

UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

MASTER RAD

Predviđanje funkcija proteina primenom grafovske neuronske mreže

Autor:

Stefan SPALEVIĆ

Mentor:

doc. dr Mladen NIKOLIĆ

15. septembar 2020



Zahvalnica i posveta

Zahvaljujem se mentoru profesoru Mladenu Nikoliću i profesorki Jovani Kovačević na saradnji, savetima, detaljnim komentarima, posvećenom vremenu, podršci i entuzijazmu koji prenose na studente. Takođe, zahvaljujem se profesorki Zorici Stanimirović na svemu tome, kao i na dugogodišnjoj podršci u svakom smislu tokom osnovnih i master studija.

Saradnja sa Petrom Veličkovićem mi je puno pomogla u izradi ovog rada, pa mu se ovom prilikom najsrdačnije zahvaljujem. Za kratko vreme Petar je, pored profesora Mladena, postao jedan od mojih uzora u oblasti mašinskog učenja. Za moju ljubav prema matematici su najzaslužniji moj tata, Đorđe Baralić i Mirjana Katić.

Posvećujem master rad svojoj porodici i Milici Galjak. Hvala vam na svemu!

Sadržaj

1	Uvod	1
2	Osnovni grafovski pojmovi	4
2.1	Grafovi	4
2.2	Usmereni grafovi	5
2.3	Matrica susedstva i Laplasova matrica grafa	6
3	Relevantni bioinformatički pojmovi	7
3.1	Biološki pojmovi	7
3.1.1	Aminokiseline	7
3.1.2	Proteini	7
3.1.3	Centralna dogma molekularne biologije	9
3.2	Problem predviđanja funkcija proteina	12
3.2.1	Reprezentacija funkcije proteina	12
3.2.2	Eksperimentalna anotacija	12
3.2.3	Problem predviđanja funkcija proteina	13
3.2.4	CAFA takmičenje	13
3.2.5	Evaluacija modela	14
4	Neuronske mreže	16
4.1	Kratak uvod o mašinskom učenju	16
4.2	Veštački neuron	17
4.3	Potpuno povezane neuronske mreže	18
4.4	Obučavanje neuronske mreže	20
4.4.1	Incijalizacija parametara	20
4.4.2	Optimizacija modela	21
4.4.3	Regularizacija kod neuronskih mreža	23
4.4.4	Podešavanje hiperparametara	24
4.5	Konvolutivne neuronske mreže	25
4.5.1	Konvolutivni slojevi	25
4.5.2	Agregacioni slojevi	28
4.5.3	Dilatacijske konvolucije	28
4.6	Grafovске neuronske mreže	30
4.6.1	Grafovski atributi zasnovani na slučajnom hodu	31
4.6.2	Grafovске konvolutivne mreže	31

4.6.3	Neuronske mreže sa prenošenjem poruka	32
4.6.4	Grafovske mreže sa mehanizmom pažnje	33
4.6.5	GAT sloj	33
5	Predložena arhitektura	36
5.1	Reprezentacija podataka	36
5.1.1	Ulaz neuronske mreže	36
5.1.2	Izlaz neuronske mreže	36
5.2	Predložena arhitektura	37
5.2.1	Uopšteni model	37
5.2.2	Model predviđanja funkcije proteina	37
5.3	Biblioteke korišćene u imlementaciji	40
6	Eksperimentalna evaluacija	42
6.1	Skup podataka	42
6.2	Statistika podataka i pretprocesiranje	42
6.3	Specifikacije procesa obučavanja modela	43
6.4	Eksperimenti za određivanje hiperparametara	44
6.5	Evaluacija i rezultati	46
7	Zaključak i dalji rad	48

Poglavlje 1

Uvod

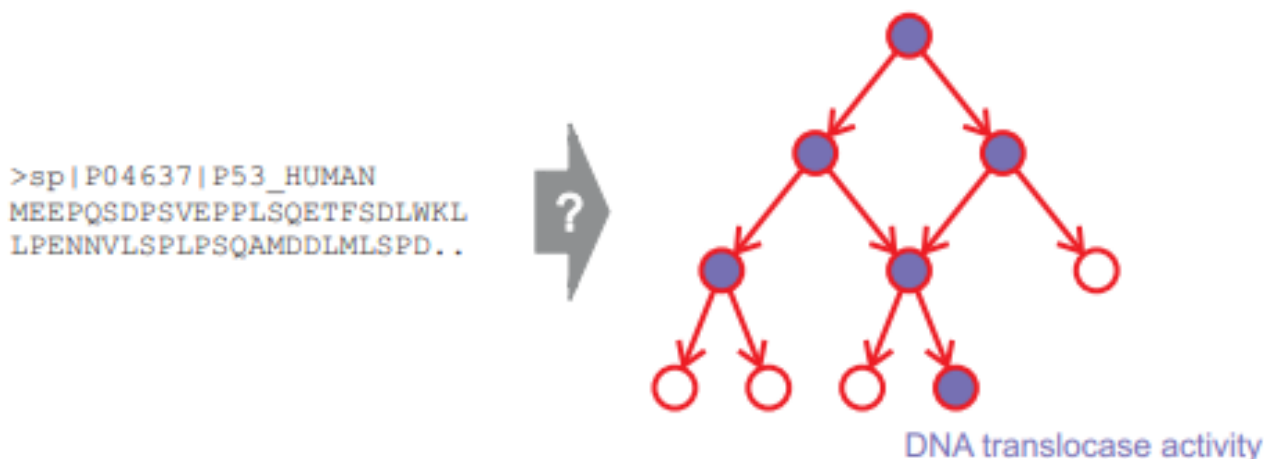
Da bismo razumeli procese u živim organizmima na molekularnom nivou, važno je da izučavamo funkcije proteina. Razne savremene bolesti nastaju usled promene funkcije izazvane mutacijama u proteinskoj sekvenci. Utvrđivanje takvih mutacija je prvenstveno zadatak biomedicine, ali u novijem dobu značajnu ulogu saveznika ove nauke igra bioinformatika.

Protein može imati više uloga, a njihovo najverodostojnije otkrivanje vrši se eksperimentalnim putem u laboratorijama pod rukovodstvom naučnika koji se bave prikupljanjem i analiziranjem bioloških podataka (biokuratora). S druge strane, eksperimenti su vremenski zahtevni, često skupi i iziskuju još puno drugih resursa što ih čini nepogonim za primenu nad velikom količinom podataka. Zbog toga je fokus naučne zajednice usmeren prema razvoju računarskih metoda za automatsko predviđanje funkcija proteina.

Cene sekvenciranja genoma su drastično opale u odnosu na 2000. godinu, kada je sekvenciran prvi humani genom što je prouzrokovalo rapidan rast broja novih proteinskih sekvenci. Podaci o proteinima se elektronski skladište u velikim biomedicinskim bazama podataka među kojima su *UniProtKB*, *Pfam*, *PDB*, *Swiss-Prot*, *TrEMBL*, itd. Broj proteina sa eksperimentalno utvrđenom funkcijom je neznatan u odnosu na ukupan broj poznatih proteinskih sekvenci, a razlika između njih postaje sve veća. Poređenja radi, trenutni broj anotacija u poznatoj bazi *Swiss-Prot* utvrđenih od strane biokuratora je oko 562 hiljade, dok je broj automatskih anotacija koje se dobijaju računarskim metodama preko 180 miliona (baza *TrEMBL*).

Može se naslutiti da je problem pogodan za metode mašinskog učenja (engl. *machine learning*) koje automatski uče zavisnosti na podacima na osnovu kojeg mogu vršiti predviđanja na novim podacima. Sa aspekta mašinskog učenja, postoje i drugi problemi koji strukturno liče na ovaj, pa je njegovo rešavanje relevantno i iz te perspektive.

U poslednje dve decenije, razvijeno je mnoštvo metoda za rešavanje ovog problema. Neke su zasnovane na poređenju proteinskih sekvenci sa sekvencama koje su već eksperimentalno izučavane, neke koriste razne proteinske strukture, fizičko-hemijska svojstva, makromolekularne interakcije, naučnu literaturu, okruženje u kojima protein intereaguje, 3D strukturu ili kombinaciju nekoliko vrsta podataka. Kako se baze podataka proteina brzo menjaju, modeli se često evaluiraju na različitim podacima, pa je bilo potrebno standardizovati evaluaciju. U tom cilju, od 2010. godine se organizuju *CAFA* takmičenja (engl. *Critical Assessment of Function Annotation*) [2] na kojima timovi iz celog sveta šalju modele koji predviđaju funkcije proteina. Evaluacija se vrši na skupu koji se formira od anotacija proteina koje se eksperimentalno utvrde tek nakon što se rok za predaju prediktora završi. Zbog toga, nijedan tim nema informacije o



Slika 1.1: Podgraf ontologije kao skup funkcija za neki protein [1].

funkcijama proteina za skup sekvenci na kome će se testirati modeli. U trenutku pisanja ovog rada, četiri *CABA* takmičenja su organizovana, dok je peto u toku. *CABA* takmičenja prate i opšti trend poboljšanja metoda za ovaj problem.

Funkcije proteina su međusobno zavisne i predstavljaju se kao čvorovi u hijerarhijskom usmerenom acikličnom grafu koji nazivamo ontologija (engl. *ontology*). Grane u ovom grafu oslikavaju zavisnost funkcija tako što potomci čvora predstavljaju dalju specijalizaciju funkcije. Skup funkcija proteina predstavlja *konzistentni* podgraf ontologije, što znači da ako je neki čvor proglašen za funkciju nekog proteina, onda su i svi preci tog čvora takođe u skupu funkcija tog proteina (slika 1.1). Određivanje svih funkcija proteina u mašinskom učenju predstavlja problem *višestruke klasifikacije* (engl. *multi-label classification*) [1].

Specifičnost reprezentacije funkcije proteina otvara put ka grafovskim neuronskim mrežama (engl. *graph neural networks*) [3, 4], koje su korišćene u ovom radu. Grafovske neuronske mreže se zasnivaju na principu razmenjivanja informacija između susednih čvorova. U kontekstu ovog problema se mogu koristiti tako što okolni čvorovi neke funkcije takođe utiču na odluku prediktora o tome da li je to funkcija proteina. Postoje različiti načini propagiranja informacije od strane susednih čvorova. Neki od njih su sumiranje i uzimanje maksimalanog uticaja. Postoje različite varijante grafovskih neuronskih mreža, ali većina njih podrazumeva definisanje načina skupljanja informacije koje će se prosleđivati susedu, zatim načina delovanja susedstva na čvor i kako se ažurira stanje atributa čvorova tokom iteracija. Grafovske neuronske mreže se tipično koriste u ulaznom prostoru. Jedan od glavnih doprinosa ovog rada je i to što grafovska neuronska mreža uči reprezentacije u izlaznom prostoru u kojem donosi odluke.

Kao ulazni podatak korišćena je samo sekvenca proteina. Iako postoji veliki broj karakteristika koje su mogle da budu iskorišćene kao atributi, korišćenje sirovog signala je izabrano kako bi neuronska mreža sama konstruisala attribute pogodne za predviđanje funkcije. Za dobijanje reprezentacija čvorova korišćeni su dilatacijski konvolutivni (engl. *dilated convolutions*) slojevi i potpuno povezani (engl. *fully connected*) sloj. Nad reprezentacijama i dodatnim fiksiranim atributima se dalje vrši grafovska obrada (što će biti detaljnije opisano).

Funkcionalne anotacije su predstavljene čvorovima u ontologiji GO (engl. *Gene Ontology*)[5]. Među tri ontologije, izabrana je ontologija molekularnih funkcija (*MFO*) redukovana sa 6335 na

123 čvora zbog ograničenih računarskih resursa. Time je fokus usmeren ka proveru primenljivosti grafovskih mreža na ovaj problem, a ne na dobijanju najboljih rezultata.

Preciznost modela kod predviđanja funkcija proteina je izražena kao udeo broja tačno otkrivenih proteinskih funkcija u odnosu na broj svih proteinskih funkcija koje je model predvideo za datu sekvencu. Odziv je definisan kao udeo otkrivenih funkcija proteina naspram broja svih funkcija koje taj protein ima. Kombinovanjem ove dve mere se može definisati F_1 skor kao harmonijska sredina preciznosti i odziva. Odabir jedne mere omogućava jednostavnije tumačenje uspešnosti modela, što je razlog korišćenja F_1 skora na *CAFA* takmičenjima kao i u ovom radu.

