četiri varijante osnovnog modela, ali se pri evaluaciji nijedan od njih nije izdvojio kao značajno bolji od ostalih. Zbog znatnog dužeg vremena obučavanja mreže a zanemarljivog poboljšanja kvaliteta modela, za kvalifikaciju na drugom nivou se odustalo od dodatnih tehnika koje su isporbavane na prvom nivou hijerarhije i konstruisana je samo osnovna varijanta modela.

Planovi za unapređivanje i nadogradnju modela uključuju:

* Dodavanje klasifikatore za treći i četvrti nivo hijerarhije.
* Isprobavanje više metoda za rad sa nebalansiranim skupom podataka, pošto problem postaje sve izraženiji na trećem i četvrtom nivou hijerarhije.
* Isprobavanje sa dodavanjem još nekih vrsta atributa, kao što je npr. PSSM matrica.
* Evaluaciju modele nad COFACTOR bazom enzima koja se pokazala kao teška za predikciju funkcije enzima.

Literatura

[1] Yu Li, Sheng Wang, Ramzan Umarov, Bingqing Xie, Ming Fan, Lihua Li, Xin Gao, *DEEPre: sequence-based enzyme EC number prediction by deep learning*, Bioinformatics, Volume 34, Issue 5, 01 March 2018, Pages 760–769

[2] Rick Ricer Michael A. Lieberman, *Biochemistry, Molecular Biology, and Genetics*, New Science Press Ltd, 2004.

[3] *Glossary of Genetics Terms, National Human Genome Research Institute.* on-line na: https://www.genome.gov/genetics-glossary/Amino-Acids?id=5

[4] *BIOCHEMISTRY - DEFINING LIFE AT THE MOLECULAR LEVEL, Online Chemistry Textbooks, Western Oregon University*, on-line na: https://wou.edu/chemistry/courses/online-chemistry-textbooks/ch450-and-ch451-biochemistry-defining-life-at-the-molecular-level/chapter-2-pr otein-structure/

[5] *Glossary of nanotechnology and related terms, Russian Corporation of nanotechnology*, on-line na: https://eng.thesaurus.rusnano.com/wiki/article561

[6] Malgosia Wilk-Blaszczak, *Four levels of protein structure - Cell Physiology*, on-line na: https://uta.pressbooks.pub/cellphysiology/chapter/chapter-2/

[7] *A schematic drawing to use tree branches to classify enzyme and non-enzyme as well as the six main functional classes of enzymes and their subclasses, ResearchGate* on-line na: https://www.researchgate.net/figure/A-schematic-drawing-to-use-tree-branches-to-classify-enzyme-and-non-enzyme-as-well-as-the_fig4_228993537

[8] Marcelo Boareto, Michel E.B.Yamagishi, NestorCaticha, Vitor B.P.Leite, *Relationship between global structural parameters and Enzyme Commission hierarchy: Implications for function prediction*, Computational Biology and Chemistry, Volume 40, October 2012, Pages 15-19

[9] S. Horiguchi, D. Ikami and K. Aizawa, *Significance of Softmax-Based Features in Comparison to Distance Metric Learning-Based Features*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 5, pp. 1279-1285, 1 May 2020, doi: 10.1109/TPAMI.2019.2911075.

[10] Mladen Nikolić, Predrag Janičić, *Veštačka inteligencija.* on-line na: http://poincare.matf.bg.ac.rs/~janicic//courses/vi.pdf

[11] Mladen Nikolić, Anđelka Zečević, *Mašinsko učenje*, on-line na: http://ml.matf.bg.ac.rs/readings/ml.pdf

[12] Jovana Kovačević, *Strukturna predikcija funkcije proteina i odnos funckionalnih kategorija i neuređenost*, Matematički fakultet, 2015.

[13] F. Chollet, *Deep Learning With Python*, New York, NY, USA: Manning Publications Co, 2017

[14] Jae Duk Seo, *DNA / Protein Representation for Machine Learning Task with interactive code*, on-line na: https://towardsdatascience.com/dna-protein-representation-for-machine-learning-task-with-interactive-code-6a a065b69227

[15] *Recurrent Neural Networks (RNNs) - Implementing an RNN from scratch in Python, Towards Data Science* on-line na: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

[16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep learning*, Cambridge, Massachusetts : The MIT Press, 2016

[17] *How to Create a Custom Loss Function in Keras.* on-line na: https://towardsdatascience.com/how-to-create-a-custom-loss-function-keras-3a89156ec69b

[18] Baptiste Rocca, *Handling imbalanced datasets in machine learning.* on-line na: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28