Skup proteinskih sekvenci sa eksperimentalno utvrđenim funkcijama obezbeđen od strane organizatora za treniranje modela *CAFA-3* takmičenja je uzet kao glavni skup u istraživanju. Podeljen je na skupove za obučavanje, validaciju i testiranje. U ovakvoj podeli, najbolji model je dostigao prosečan F_1 skor 0.6 na test skupu. Zbog različitih skupova za testiranje, nije moguće direktno poređenje sa rezultatima *CAFA* takmičenja. Najboljih 10 modela na *CAFA-3* takmičenju ima F_1 skor u rasponu od 0.5 do 0.6 za celu ontologiju molekularnih funkcija *MFO* [6]. Izvršeno je poređenje sa osnovnim modelom koji ne koristi grafovsku strukturu funkcija i zabeležno poboljšanje F_1 skora sa 0.584 na 0.600. Glavni doprinosi rada su uspešna primena grafovskih neuronskih mreža na problemu predviđanja funkcije proteina, što ohrabruje dalji rad na primeni ove metode na punom problemu i obradu reprezentacija u izlaznom prostoru što je retko viđeno u dosadašnjoj literaturi [7].

Poglavlje 2

Osnovni grafovski pojmovi

U ovoj glavi će ukratko biti opisani osnovni grafovski pojmovi koji se koriste u radu. Grafovska struktura je prisutna u podacima kojima se funkcije proteina predstavljaju, kao i u metodama koje su korišćene.

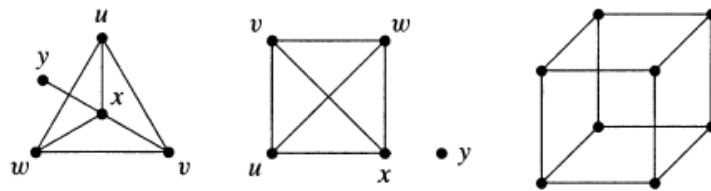
2.1 Grafovi

U matematičkom smislu, *graf* G se sastoji od skupa čvorova $V(G)$ i skupa grana $E(G)$, pri čemu svaka grana povezuje neka dva čvora. Grana može povezivati i dva ista čvora, u tom slučaju se naziva *petlja*. Ukoliko dva čvora povezuje više grana, tada kažemo da između ta dva čvora postoji *višestruka grana*. *Prost graf* je graf koji ne sadrži petlje niti višestruke grane. *Podgraf* H grafa G (u oznaci $H \subseteq G$) je graf tako da važi $V(H) \subseteq V(G)$ i $E(H) \subseteq E(G)$. Graf je *konačan* ako ima konačan broj čvorova i grana. Nadalje se podrazumeva da su svi grafovi koji se razmatraju konačni [8].

Neka su čvorovi u i v povezani nekom granom e , tada se u i v nazivaju *susedni čvorovi*. *Susedstvo* nekog čvora u je označeno sa $\mathcal{N}_u$ i predstavlja skup svih čvorova koji su susedni čvoru u . Često se i sam čvor ubraja među svoje susede.

Put je prost graf čiji se čvorovi mogu poređati u niz tako tako da za svaka dva uzastopna čvora važi da su povezani granom. Ako niz počinje iz čvora u i završava se u čvoru v , tada se ovaj put naziva u, v -put. *Cikl* je prost graf čiji čvorovi mogu biti ciklično raspoređeni tako da su svaka dva čvora susedna ako i samo ako su uzastopna u cikličnom rasporedu. Uglavnom se cikl i put posmatraju kao podgrafovi nekog grafa. Graf G je *povezan* ako za svaka dva čvora $u, v \in V(G)$ važi da postoji u, v -put u grafu G . U suprotnom kažemo da je graf *nepovezan*.

Na slici 2.1 su prikazani primeri tri jednostavna grafa. Prvi graf s leve strane (u oznaci G_1) ima čvorove $V(G_1) = \{u, v, w, x, y\}$ i grane $E(G_1) = \{uv, uw, ux, vw, vx, wx, xy\}$. Primer jednog u, y -puta u grafu G_1 je $uvwxy$, a primer jednog cikla je $xvwx$. Susedstvo čvora u je sledeći skup: $\mathcal{N}_u = \{u, v, x, w\}$. Graf u sredini G_2 nije povezan jer se iz čvora y ne može formirati nijedan put. Grafovi se mogu predstaviti i u prostoru, što pokazuje skica desnog grafa G_3 koji je predstavljen kockom [8].



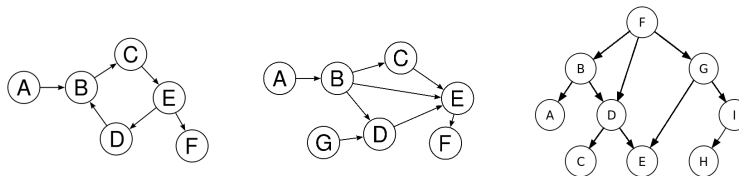
Slika 2.1: Primeri jednostavnih grafova [8].

2.2 Usmereni grafovi

Usmereni graf je graf kod kojeg svaka grana povezuje neka dva čvora, ali u utvrđenom smeru, odnosno za granu e koja povezuje neka dva čvora u i v se mora precizno naglasiti da li je grana od u ka v ili obrnuto. Smer se može naglasiti predstavljanjem grane kao uređenog para $e = (u, v)$ za $u, v \in V(G)$, pri čemu je smer iz u ka v u oznaci $u \rightarrow v$. Kod *neusmerenog grafa*, grane su određene samo izabranim parom čvorova.

Usmereni put je usmereni graf kod kojeg se čvorovi mogu poredati u redosledu tako da za svaka dva uzastopna čvora v_{i-1} i v_i važi da postoji usmerena grana $e = (v_{i-1}, v_i)$. *Usmereni cikl* predstavlja usmereni graf kod kojeg se čvorovi mogu poredati u redosledu tako da za svaka dva uzastopna čvora u i v cikličnom redosledu važi da postoji usmerena grana $e = (u, v)$. *Usmerena petlja* predstavlja granu oblika $e = (u, u)$ gde je $u \in V(G)$. Slično kao kod grafova, *prost usmeren graf* ne sadrži više grana koje povezuju isti par čvorova, kao ni petlje koje su usmerene. Kod prostih usmerenih grafova, ako za čvorove u i v u grafu G postoji usmerena grana $e = (u, v)$, tada kažemo da je čvor u *prethodnik* čvora v , a za čvor v kažemo da je *sledbenik* čvora u .

Graf se naziva *aciklički* ukoliko ne postoji cikl u grafu. *Usmeren aciklički graf* predstavlja usmeren graf bez usmerenih ciklova. U ovom radu je od interesa povezan usmereni aciklički graf kod koga postoji jedinstven čvor koji je označen kao *koreni čvor*. U usmerenom grafu sa korenom su grane orijentisane *od korena* ili *prema korenu*. U ovom radu je od značaja usmereni graf čija je orijentacija grana *od korena*. *Konzistentni podgraf* $\mathcal{H}$ usmerenog acikličkog grafa sa korenom $\mathcal{G}$ predstavlja usmeren aciklički graf tako da je $V(\mathcal{H}) \subset V(\mathcal{G})$ i $E(\mathcal{H}) \subset E(\mathcal{G})$ pri čemu koren grafa $\mathcal{H}$ je istovremeno i koren stabla $\mathcal{G}$. Na slici 2.2 se nalaze primeri usmerenog grafa, usmerenog acikličkog grafa i usmerenog acikličkog grafa sa korenim čvorom (slovo F) [8].



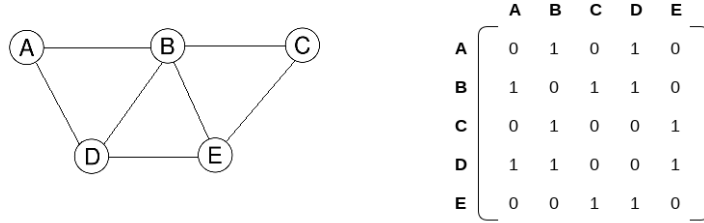
Slika 2.2: Primeri jednostavnih usmerenih grafova [9, 10].

Kod usmerenih acikličkih grafova *direktni potomak* nekog čvora $u \in V(G)$ predstavlja bilo koji čvor $v \in V(G)$ takav da postoji grana (u, v) . Ako je v direktni potomak od u , tada je u čvor roditelj (direktni predak) za čvor v . *Potomci* čvora u predstavljaju sve čvorove $v \in V(G)$

takve da postoji usmereni u, v -put. *Predak* čvora u je bilo koji čvor $v \in V(G)$ takav da postoji v, u -put.

2.3 Matrica susedstva i Laplasova matrica grafa

Kod neusmerenih grafova *stepen* čvora u predstavlja broj grana koje su povezane sa čvorom u . Kod usmerenih grafova se definišu *ulazni* i *izlazni* stepen kao broj grana koje ulaze u dati čvor odnosno broj grana koje izlazi iz datog čvora. Pretpostavimo da graf G nema višestruke grane, ali može imati petlje. Neka je $V(G) = \{v_1, v_2, \dots, v_n\}$ i $E(G) = \{e_1, e_2, \dots, e_m\}$. *Matrica susedstva* $\mathbf{A}$ za graf G predstavlja matricu dimenzija $n \times n$ čiji svaki element A_{ij} ima vrednost 1 ukoliko između i i j postoji grana, a 0 u suprotnom. Ukoliko je reč o usmerenom grafu, tada $A_{ij} = 1$ može označavati da postoji usmerena grana od i ka j , odnosno $A_{ij} = 0$ ukoliko ne postoji takva grana. Kod neusmerenih grafova matrica susedstva je simetrična. Granama se mogu dodeliti i težine, i tada $\mathbf{A} \in \mathbb{R}^{n \times n}$. U tom slučaju se graf naziva *težinskim*. Na slici 2.3 se nalazi jednostavan graf i njegova matrica susedstva koja je binarna i simetrična zbog toga što grane nisu usmerene i nisu otežane [8].



Slika 2.3: Primer grafa i njegove matrice susedstva [11].

Još jedan bitan pojam za ovaj rad je Laplasova matrica grafa Δ koja se dobija na sledeći način: $\Delta = \mathbf{D} - \mathbf{A}$, gde je $D_{ii} = \sum_j A_{ij}$ dijagonalna matrica stepena čvorova. Laplasova matrica grafa omogućava analiziranje grafa pomoću linearne algebre. Izučavanjem dekompozicije te matrice na sopstvene vrednosti i sopstvene vektore bavi se spektralna teorija grafova (engl. *spectral graph theory*) [12].

Poglavlje 3

Relevantni bioinformatički pojmovi

U ovom poglavlju će biti opisani sledeći pojmovi: građa proteina, proces njihovog nastanka u organizmu, način predstavljanja proteinskih funkcija, problem predviđanja funkcije proteina, takmičenje *CAFA challenge* i način ocenjivanja mere kvaliteta modela.

3.1 Biološki pojmovi

Ubrzani razvoj računarskih performansi proizveo je nastanak novijih naučnih disciplina koje koriste moderne tehnike pri izučavanju. Bioinformatika je interdisciplinarna oblast u kojoj se razvijaju alati i metode za obradu bioloških podataka. Kako je zabeležen nagli uspon mašinskog učenja, do sada su u literaturi predložene brojne metode koje se mogu primeniti na probleme iz ove oblasti [1].

3.1.1 Aminokiseline

Aminokiseline su sačinjene od centralnog ugljenikovog atoma C_α koji je povezan sa jednim vodonikovim atomom, amino grupom ($-NH_2$), karboksilnom grupom ($-COOH$) i bočnim lancem koji se označava kao R -grupa [13]. Specifičnosti različitih vrsta aminokiselina ogledaju se u R -grupi. Ona je tipično formirana od hemijskih elemenata C, H, O, i N, ali se mogu javiti i drugi. Postoji oko 500 vrsta, ali su od interesa 22 *proteinogene aminokiseline* među kojima je 20 standardnih i 2 nestandardne. Proteinogene aminokiseline učestvuju u izgradnji proteina i dobijaju se uz pomoć ćelijskih mehanizama. Kada se formira veza između dve aminokiseline, molekul vode se otpušta i između njih se stvara kovalentna (peptidna veza). Kada se aminokiseline spoje u niz, nastaju *polipeptidni lanci*. Nazivi aminokiselina imaju troslovnu i jednoslovnu oznaku, što je prikazano u tabeli 3.1.

3.1.2 Proteini

Makromolekuli su molekuli sastavljeni od velikog broja atoma čije su gradivne jedinice jednostavniji molekuli koji se nazivaju *monomeri*. Proteini su biološki makromolekuli sačinjeni od aminokiselina koji imaju važnu ulogu u strukturi i funkcionisanju živih bića [14], a neke od njih su:

Aminokiselina	Troslovna oznaka	Jednoslovna oznaka
alanin	Ala	A
arginin	Arg	R
asparagin	Asn	N
asparaginska kiselina	Asp	D
cistein	Cys	C
fenilalanin	Phe	F
glicin	Gly	G
glutamin	Gln	Q
glutaminska kiselina	Glu	E
histidin	His	H
izoleucin	Ile	I
leucin	Leu	L
lizin	Lys	K
metionin	Met	M
prolin	Pro	P
serin	Ser	S
tirozin	Tyr	Y
triptofan	Trp	W
treonin	Thr	T
valin	Val	V

Tabela 3.1: Aminokiseline sa troslovnim i jednoslovnim oznakama

- učestvovanje u metaboličkim procesima
- sprovođenje DNK replikacije
- ćelijsko signaliziranje
- učestvovanje u strukturnoj izgradnji ćelija
- transportovanje molekula
- imunološka uloga (antibakterijska i antivirusna)
- učestovanje u savijanju drugih proteina
- sprovođenje mišićnih kontrakcija
- transport molekula sa jedne lokacije na drugu
- pospešivanje hemijskih reakcija u organizmu

Struktura proteina ima četiri aspekta: primarna, sekundarna, tercijarna i kvatenarna struktura (slika 3.1).

Primarna struktura

Primarna struktura proteina je predstavljena linearnim nizom aminokiselina (proteinska sekvencija) koje su međusobno povezane peptidnim vezama. Sekvenca se u računarstvu skladišti kao niz nad alfabetom od 20 slova, a ponekad se ostavlja i jedan simbol više zbog nestandardnih aminokiselina. Broj mogućih nizova raste eksponencijalno sa bazom 20 kako se uvećava dužina sekvence. U prirodi je to važno zbog diverzifikacije informacije čime se kontrolišu kompleksni biološki sistemi. Makromolekuli sa kraćom sekvencom aminokiselina se nazivaju *oligopeptidi* ili samo peptidi, dok su *proteini* karakterisani dužim nizom aminokiselina¹. Tipični proteini imaju par stotina aminokiselina, najmanji je *insulin* od 57, a najveći *titin* od 34350 aminokiselina [1]. Informacija na osnovu koje se utvrđuje koje će aminokiseline biti izabrane pri formiranju proteina i njihov redosled u sekvenci sadržana je u delovima *DNK* (dezoksiribonukleinske kiseline).

Primarna struktura čini jezgro informacije o posmatranom proteinu. Na primer, protein se savija u prostoru što je uslovljeno rasporedom aminokiselina u sekvenci. Postoji još faktora, ali se primarna sekvencija može smatrati fundamentalnim podatkom.

Sekundarna struktura

Sekundarna struktura nastaje prilikom lokalnih stabilizacija aminokiselina tokom građenja vodonične veze. Najpoznatiji oblici koji se javljaju u proteinima su α -spirala i β -traka, ali postoje i razni drugi oblici.

Tercijarna struktura

Tercijarna struktura obuhvata 3D konformacije proteinske sekvence. Priroda proteina je da svoj linearni niz aminokiselina uvija u 3D prostoru. U sklopu toga, može se posmatrati raspodela verovatnoća po svim oblicima u 3D strukturi. Verovatnoće za različite oblike puno variraju. Postoje vrste proteina koje imaju svoj dominantan oblik, a i oni koji su labilni po tom pitanju. Zanimljivo je da nekad veoma različite proteinske sekvence se mogu uvijati u vrlo slične oblike. U praktičnom smislu, proteini se ugrubo savijaju u konačan broj struktura za koje se može izvršiti klasifikacija. Stabilni proteini imaju svoj dominantan oblik, dok ostali mogu imati minorne fluktuacije ili da iskazuju visoku nestabilnost pa se izdvajaju u posebnu grupu.

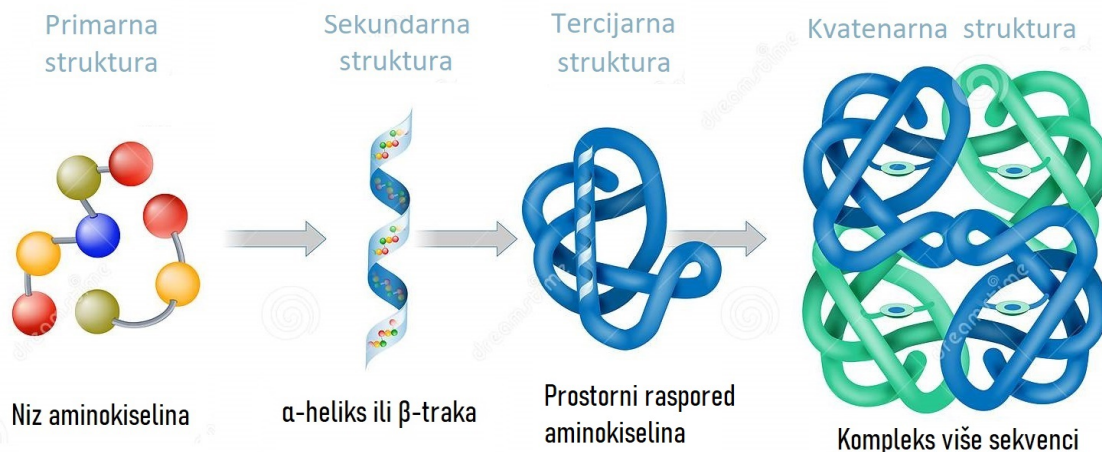
Kvaternarna struktura

Kvaternarna struktura proteina je karakterisana načinom udruživanja nekoliko sekvenci i njihovih interakcija. Nemaju svi proteini kvaternarnu strukturu, već samo oni kompleksniji koji su sastavljeni od više lanaca. Nekat protein izražava svoju biološku aktivnost samo kada su ti lanci oformljeni u kompleks. Primer takvog proteina je *hemoglobin*.

3.1.3 Centralna dogma molekularne biologije

Dezoksiribonukleinska kiselina (DNK) sadrži uputstva za razvoj i funkcionisanje svih živih organizama. DNK je makromolekul koji se sastoji od dva lanca spiralno isprepletana, čije su gradivne jedinice nuklotidi (nukleotidne baze, ili samo baze) kojih ima 4 vrste: adenin (A), citozin (C), guanin (G) i timin (T). Ta dva lanca su međusobno povezana vodoničnim vezama.

¹Granica u literaturi nije precizno određena



Slika 3.1: Struktura proteina: primarna, sekunarna, tercijarna i kvatenarna [15].

Naime, svaki nukleotid jednog lanca može biti uparen s određenim nukleotidom drugog i to tako da se adenin spaja uvek sa timinom (spajaju se sa dve vodikove veze), a citozin uvek sa guaninom (tri vodikove veze), i obrnuto. Preciznije, postoje sledeće veze: $A-T, T-A, C-G, G-C$. Zbog pomenute komplementarnosti, redosled nukleotida u jednom lancu u potpunosti određuje redosled nukleotida u drugom. Time se postiže da DNK ima jaču potporu. DNK sadrži genetičku informaciju koju nasleđuju potomci. Ta informacija određena je redosledom parova nukleotida. Lanac DNK sadrži gene, područja koja regulišu gene i područja za koju nije poznato da imaju neku funkciju. Geni su šifre koje nose informacije neophodne za sintezu sekvenci aminokiselina koje formiraju proteine. Centralna dogma molekularne biologije objašnjava pomenuti proces. Nju čine *replikacija*, *transkripcija* i *translacija* koji će biti opisani u nastavku (videti sliku 3.2 levo). Celokupan DNK sadržaj nekog organizma se naziva *genom*, dok se skup svih gena naziva *genotip* [16].

Replikacija

Potomci nasleđuju od svojih roditelja DNK čime se obezbeđuje kontinuitet bioloških vrsta. To se postiže tako što se molekuli DNK udvajaju prilikom svake deobe ćelije. Enzimi koriste roditeljski molekul DNK kao kalup za sintezu novih identičnih kopija. Otuda i naziv replikacija, jer su novonastali molekuli DNK kopije odnosno replike roditeljskog molekula DNK. Deobom će obe ćelije dobiti po jednu DNK tako da će obe imati isti genotip. Proces replikacije obuhvata odmotavanje lanaca DNK, određivanje mesta od koga će početi replikacija, sintezu novog lanca i drugo. Sinteza novog lanca se vrši tako što se na osnovu kalupa DNK nižu nukleotidi komplementarni u odnosu na taj DNK lanac. Tako nastaju dva ista molekula DNK, od kojih svaki sadrži jedan stari (roditeljski) i jedan novosintetisani lanac.

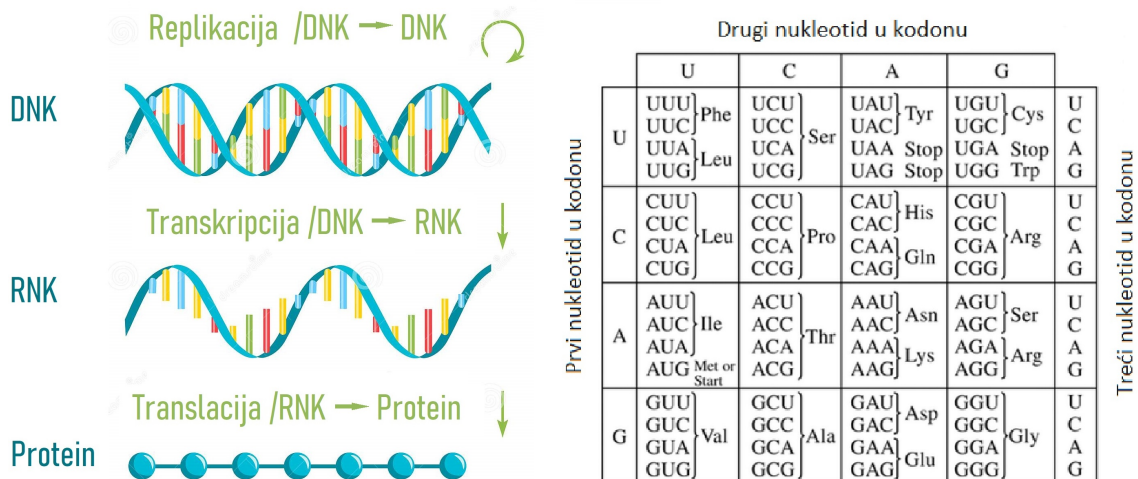
Izmene - *mutacije* kod DNK polnih ćelija mogu imati ozbiljnije posledice pošto sve ćelije jednog organizma nastaju deobama oplodene jajne ćelije. Poznata su mnoga nasledna oboljenja nastala mutacijom u nekom genu (srpasta anemija, cistična fibroza, mišićna distrofija, itd.) [16]

Transkripcija

Transkripcija je proces kopiranja informacija sa molekula DNK na molekul RNK. Za sprovođenje transkripcije odgovoran je enzim RNK polimeraza. Za sintezu RNK ne koristi se ceo molekul DNK već samo gen. U procesu transkripcije, samo jedan od dva lanca DNK služi za kopiranje i naziva se *matrični*, dok je drugi označen kao *nematrični*. Prateći nukleotide matričnog lanca, RNK polimeraza formira novi niz nukleotida komplementaran polaznom, samo što se umesto timina u sekvenci nalazi *uracil*. Kako je nematrični lanac takođe komplementaran matričnom, on ima iste nukleotide kao formirani molekul RNK s tim što je umesto timina uracil. Kod eukariota tako formirana RNK odlazi u citoplazmu gde se dalje obrađuje modifikacijom krajeva lanca kao i odsecanjem u unutrašnjosti *introna* (za koje se smatra da nemaju ulogu) i povezivanjem *egzona* (koji sadrže glavnu informaciju). Tako nastaje zrela *informaciona RNK* (*iRNK*) koja sadrži uputstvo za sintezu proteina [17].

Translacija

Translacija je proces u kome se na osnovu informacione RNK (*iRNK*) nastale u transkripciji sintetišu polipeptidni lanci. RNK nastala u transkripciji je dobijena iz gena. Introni i egzoni u genima imaju i ulogu štednje skladišnog prostora. Naime, u procesu transkripcije se odsecaju introni, a povezuju egzoni i to može biti učinjeno na različite načine. Time jedan gen može sadržati informaciju za više proteina. Sve komponente koje učestvuju u biosintezi proteina okupljaju se u ribozomima, u kojima se aminokiseline spajaju peptidnim vezama. *iRNK* daje uputstvo na osnovu kojeg se vrši spajanje. Svaki uzastopni triplet nukleotida (*kodon*) formira translatornu RNK (*tRNK*) za koju se veže aminokiselina određena tim kodonom. Aminokiseline se zatim nižu jedna za drugom formirajući proteinsku sekvencu. Svaki kodon jedinstveno određuje aminokiselinu, ali preslikavanje nije injektivno, odnosno više tripleta mogu kodirati istu aminokiselinu [16]. Na slici 3.2 desno je prikazano kodiranje kodonima. Tabela predstavlja pogodnu shemu određivanje aminokiseline koju formiraju neka tri izabrana nukleotida.



Slika 3.2: **Levo:** Centralna dogma molekularne biologije [18]. **Desno:** Tabela kodiranja tripleta aminokiselina. [19].

3.2 Problem predviđanja funkcija proteina

Kako su proteini veoma značajni u funkcionisanju živih bića, veoma je važno je razvijati računarske metode za automatsko prepoznavanje njihovih funkcija. To nije nimalo lak posao i još uvek se traga za što boljim metodama. Brojni faktori mogu uticati na ulogu proteina, a neki od njih su: primarna i ostale proteinske strukture, okruženje u kojem protein deluje, odnosno kakva fizičko hemijska svojstva se ispoljavaju, interakcije sa drugim proteinima, itd [1].

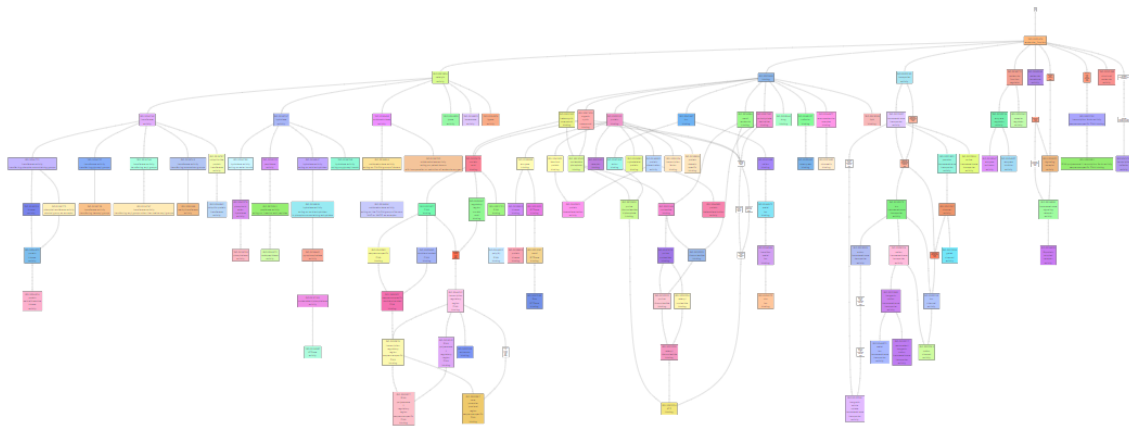
3.2.1 Reprezentacija funkcije proteina

Obimno proučavanje funkcija proteina i uključivanje računarskih tehnika zahteva dobru organizaciju naziva proteinskih funkcija. Zbog toga je uvedena *ontologija* koja predstavlja usmereni aciklički graf u kome su čvorovi funkcije proteina. U oblasti funkcija proteina je napoznatija *GO* ontologija (engl. *Gene Ontology*) [5]. U ovoj ontologiji su funkcije proteina organizovane hijerhijski, tako da ukoliko je određen čvor funkcija nekog proteina, tada i svi preci tog čvora moraju biti takođe u skupu funkcija istog tog proteina. Zapravo, svaki dublji čvor predstavlja dalju specijalizaciju funkcije svojih roditelja. Koren predstavlja uopštenje svih funkcija. Čvorovi mogu imati više direktnih potomaka, kao i više roditelja. Grane između čvorova označavaju odnos između dva čvora („je”, „je deo”, itd.). Postoje tri velike ontologije vezane za proteinske funkcije i to su *molekularne funkcije* (engl. *Molecular Function Ontology - MFO*), *biološki procesi* (engl. *Biological Process Ontology - BPO*) i *ćelijske komponente* (engl. *Cellular Component Ontology - CCO*). One pokrivaju i različite aspekte na proteinske funkcije. U molekularnim funkcijama se teži da se objasni kakva je aktivnost proteina, strukturni i mehanički procesi u reakcijama. U biološkim procesima je akcenat na višem nivou funkcionisanja (na primer, zbog čega je izvršena neka akcija), dok se u ćelijskim komponentama stavlja akcenat na lokacijama u kojima se vrše određene funkcije. U ovom radu je odabrana ontologija molekularnih funkcija koja je redukovana sa 11113 (u toku pisanja ovog rada) na 123 čvora (videti sliku 3.3). Redukcija je izvršena zbog ograničenih računarskih i vremenskih resursa, a primarni cilj rada je primena novih metoda. Zarad jednostavnijeg analiziranja, pogodno je bilo smanjiti ontologiju, a dalji rad podrazumeva uključivanje i preostalih čvorova [1].

Skup funkcija za određeni protein određen je konzistentnim podgrafom ontologije. Podatak o svim funkcijama proteina se najčešće daje u obliku niza listova podgrafa koje predstavlja skup funkcija. Ako su zadati listovi, jasno je da se informacija o svim funkcijama proteina može dobiti propagiranjem svih čvorova do korena. Jedan takav primer dat je slikom 3.4.

3.2.2 Eksperimentalna anotacija

Kao što je pomenuto u uvodu, eksperimentalna anotacija najverodostojnija po pitanju utvrđivanja funkcija proteina. Međutim, može doći do grešaka usled raznih uzroka poput pogrešnog tumačenja od strane biokuratora, neadekvatnog sprovođenja eksperimenta, nemogućnosti da se neke pojave posmatraju unutar organizma, itd. Takođe, teško je utvrditi i da neki protein *nema* određenu funkciju. Uprkos tome, biokuratori sakupljaju informacije od izuzetne vrednosti koje su glavni izvor informacije o proteinskim funkcijama. Ipak, nekad nije moguće sprovesti odgovarajući broj eksperimenata zbog ograničenja poput budžetskih, etičkih, vremenskih i ostalih razloga, tako da je upotreba računara u ove svrhe važna zbog efikasnosti [1].



Slika 3.3: Prikaz grafovske strukture redukovane ontologije molekularnih funkcija (MFO) na 123 čvora i 145 grana [20].

3.2.3 Problem predviđanja funkcija proteina

U ovom odeljku će biti opisan problem predviđanja funkcije novog proteina na osnovu prethodnih podataka [1]. Neka je dat skup eksperimentalno validiranih proteina $P = \{p_1, p_2, \dots, p_N\}$. Svakom od njih se pridružuje njegova eksperimentalna anotacija. Tako je dobijen skup $\mathcal{D}_l = \{(p_1, T_1), (p_2, T_2), \dots, (p_N, T_N)\}$. Sa $T_i \subseteq \mathcal{O}$ je označen *konzistentni podgraf* ontologije $\mathcal{O}$ za protein p_i . Protein p_i je dat sekvencom, ali mogu biti prisutni i drugi podaci kao što je informacija o 3D strukturi, poreklo organizma u kojem deluje, interakcije sa drugim proteinima, itd.

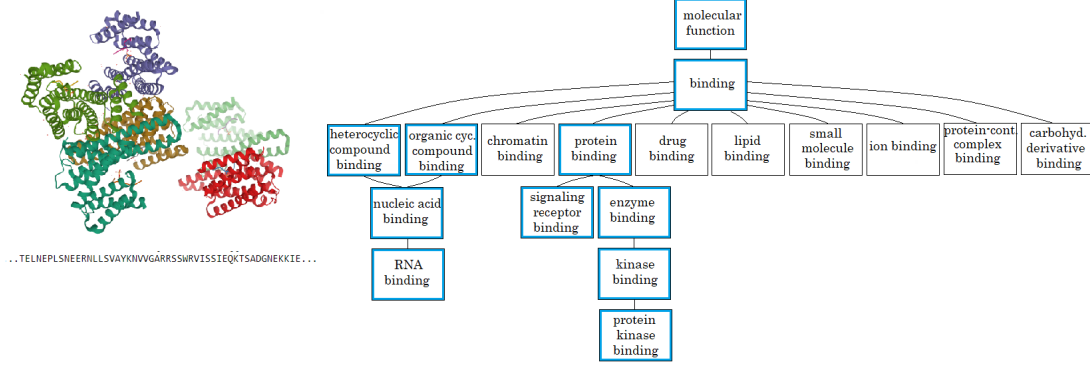
Problem *predviđanja funkcija proteina* se može definisati na sledeći način. Dat je skup eksperimentalno validiranih proteina $\mathcal{D}_l$, opciono, dodatni skup karakteristika proteina $\mathcal{D}$. Zadat je protein p za koji je poznata proteinska sekvenca i karakteristike, ali nije poznata anotacija. Predviđanje funkcije proteina predstavlja određivanje konzistentnog podgrafa $\hat{T} \subseteq \mathcal{O}$ koji je najverovatnija eksperimentalna anotacija tog proteina. Odnosno:

$$\hat{T} = \arg \max_{T \subseteq \mathcal{O}} \mathcal{P}(T \mid p) \quad (3.1)$$

U prethodnoj jednakosti 3.1, $\mathcal{P}(T \mid p)$ označava aposteriornu verovatnoću da je $T \subseteq \mathcal{O}$ baš podgraf za protein p koja se dobija pomoću odabrane metode kojom se vrši predviđanje.

3.2.4 CAFA takmičenje

Sekvenciranjem genoma saznajemo sadržaje DNK sekvenci, što otkriva gene kao njihove fragmente. Geni daju uputstvo za sintezu proteina prema centralnoj dogmi molekularne biologije. Opšti trend cene sekvenciranja genoma je opadajući što prouzrokuje rapidan rast broja novih sekvenci DNK, a samim tim i proteinskih sekvenci. Potrebno je odrediti ulogu tih proteina, a kako su eksperimenti skupi, teški za izvođenje, iziskuju vreme i druge resurse, za sada je nemoguće organizovati eksperimente koji bi izvršili anotaciju za većinu poznatih sekvenci. Automatizacija postaje neizbežni put ka ublaženju ovog problema. U tom cilju se organizuju takmičenja CAFA (engl. The Critical Assessment of protein Function Annotation algorithms)



Slika 3.4: Reprezentacija jednog podgraфа svih funkcija proteina [21] u redukovanom grafu [22].

Funkcija proteina je data sledećim skupom listova: *RNA binding*, *signaling receptor binding* i *protein kinase binding*. Kao što je napomenuto, svi preci datih čvorova moraju biti označeni pa su tako svi uokvireni nazivi plavom bojom takođe u skupu funkcija datog proteina. Po nazivima nekih čvorova se jasno može primetiti da oni koji imaju veću dubinu u grafu, predstavljaju dalju specijalizaciju u odnosu na njihove pretke. Pomenutom organizacijom se mogu i izvesti činjenice poput: „Dati protein ima ulogu proteinske kinaze (*protein kinase binding*) koja je kinaza (*kinase binding*) iz tipa enzimskih vezivanja (*enzyme binding*) koja su iz grupe proteinskih vezivanja (*protein binding*), a proteinska vezivanja su uopšteno vezivanja (*binding*) iz grupe molekularnih funkcija (*molecular function*)”. Takva interpretacija je moguća, ali je kompaktnija informacija u obliku označenog podgraфа.

kako bi se evaluirale metode za predviđanje funkcije proteina i pratio njihov opšti razvoj. Za sada su održana četiri CAFA takmičenja počevši od 2010, dok je peto po redu aktuelno u toku pisanja ovog rada [2].

3.2.5 Evaluacija modela

Neka je $\hat{T}_i(\tau)$ podgraf ontologije koji predstavlja predviđanje metode, pri čemu je dat ulazni parametar τ kao prag verovatnoće na osnovu kojeg se odlučuje da li se određeni čvor predviđa kao funkcija proteina ili ne. Svaki čvor dobija aposteriornu verovatnoću koje formira prediktor na osnovu podataka koje ima o datom proteinu, ali i ostalim proteinima, a zatim se na osnovu praga odlučuje da li je za neki čvor pozitivno predviđanje ili ne. Na primer, ako je verovatnoća 0.6 da je neki čvor funkcija datog proteina, za prag $\tau = 0.5$ taj čvor će biti klasifikovan da jeste, dok za prag $\tau = 0.8$ ta verovatnoća ukazuje da to nije funkcija tog proteina. Tako se mogu definisati [1] preciznost (engl. *precision*):

$$pr_i(\tau) = \frac{\sum_{v \in \mathcal{O}} I(v \in \hat{T}_i(\tau) \wedge v \in T_i)}{\sum_{v \in \mathcal{O}} I(v \in \hat{T}_i(\tau))} \quad (3.2)$$

i odziv (*recall*):

$$rc_i(\tau) = \frac{\sum_{v \in \mathcal{O}} I(v \in \hat{T}_i(\tau) \wedge v \in T_i)}{\sum_{v \in \mathcal{O}} I(v \in T_i)} \quad (3.3)$$

Sa I je označena indikatorska funkcija za koju važi $I(\top) = 1$ i $I(\perp) = 1$. Kod nebalansiranih klasa, odgovarajuća mera može biti harmonijska sredina preciznosti i odziva koja se naziva F_1 skor [1]. Preciznije, F_1 skor se definiše pomoću formule:

$$F_1(\tau) = \frac{2 \cdot pr(\tau) \cdot rc(\tau)}{pr(\tau) + rc(\tau)} \quad (3.4)$$

Poglavlje 4

Neuronske mreže

Ova glava sadrži kratak uvod o mašinskom učenju, nakon čega se posebno izdvajaju neuronske mreže kao modeli koji se koriste u ovom radu. Opisane su tri vrste neuronskih mreža: *potpuno povezane*, *konvolutivne* i *grafovske*. Svi procesi koji se koriste za kreiranje modela opisani su u okviru potpuno povezanih neuronskih mreža.

4.1 Kratak uvod o mašinskom učenju

Mašinsko učenje je grana veštačke inteligencije koja se bavi kreiranjem sistema koji su sposobni da generalizuju znanja o zavisnostima u podacima na osnovu prethodnih iskustava. Ovu oblast delimo na:

- *nadgledano učenje* (engl. *supervised learning*)
- *nenadgledano učenje* (engl. *unsupervised learning*)
- *učenje potkrepljivanjem* (engl. *reinforcement learning*)

Elementi određenog skupa podataka se nazivaju instance (engl. *instances*). Pošto su skladištene elektronski u računaru, potrebno je definisati njihove opise shodno toj reprezentaciji. Skup atributa (engl. *features*) se određuje tako da predstavlja svojstva podataka iz realnog sveta u računarskom obliku. Skup atributa je zajednički za instance istog skupa. Domen atributa je najčešće numerički ili kategorički, ali se mogu javiti i kompleksniji pojmovi kao što su grafovi, sekvence itd. Svaka instanca može imati različite vrednosti atributa, pri čemu su elementi vektora $\vec{x}$ vrednosti atributa za datu instancu. Nadgledano učenje i učenje potkrepljivanjem podrazumevaju i uvođenje pojma ciljne promenljive (engl. *target variable*), koja može biti jednodimenzionalna ili višedimenzionalna. Pretpostavljamo da postoji neka funkcionalna zavisnost između ulaza i izlaza. Cilj je napraviti model koji će dati što bolju aproksimaciju te zavisnosti. Vrednosti atributa za instancu $\vec{x}$ predstavljaju ulaz za model, dok vrednost ciljne promenljive, u oznaci y , predstavlja izlaz modela. Model je određen nekim skupom parametara [23].

Nadgledano učenje je verovatno najzastupljeniji vid mašinskog učenja i odlikuje ga postojanje znanja o pravoj vrednosti ciljne promenljive (engl. *ground truth*) za deo skupa koji služi za obučavanje (engl. *training*) i za evaluaciju modela (engl. *evaluation*). Svrha obučavanja je

podešavanje vrednosti parametara modela tako da se minimizuje greška koja se ustanovljava poređenjem predviđanja modela određenim trenutnim skupom parametara i prave vrednosti ciljne promenljive.

Dva osnovna problema nadgledanog učenja su regresija (engl. *regression*) i klasifikacija (engl. *classification*). Regresija predstavlja problem predviđanja neprekidne ciljne promenljive, a klasifikacija je problem predviđanja kategoričke ciljne promenljive odnosno one koja ima konačan skup vrednosti. U zavisnosti koji problem je u pitanju definišu se različite vrste grešaka.

Dobra generalizacija se karakteriše konstruisanjem modela koji teži da minimizuje grešku na skupu za obučavanje, ali ne prilagođavajući se previše podacima. *Preprilagođavanje* (engl. *overfitting*) je pojava u mašinskom učenju kada prilikom obučavanja model podešava parametre minimizirajući grešku na skupu za obučavanje tako da se generalizacija naruši. Protivteža preprilagođavanju su metode regularizacije (engl. *regularization*). Provera kvaliteta predviđanja vrši se na test skupu koji služi za evaluaciju modela (engl. *test set*). Uloga test skupa je ocenjivanje prediktivnih performansi modela [23].

4.2 Veštački neuron

Neuron je model mašinskog učenja koji za argumente prima vektor $\vec{x} \in \mathbb{R}^n$ i računa kompoziciju nelinearne funkcije i funkcije koja predstavlja linearnu kombinaciju koordinata vektora $\vec{x}$ uz dodatni slobodni član b [24]. Linearna kombinacija je određena vektorom težina $\vec{w}$ (engl. *weights*). Za nelinearnu funkciju se koristi *funkcija aktivacije* (engl. *activation function*) koja se u literaturi najčešće označava sa σ . Izlaz ovog modela u jednodimenzionalnom slučaju je vrednost $y \in \mathbb{R}$.

$$y = \sigma\left(b + \sum_{k=1}^n w_k \cdot x_k\right) = \sigma(b + \vec{w}^T \vec{x}) \quad (4.1)$$

Funkcije aktivacije se koriste kako bi se obezbedila nelinearnost modela. Naime, linearni modeli ne mogu dovoljno dobro aproksimirati neke funkcije, što je osnovni cilj modela mašinskog učenja. U nastavku je data klasifikacija nekih aktivacionih funkcija (videti sliku 4.1):

- *Sigmoidna funkcija*:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Ranije je bila veoma korišćena jer ima neke pogodne osobine kao što su jednostavno izračunavanje izvoda kada je poznata vrednost funkcije odnosno $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. Takođe, opseg vrednosti $[0, 1]$ daje mogućnost modelovanja Bernulijeve raspodele koja je pogodna u klasifikaciji. U izboru između klase 0 i 1 pogodno je dati verovatnoću, pri čemu vrednosti bliske nuli ukazuju veću pouzdanost modela da je to zaista 0. Analogno za vrednosti bliske jedinici važi da je model sigurniji u predviđanje da to zaista jeste 1. Mana sigmoidne funkcije je što udaljavanjem od nule počinje da se ponaša slično kao konstanta, pa je izvod funkcije na tom intervalu blizak nuli. Zbog toga se javlja problem nestajućih gradijenata (engl. *vanishing gradients*). Ukoliko se prilikom lančanog pravila gradijenata u proizvodu pojavljuju vrednosti bliske nuli, može nastati potkoračenje zbog zapisa decimala

u obliku pokretnog zareza. Potkoračenje kod vrednosti gradijenta predstavlja problem u optimizaciji greške neuronske mreže (videti odeljak 4.4.2).

- *ReLU* - ispravljena linearna jedinica (engl. *rectified linear unit*)

$$\sigma(x) = \max(x, 0)$$

U modernim neuronskim mrežama ReLU funkcija je uglavnom preporučeni izbor. Iako liči na linearnu funkciju (jer je deo-po-deo linearna i „prelomljena“ na dva dela) ona ima sposobnost unošenja nelinearnosti. Intuitivno, činjenica da se neprekidne funkcije mogu proizvoljno dobro aproksimirati deo-po-deo linearnom funkcijom ukazuje da prelom kod ispravljene linearne jedinice doprinosi ekspresivnosti modela. Veoma jednostavna definicija i jednostavan izvod daju ovoj aktivacionoj funkciji neke pogodnosti pri optimizaciji zbog čega se toliko često koristi. Osim toga, inspiracija za ReLU funkciju dolazi i iz načina funkcionisanja čovekovog mozga. Naime, veliki broj neurona u nekom trenutku ne proizvodi električne impulse, što odgovara definisanju vrednosti 0 u negativnom delu x -ose. Međutim, jedna mana koja se pojavljuje je vrednost nula u negativnom delu ove funkcije. To može prouzrokovati nestajuće gradijente zbog čega su predložene dve nove varijante: *LeakyReLU* i *ELU*.

- *LeakyReLU* - nakrivljena linearna jedinica

$$f(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$

Umesto konstantne nule u negativnom delu grafika, može se izabrati hiperparametar α koji je obično mali po apsolutnoj vrednosti (na primer $\alpha = 0.01$), a može se staviti i kao parametar za učenje (nije fiksiran). Tada se dobija parametrizovana verzija koja je poznata kao *PReLU* (engl. *parametric ReLU*)

- *ELU* - eksponencijalna linearna jedinica (engl. *exponential linear unit*)

$$f(x) = \begin{cases} x & x \geq 0 \\ \exp(x) - 1 & x < 0 \end{cases}$$

Ima slične pogodnosti kao *nakrivljena linearna jedinica* i diferencijabilna je u nuli.

4.3 Potpuno povezane neuronske mreže

Neuronske mreže (engl. *neural networks*) su modeli mašinskog učenja koji su organizovani po slojevima (engl. *layers*). Ulaz u neuronsku mrežu je dat vektorom atributa. Svaki sloj sadrži svoje računске jedinice (neurone). Poslednji sloj je izlaz iz mreže (engl. *output layer*) i predstavlja predviđanje koja mreža daje. Slojevi koji nisu izlazni nazivaju se skriveni slojevi (engl. *hidden layers*). Ukoliko mreža sadrži više od jednog skrivenog sloja govorimo o dubokoj neuronskoj mreži (engl. *deep neural network*), a proces učenja kod takve mreže nazivamo

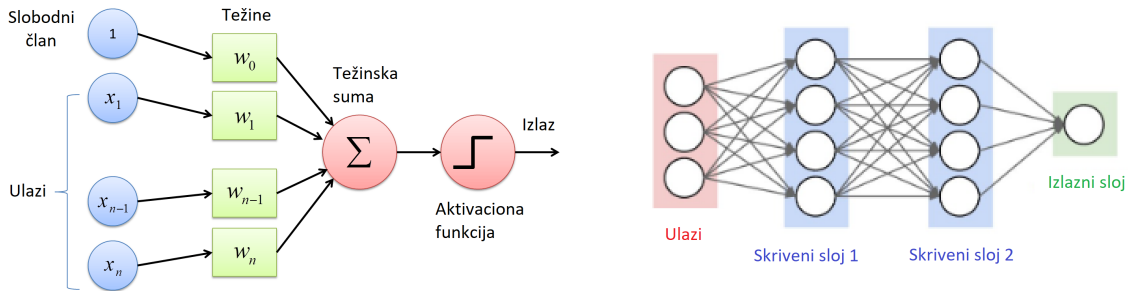


Slika 4.1: Aktivacione funkcije korišćene u ovom radu.

duboko učenje (engl. *deep learning*). Kod potpuno povezanih neuronskih mreža (engl. *fully connected neural networks*), prvi sloj nakon ulaza sadrži neurone koji kao argumente primaju samo vektor ulaza $\vec{x}$. Svaki naredni sloj sadrži neurone sa argumentima čije su vrednosti dobijene u neuronima iz prethodnog sloja.

Matematički zapisano, ako je ulaz u i -ti sloj $\vec{x}$ onda j -ti neuron sa svojim težinama $\vec{w}_j$ i slobodnim članom b_j računa izlaznu vrednost y_j na sledeći način:

$$y_j = \sigma(b_j + \vec{w}_j^T \vec{x}) \quad (4.2)$$



Slika 4.2: Na slici **levo** se nalazi *neuron*. Na slici **desno** se nalazi mala potpuno povezana neuronska mreža sačinjena organizovanjem neurona u slojeve [25].

Iako se sloj definiše kao skup neurona, potrebno je fiksirati redosled kako bi se uvek znalo koji vektor se prosleđuje narednom sloju. Izračunavanja za svaki sloj i mogu se izraziti matričnim zapisom preko formule (4.2) kada objedinimo sve neurone koje on poseduje dobijamo vektor vrednosti neurona $\vec{y}$:

$$\vec{y} = \sigma(W_i \vec{x} + \vec{b}_i) \quad (4.3)$$

Matrica W_i i vektor slobodnih članova $\vec{b}_i$ se definišu za svaki sloj. Matrica W_i je sastavljena od redova $w_i^{(j)}$ koji su težine za neuron j i naziva se težinska matrica (engl. *weight matrix*). Formulom (4.3) se dati ulazni vektor $\vec{x}$ (koji je odgovarajuće dimenzije) transformiše u izlazni vektor koji predstavlja ulaz za naredni sloj i tako dalje.

Umesto slobodnog člana b u definiciji, moguće je proširiti vrednosti atributa sa jedinicom, a u težinama slobodni član označiti sa $w_0 = b$. Tako se skalarnim množenjem dobija isti rezultat kao ranije samo kompaktnije zapisan. Primer potpuno povezane neuronske prikazan je na slici 4.2). Naziv „mreža” potiče od puno veza koje postoje između slojeva. Univerzalna teorema o aproksimaciji neuronskih mreža izražava sposobnost neuronskih mreža da aproksimiraju široku klasu funkcija [26].

Teorema 1. *Neka je g ograničena i strogo rastuća neprekidna funkcija. Tada za svaku funkciju $f \in C[0, 1]^n$ i svako $\epsilon > 0$, postoji $m \in \mathbb{N}$, matrica $W \in R^{m \times n}$, vektor $w_0 \in R^m$ i vektor $v \in R^m$, tako da za svako $x \in [0, 1]^n$ važi*

$$|v^T g(Wx + w_0) - f(x)| < \epsilon$$

Naime, svaka neprekidna funkcija na kompaktnom skupu $[0, 1]^n$ može se proizvoljno dobro aproksimirati sa neuronskom mrežom koja ima samo jedan skriveni sloj. Međutim, dokaz ove teoreme nije konstruktivnog tipa, zbog čega ne mora nikako značiti da je tu neuronsku mrežu lako naći. Iako univerzalna teorema o aproksimaciji ne uvodi postojanje više slojeva, praksa pokazuje da su arhitekture dubokih neuronskih mreža jako uspešne. Nije poznat ni način za utvrđivanje optimalnog broja slojeva.

4.4 Obučavanje neuronske mreže

U prethodnom izlaganju, pomenuti su razni pojmovi koji karakterišu model (broj slojeva, težine, slobodni članovi itd.), ali nije precizirano kako se oni određuju. Neki mogu odabrani na osnovu podataka, domenskih znanja, isprobavanjem, itd. dok drugi deo parametara ostaje da bude kreiran u toku učenja (engl. *learning*) odnosno obučavanja modela. Inicijalizovane početne vrednosti parametara se u toku ove faze prilagođavaju, tako da model što bolje aproksimira vezu između ulaznih podataka i ciljne promenljive.

4.4.1 Inicijalizacija parametara

Standardni proces nadgledanog učenja podrazumeva inicijalizaciju parametara, a zatim njihovo iterativno ažuriranje. Važno je odabrati odgovarajući način kreiranja početnih vrednosti jer toga zavisi koliko brzo će proces konvergirati i ka kom lokalnom minimumu. Trenutno, najčešće se koristi inicijalizacija iz neke empirijski izabrane raspodele verovatnoće. Uglavnom su to *uniformna* i *normalna* raspodela čije je matematičko očekivanje 0. Kako svaki neuron ima svoje parametre, može se uzeti raspodela koja je vezana za konkretni neuron. Uticaj na odabir raspodele, između ostalog, mogu imati i broj ulazih veza (n_{in}) u taj neuron kao i broj izlaznih veza (n_{out}) ka narednom sloju koje direktno polaze iz posmatranog neurona. Ukoliko je poznato neko domensko znanje o problemu, moguće je tu činjenicu iskoristiti u svrhu postavljanja početnih vrednosti parametara.

U ovom radu korišćena je *uniformna normalizovana inicijalizacija* [27] (engl. *uniform normalized initialization*) u kojoj se svaki parametar w_i iz težina $\vec{w}$ nekog neurona bira iz raspodele: $w_i \sim \mathcal{U}(-a, a)$, gde je:

$$a = g \cdot \sqrt{\frac{6}{n_{in} + n_{out}}}, \quad (4.4)$$

pri čemu g označava opcioni skalirajući faktor. Izbor uniformne normalizovane inicijalizacije povoljno utiče na problem nestajućih i eksplodirajućih gradijenata.

4.4.2 Optimizacija modela

U mašinskom učenju, modeli se kreiraju na osnovu postojećih podataka. Neka su dati parovi ulaz-izlaz u vektorskom obliku $\{(\vec{x}_k, \vec{y}_k)\}_{k=1, \dots, N}$ koji predstavljaju skup na kome će se model obučavati. Vrednosti atributa u obliku vektora $\vec{x}_k$ predaju se modelu f_θ koji na osnovu skupa svojih parametara θ određuje izlaz $\vec{\hat{y}}_k = f_\theta(\vec{x}_k)$. U zavisnosti od vrednosti skupa parametara θ , dobijaju se različiti modeli. Cilj je pronaći model koji najbolje aproksimira zavisnost ulaza i izlaza u nekom smislu. Potrebno je uvesti meru za odstupanje predviđanja ciljne promenljive $\vec{y}_k$ od njene prave vrednosti $\vec{y}_k$. Definišemo grešku (engl. *loss*) modela kao funkciju koja u slučaju *regresije* može biti prosečno odstupanje od stvarne vrednosti (engl. *mean squared error*). Matematički zapisano:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{k=1}^N \left\| \vec{y}_k - \vec{\hat{y}}_k \right\|^2 \quad (4.5)$$

Ako se radi o problemu binarne klasifikacije, standardna greška koja se uzima je *unakrsna entropija* (engl. *binary cross entropy*). Ako uprosečimo vrednost ove greške po instancama dobijamo:

$$\mathcal{L}_{BCE} = -\frac{1}{N} \sum_{k=1}^N y_i \cdot \ln(\hat{y}_i) + (1 - y_i) \cdot \ln(1 - \hat{y}_i) \quad (4.6)$$

Prirodno je želeći da model što manje greši, međutim veoma mala greška ne mora značiti da je pronađen odličan model. To što se model savršeno prilagođava podacima za obučavanje, ne znači da će dobro predviđati na novim (neviđenim) podacima. Kako bi se uspostavila ravnoteža između toga što model treba dovoljno, ali ne i previše da se prilagodi podacima za obučavanje, uvodi se tehnika *regularizacije* (engl. *regularisation*) kojom se uspostavlja protivteža preprilagođavanju. Ova tehnika će opširnije biti opisana u narednom odeljku 4.4.3. U nastavku podrazumevamo da u $\mathcal{L}$ možemo imati i regularizacione izraze. Regularizacioni izraz je funkcija čiji su argumenti parametri koji se koriste u optimizaciji, a njihova uloga dodavanja funkciji greške je onemogućavanje modelu da se previše prilagodi podacima podešavanjem parametara.

Funkcija greške $\mathcal{L}$ zavisi od parametara $\vec{\theta}$ jer se različitim izborom dobijaju različiti modeli. Cilj je, dakle, rešiti problem minimizacije:

$$\min_{\vec{\theta}} \mathcal{L}(\vec{\theta}), \quad \vec{\theta} \in \mathbb{R}^n, \quad (4.7)$$

gde n predstavlja broj atributa, odnosno dimenziju vektora $\vec{\theta}$. Pošto nisu data ograničenja, ovo je problem bezuslovne optimizacije. On se može numerički rešavati iterativno. Osnovni princip je da se generiše niz

$$\theta_{j+1} = \theta_j + \alpha_j \vec{d}_j, \quad j = 0, 1, 2, 3, \dots \quad (4.8)$$

tako da vektor $\vec{d}_j \in \mathbb{R}^n$ određuje pravac kretanja iz θ_j . Realni pozitivni broj α_j utiče na dužinu koraka vektora $\vec{d}_j$. Odabir $\vec{d}_j$ i α_j vrši se na taj način da iterativni proces ima trend opadanja vrednosti funkcije i konvergira ka nekom od lokalnih minimuma funkcije.

Jedna od najpoznatijih klasa metoda bezuslovne optimizacije za diferencijabilne funkcije su *gradijentne metode*. Njihova glavna karakteristika je korišćenje gradijenata u svrhu izbora pravca kretanja. Gradijent je pravac najstrmijeg rasta funkcije, dok suprotan od tog pravca predstavlja najstrmiji spust. Postoje različite varijante gradijentnog spusta i neke od njih ne podrazumevaju direktno kretanje najstrmijim spustom. Neke od njih koriste akumuliranje uticaja nekoliko prethodnih gradijenata koju koristi *metod inercije*, *Nesterovljev ubrzani gradijentni spust*, a najkorišćenija metoda optimizacije u modernim neuronskim mrežama je *Adam* [28] koji se zasniva na ocenama prvog i drugog momenta gradijenata [29].

Računanje gradijenata za veliko N (što znači da postoji veliki broj podataka) je vremenski zahtevna operacija za računar i zbog toga se gotovo uvek kod neuronskih mreža pribegava alternativnoj varijanti. Umesto trošenja značajnih resursa za izračunavanje najstrmijeg pravca spusta (koji inače i ne mora voditi najbrže ka minimumu), koristi se računanje *stohastičke aproksimacije gradijenta* koji služi kao neka vrsta aproksimacije. Tako se dolazi do metode stohastičkog gradijentnog spusta (engl. *stochastic gradient descent*). Proces počinje nasumičnim mešanjem skupa za obučavanje (engl. *shuffling*). Nakon toga, ceo skup za obučavanje se particioniše tako da svi podskupovi, osim možda poslednjeg koji se formira, imaju fiksiran broj elemenata (taj broj predstavlja hiperparametar). Obučavanje se vrši prolaženjem kroz petlju, tako što se podskupovi instanci (engl. *mini - batch*) redom predaju funkciji greške jedan po jedan. Poskupovi instanci su označeni kao $\mathcal{B}_k$ (k je indeks koji predstavlja redni broj grupe instanci u petlji). Za svaki takav podskup, računa se gradijent funkcije greške *samo* na tim instancama i ažuriranje parametara se vrši u tom pravcu.

$$\theta_{j+1} = \theta_j - \alpha_j \nabla \mathcal{L}_{\mathcal{B}_k}(\vec{\theta}_j), \quad (4.9)$$

gde je $\mathcal{L}_{\mathcal{B}_k}(\vec{\theta})$ funkcija greške izračunata samo na instancama iz skupa $\mathcal{B}_k$, a $j \in \{0, 1, 2, 3, \dots\}$. Ažuriranje parametara se vrši na osnovu poskupova instanci. Matematičko očekivanje pravaca stohastičkih gradijenata svih ovih vektora je upravo gradijent, što sa teorijske strane daje potporu za korišćenje ove metode. Sa praktične strane gledišta, motivi su brojni (na primer, ne mogu svi podaci stati u memoriju grafičke karte odjednom), zbog čega je i ovaj način optimizacije kod neuronskih mreža preferiran. Ovde je predstavljena stohastička varijanta gradijentnog spusta. Slično se uvode i stohastičke verzije ostalih metoda optimizacija koje se koriste.

Nakon što se vrednosti parametara ažuriraju prolaskom kroz celu petlju odnosno sve grupe instanci $\mathcal{B}_k$, $k = 1, 2, 3, \dots$ budu posećene jedanput, završava se jedna epoha (engl. *epoch*). Broj epoha može biti fiksiran ili kontrolisan od strane nekog kriterijuma zaustavljanja. Kriterijumi zaustavljanja koji se najčešće koriste su u vezi sa odnosnom dve ili nekoliko uzastopnih tačaka iteracije. Odnosno, ako su dve uzastopne tačke dovoljno blizu ili su blizu njihove vrednosti, onda se proces zaustavlja. Kriterijum zaustavljanja koji se tipično koristi (i opisan je u odeljku 4.4.4) je prekidanje procesa ukoliko se mera kvaliteta modela ne poboljšava nakon fiksnog broja epoha.

Vrednosti koraka α_j određuju intezitet (stohastičke aproksimacije) gradijenta iz tačke $\vec{\theta}_j$. Ukoliko se uzima konstantna vrednost, proces će konvergirati sa nesavladivom greškom. Robins-Monroovi uslovi (4.10) obezbeđuju konvergenciju:

$$\sum_{j=1}^{\infty} \alpha_j = \infty \quad \sum_{j=1}^{\infty} \alpha_j^2 < \infty \quad (4.10)$$

Međutim, metoda optimizacije *Adam* koji je trenutno najzastupljeniji u modernim neuron-skim mrežama na osnovu istorije obučavanja automatski prilagođava dužinu koraka, potrebno je samo na početku postaviti inicijalnu dužinu (engl. *learning rate*), a preostali deo posla prepustiti metodi optimizacije.

4.4.3 Regularizacija kod neuronskih mreža

Osnovni cilj regularizacije je protivteža preteranom prilagođavanju modela podacima (tzv. preprilagođavanje). Regularizacija podrazumeva širok spektar tehnika pomoću kojih se sprečava gubitak sposobnosti modela da dobro generalizuje usled prevelikog adaptiranja skupu za obučavanje. U podacima se mogu pojaviti šum (engl. *noise*) i odudarajući podaci (engl. *outliers*). Šum predstavlja neku grešku u podacima koja može biti prouzrokovana na različite načine. Može nastati kao greška pri sakupljanju podataka recimo zbog alata koji se koriste u merenju ili od strane čovekovog uticaja. Odudarajući podaci su oni koji po svojim vrednostima atributa nisu slični ostalim podacima koji se nalaze u skupu. Ako model uči parametre kroz optimizacioni proces tako da minimalno greši na skupu za obučavanje, šum i odudarajući podaci će takođe biti naučeni. Modeli koji imaju puno parametara imaju tendenciju preprilagođavanja jer na neki način pokušavaju da memorišu skup pomoću obimnog prostora za beleženje informacija o zavisnosti ulaza i izlaza. Sve to utiče nepovoljno na ponašanje modela na nekom drugom skupu iste strukture podataka, ali drugačijih vrednosti. Tada kažemo da model ne uspeva da generalizuje van skupa za obučavanje, što se suprosvavlja cilju mašinskog učenja. Postoji mnoštvo metoda regularizacije, a ovde će biti navedena osnovna varijanta i one koji su korišćene u radu.

Jedan način za regularizaciju je korišćenje regularizacionog izraza. Pomenutoj funkciji greške $\mathcal{L}$, moguće je dodati izraz $\Omega(\vec{\theta})$ pomnožen sa skalirajućim faktorom $\lambda \in \mathbb{R}^+$. Ova metoda se može koristiti u raznim modelima mašinskog učenja. Izraz $\Omega(\vec{\theta})$ treba odabrati tako da ispuni svoju ulogu protivteže preprilagođavanju. Najčeće korišćen regularizacioni izraz je kvadrat ℓ_2 norme. Problem minimizacije funkcije greške dobija oblik:

$$\min_{\vec{\theta}} (\mathcal{L}(\vec{\theta}) + \frac{\lambda}{2} \|\vec{\theta}\|_2^2) \quad (4.11)$$

Hiperparametar λ treba obazrivo odabrati jer on kontroliše uticaj regularizacionog izraza. Ukoliko je λ veoma malo, minimizacija je usmerena ka funkciji greške, što nas vraća na početne probleme preprilagođavanja. U slučaju da je λ jako veliki broj, daje se previše značaja regularizacionom izrazu. Konkretno, kod kvadrata ℓ_2 norme, svaki pokušaj da parametar udaljimo od 0 puno se kažnjava. Kada $\lambda \rightarrow +\infty$, svi parametri bi morali biti 0. Kao i u mnogim aspektima mašinskog učenja, u praksi se isprobavaju različite vrednosti i utvrđuje se koja najbolje radi. Još jedan regularizacioni izraz koji se koristi je ℓ_1 norma. U tom slučaju proces optimizacije funkcije greške dobija oblik:

$$\min_{\vec{\theta}} (\mathcal{L}(\vec{\theta}) + \frac{\lambda}{2} \|\vec{\theta}\|_1), \quad (4.12)$$

gde je $\|\vec{\theta}\|_1 = \sum_i |\theta_i|$. U odnosu na ℓ_2 regularizaciju, regularizacioni izraz ℓ_1 nije diferencijabilan. Međutim, u praksi se pokazuje da to ne predstavlja veliki problem jer optimizacioni algoritam retko nailazi na tačke u kojima funkcija greške nije diferencijabilna. Ukoliko se to ipak desi, pogrešan pravac pri minimizaciji se uglavnom nadoknadi daljim tokom optimizacije. Glavna razlika ℓ_1 regularizacije u odnosu na ℓ_2 je u tome što ℓ_2 teži smanjivanju apsolutnih vrednosti

nekih koeficijenata koji postanu mali, ali ipak različiti od nule, kod ℓ_1 regularizacije manje važni koeficijenti postanu baš jednaki nuli. Ipak, problem nediferencijalnosti može postati ozbiljan ukoliko funkcija greške nije diferencijabilna u tački minimuma. Zbog takvih problema, potrebno je koristiti posebne optimizacione algoritme [23].

Veoma rasprostranjena metoda regularizacije koja se koristi kod neuronskih mreža je *regularizacija izostavljanjem* (engl. *dropout*) [30]. Tokom optimizacionog procesa, neki neuroni se gase (postavljaju na 0) sa određenom verovatnoćom čime se povećava robusnost mreže. Nije poželjno da mreža nauči koncepte na osnovu samo nekih detalja i da se osloni previše samo na određene neurone, već da uključuje više neurona u svojem predviđanju.

Intuicija prethodnog pasusa može se naći u realnom svetu. Nastavnik koji želi da što bolje nauči učenike iz odeljenja kojem predaje, trudiće se da uključi što više učenika pri kolektivnom ispitivanju. Ukoliko se javljaju samo dva učenika koji sve znaju, nastavnik bi njima mogao da zabrani da odgovaraju i prepusti to ostatku odeljenja. Time bi se i ostali aktivirali što doprinosi većem kolektivnom znanju. Sličan princip se koristi i kod regularizacije izostavljanjem.

Još jedan vid regularizacije koji je korišćen u ovom radu je *rano zaustavljanje* (engl. *early stopping*) koje će biti opisano u narednom odeljku.

4.4.4 Podešavanje hiperparametara

U mašinskom učenju, *hiperparametar* predstavlja promenljivu čija je vrednost određena pre procesa učenja. Nasuprot tome, *parametar* je promenljiva čija se vrednost menja tokom procesa obučavanja kako bi se kreirao što bolji model (u odnosu na neku meru kvaliteta). Primer hiperparametra je topologija mreže (dubina – broj slojeva, širina - broj neurona u svakom sloju), a kasnije će ih biti opisano još. Tipični parametri su težine u težinskim matricama slojeva i vektori slobodnih članova, međutim javljaju se i drugi.

Pre obučavanja modela određuju se vrednosti hiperparametara. Neki od njih su: dubina (broj slojeva), širina mreže (broj neurona po sloju), početni korak u optimizaciji (*learning rate*), verovatnoća u regularizaciji izostavljanjem, broj instanci na kojima se računa aproksimacija gradijenta u stohastičkoj varijanti (engl. *batch size*) i tako dalje. Sve su to primeri hiperparametara koje treba pažljivo odabrati pre obučavanja mreže. Naravno, jedan pristup se sastoji u pravljenju svih kombinacije hiperparametara i prostom proverom utvrdi koja najbolje radi. Međutim, ovaj pristup dovodi do kombinatorne eksplozije što u velikom broju slučaja ne može biti podržano od strane računara. Ako obučavanje mreže traje par sati, ispitivanje previše eksperimenata iziskuje puno vremena osim ako se ne pokreću istovremeno.

Traženje optimalnih hiperparametara može biti jako zahtevan problem koji se ne može rešiti u realnom vremenu. U praksi, obično je zadovoljavajuće imati dovoljno dobar izbor hiperparametara. Poslednje tri decenije, heuristike su postale veoma popularan izbor kao metode koje se koriste pri nalaženju suboptimalnih rešenja problema optimizacije u prikladnom vremenu. Za razliku od egzaktnih algoritama optimizacije, heuristike ne garantuju optimalnost dobijenih rešenja, ali je njihova glavna prednost brzina kojom se nalazi dovoljno kvalitetno rešenje. Korišćenje heuristika se može upotrebiti u podešavanju hiperparametara [31].

U poglavlju eksperimentalne evaluacije (videti odeljak 6.3) biće detaljnije opisan postupak traženja adekvatnih vrednosti hiperparametara. Veoma je bitno da se njihovo podešavanje ne vrši na skupu koji će biti korišćen za ocenjivanje modela. Test skup ne sme ni na koji način da utiče na formiranje modela. Njegova uloga je samo da na kraju oceni koliko model dobro

radi. Moguće podeliti skup koji je predviđen za obučavanje na dva dela. Prvi zaista služi za obučavanje modela, dok drugi nazivamo *validacioni skup* kojim proveravamo kako model radi za različite vrednosti hiperparametara. Kako je cilj generalizacija, želja je da model dobro radi ne samo na skupu za obučavanje, već i na nekom drugom do tada neviđenom skupu, odnosno novim podacima. Postoje i različiti načini vršenja validacije, odnosno provere kako rade različite varijante modela na drugom skupu koji nije korišćen za obučavanje.

U ovom radu korišćena je metoda *ranog zaustavljanja* koja je takođe i vid regularizacije. Proces obučavanja modela na skupu za obučavanje traje dok god se određena mera kvaliteta poboljšava na validacionom skupu. Ukoliko prođe određen broj epoha (engl. *patience*), a ta mera nije poboljšana, uzima se poslednji najbolji model nakon koga više nema poboljšanja. Dobijeni modeli se porede na osnovu najbolje mere kvaliteta i tako izabere najbolja konfiguracija hiperparametara. Postojanje kriterijuma ranog zaustavljanja se vrši radi rešavanja problema prilagođavanja i zbog *robustnosti sistema*, poželjno je da male promene parametra ne prouzrokuju velike oscilacije sistema, što je mera robustnosti.

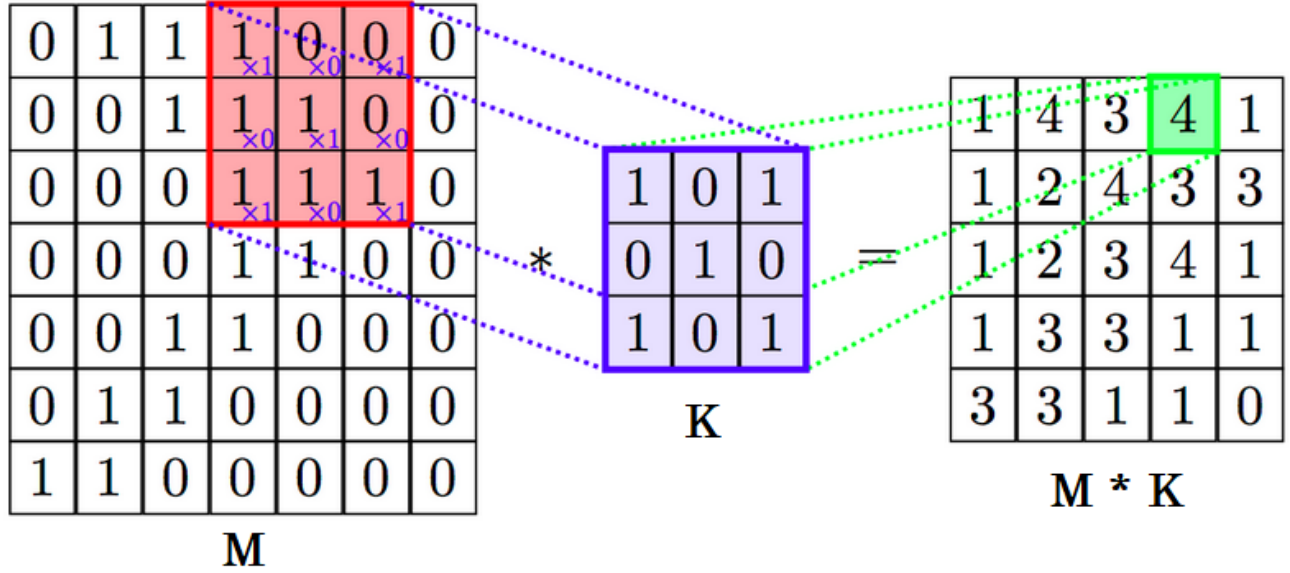
4.5 Konvolutivne neuronske mreže

Tradicionalne potpuno povezane neuronske mreže se odlikuju velikom gustinom veza jer svaki njihov tekući sloj sadrži parametre za sve jedinice iz prethodnog sloja, što rezultira rapidnim rastom broja parametara mreže pri povećavanju broja slojeva i neurona. Konvolutivne neuronske mreže (engl. *Convolutional Neural Networks*), skraćeno *CNNs* ili *ConvNets* su karakterisane primenom operatora konvolucije i manjim brojem veza između neurona prethodnog sloja ka tekućem. Tipična topologija za podatke podrazumeva strukturu mrežaste rešetke ili sekvence u jednodimenzionalnom slučaju. Standardni primeri za to su slike, zvuk, video zapis i vremenske serije. Ovde će biti opisani jednodimenzioni i dvodimenzioni slučaj, ali se ta razmatranja mogu uopštiti po analogiji.

Arhitektura potpuno povezane neuronske mreže podrazumeva ulaze u obliku vektora koji je jednodimenzionalan. Predstavljanje slike u tom obliku prouzrokuje gubljenje informacije o topologiji signala. Naime, uglavnom se redovi (kolone) piksela nadovežu u jedan dugački vektor. Kod slika velikih dimenzija se pojavljuje ogroman broj veza između neurona, a njihov redosled ne nosi korisnu informaciju jer dva piksela susedna u koloni (redu) su dosta udaljena u nadovezanom vektoru. Konvolutivne mreže se zasnivaju na ideji ekstrahovanja informacije u svakom narednom sloju, počevši od ulaznog vektora formiranog od sirovog signala, čime se gradacijski detektuju sve složeniji oblici na osnovu prethodnih slojeva. Ovakve operacije počivaju na matematičkom operatoru konvolucije.

4.5.1 Konvolutivni slojevi

Neka je data matrica $\mathbf{M} \in \mathbb{R}^{h \times w}$ i matrica $\mathbf{K} \in \mathbb{R}^{n \times m}$, gde je $n < h$ i $m < w$. Matricu $\mathbf{K}$ nazivamo *filterom*. Rezultat primene konvolucije filtera $\mathbf{K}$ na matricu $\mathbf{M}$ je takođe matrica, označena sa $\mathbf{X}$, čiji se elementi intuitivno gledano dobijaju kretanjem filtera po matrici i računanjem skalarnih proizvoda filtera $\mathbf{K}$ sa svim podmatricama od $\mathbf{M}$ dimenzije $n \times m$.



Slika 4.3: Primer izračunatog skalarnog proizvoda filtera **K** i podmatrice od **M** čiji se rezultat skladišti na odgovarajućem polju novonastale matrice dobijene primenom operatora konvolucije [32].

Precizno matematički:

$$X_{kl} = (M * K)_{kl} = \sum_{i=1}^n \sum_{j=1}^m K_{ij} \cdot M_{k+i-1, l+j-1} \quad (4.13)$$

Svaki element rezultujuće matrice **X** sadrži skalarni proizvod odgovarajuće podmatrice od **M** dimenzije $n \times m$ i filtera **K**. Ilustracija primene operatora konvolucije prikazana je na slici 4.3. Ipak, ovo nije pravi matematički operator konvolucije već *unakrsne korelacije* (engl. *cross-correlation*), ali su oni za potrebe mašinskog učenja suštinski isti. Dimenzije tako izračunate matrice **X** su jednake $(h-n+1) \times (w-m+1)$. Međutim, česta je praksa da se uradi transformacija tako da se dimenzije matrice očuvaju. Tu transformaciju nazivamo *proširivanje* (engl. *padding*) koje podrazumeva dodavanje dovoljno nula ravnomerno po ivicama početne matrice **M** tako da konvolucija proizvede matricu istih dimenzija (videti sliku 4.5 levo). Tipično se koriste nule, ali mogu i druge vrednosti. Filter se još naziva i *kernel*, ali se ipak ovde ne koristi taj naziv zbog njegovog višestrukog značenja u raznim kontekstima. Filter se po matrici može pomerati sa korakom 1 ili sa nekim drugim fiksiranim *korakom konvolucije* (engl. *stride*) [33].

Slika može biti predstavljena u obliku više kanala, recimo u *RGB modelu*. Tada postoji matrica realnih vrednosti za svaku od nijansi crvene, zelene i plave. U tom slučaju se za svaki kanal definiše posebno filter. Tako nastaje *tenzorski filter* (engl. *kernel tensor*) koji je dimenzija $3 \times m \times n$, odnosno u njemu se skladište 3 filtera dimenzije $m \times n$. Za svaki izlazni kanal postoji po jedan tenzorski filter. Više izlaznih kanala zahteva tenzorski filter za svaki od njih. Pored operacija konvolucije, kanali imaju i svoje slobodne članove kao i aktivacionu funkciju pomenutu ranije. Prolaskom kroz ove dve transformacije dobijaju se takozvane atributske mape (engl. *feature maps*). Nad njima se obično dalje primenjuje agregacija (engl. *pooling*) koja smanjuje dimenziju te matrice (videti odeljak 4.5.2), a zatim se mogu opet primenjivati operatori konvolucije.



Slika 4.4: Filteri za prepoznavanje jednostavnih (**levo**) i kompleksnih oblika (**desno**)[34].

Može se primetiti da operator konvolucije uzima u obzir povezanost piksela u slučaju slika. Zatim, naredni slojevi konvolucije uzimaju u obzir bliskost atributa formirane atributske mape i tako dalje.

Konvolutivne neuronske mreže takođe imaju svoje neurone. Veze su proređene tako da svaki neuron u narednom sloju kao ulaze prima okvir od $n \times m$ neurona iz prethodnog sloja, gde su n i m dimenzije filtera. Jedna od značajnih osobina konvolucije je *deljenje parametara*. Svaki filter sadrži $n \times m$ parametara koji predstavljaju težine svih neurona u posmatranom sloju za jedan kanal. Ceo jedan sloj za jedan izlazni kanal ima samo $n \times m$ parametara koji su zajednički za sve te neurone. Naredni slojevi i kanali imaju svoje filtere. Učenje težina filtera zapravo je osnovni zadatak konvolutivnih neuronskih mreža i izvršava se optimizacijom kao i pre.

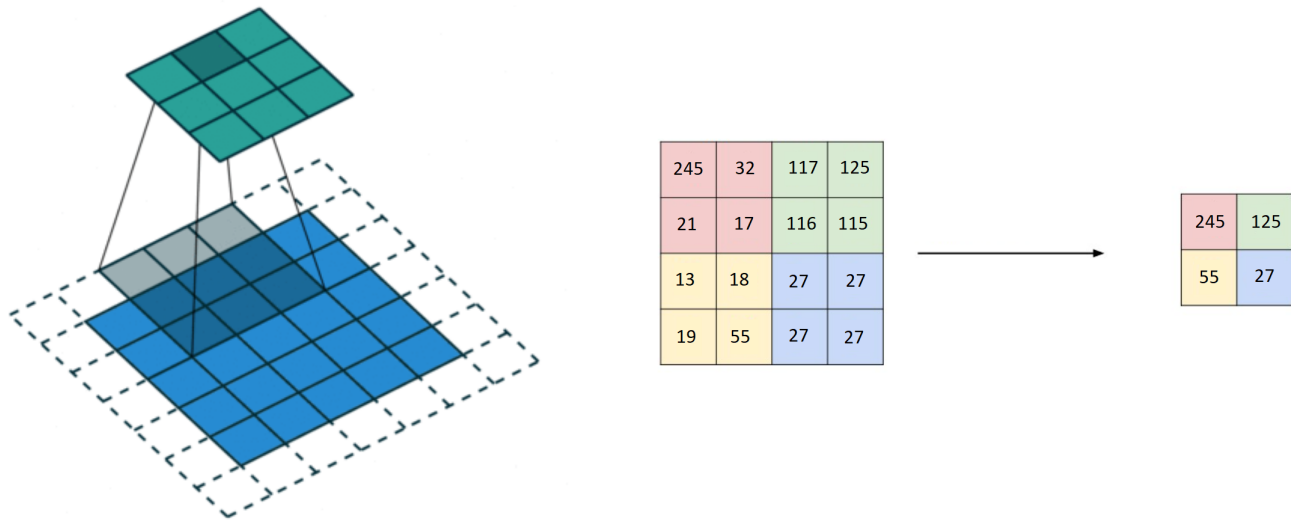
Pre upotrebe u mašinskom učenju, konvolutivni filteri su konstruisani ručno. Jedna od najmoćnijih osobina konvolutivnih neuronskih mreža je upravo automatska konstrukcija filtera tokom obučavanja, a potom njihovo korišćenje u predviđanju. Zanimljivo svojstvo kod slika je da su filteri u niskim slojevima mreže specijalizovani gotovo uvek da prepoznaju najjednostavnije strukture kao što su horizontalne, kose i vertikalne crtice (videti sliku 4.4 levo), dok filteri u višim slojevima prepoznaju kompleksnije oblike (videti sliku 4.4 desno). Prepoznavanje proizilazi iz toga da se konvolucijom maksimalne vrednosti dobijaju kada se upravo vrednosti filtera javu u podmatrici koja se filterom množi. Odnosno, aktivacija neurona je vrlo visoka ako je na slici pronađen deo slike dimenzije $n \times m$ koji ima vrednosti bliske vrednostima težinama filtera na odgovarajućim mestima. Uzrok tome je osobina skalarnog proizvoda. Naime, problem:

$$\max_{\|\vec{w}\|=1} \vec{w} \cdot \vec{x}$$

ima rešenje $\vec{x} = \vec{w}$, ako je $\vec{x}$ jedinični normirani vektor. Iako se često ističe da neuronske mreže nisu interpretabilni modeli, prethodno razmatranje ipak naglašava delimičnu intepretabilnost konvolutivnih neuronskih mreža. Naime, u prvim slojevima se filteri mogu vizuelizovati tako što se prikažu njihove vrednosti matrica. Tako se dobija oblik koji taj filter pronalazi na slici. Moguće je to postići i sa filterima narednih slojeva, ali to obično podrazumeva kompleksnije probleme optimizacije.

Još jedna vrlina konvolutivnih mreža je *translatorna invarijantnost*. Mreža pokušava da nauči filtere koji prepoznaju određene oblike, ali nevezano za poziciju na kojoj se nalaze.

4.5.2 Agregacioni slojevi

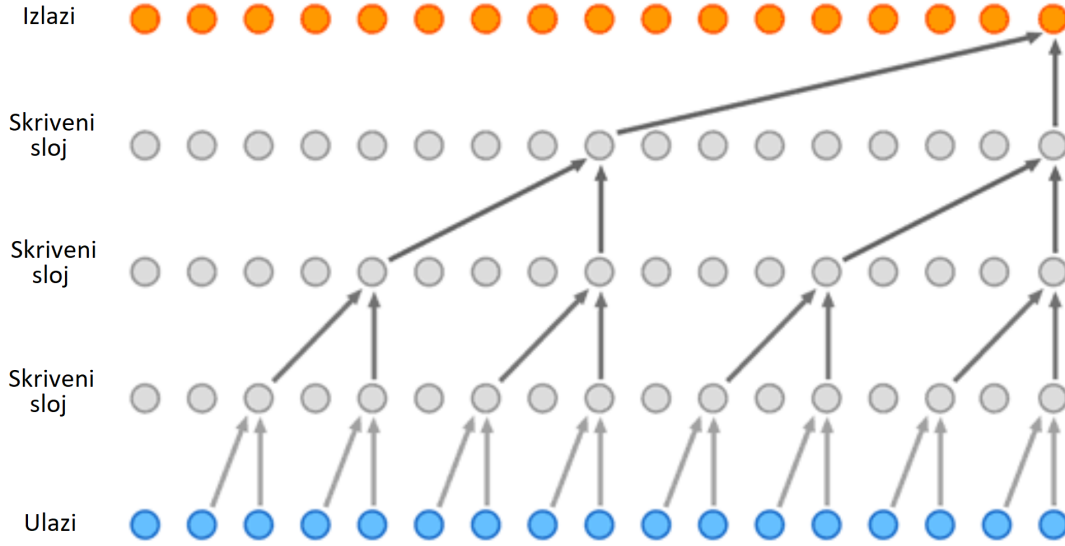


Slika 4.5: Konvolucija filterom 3×3 nad matricom 5×5 (**levo**) i agregacija računanjem maksimuma u podmatricama 2×2 (**desno**) [34].

Nakon što se konvolucijom formiraju atributske mape, može se koristiti *agregacija* koja smanjuje dimenzije matrice na koju je primenjena. To se ne radi u slučaju da je bitno da informacija izvučena konvolucijom ostane potpuno očuvana. Agregacija ukupnjuje informacije dobijene konvolucijom. Region koji je obuhvaćen delovanjem konvolucije ili agregacije zovemo *receptivno polje* (engl. *receptive field*). U slučaju korišćenja konvolutivnih slojeva isprepletenih agregacijom ostvaruje se eksponencijalan rast receptivnog polja. Još jedna prednost agregacije je značajno opadanje broja parametara sa njihovim korišćenjem što povoljno utiče na problem prilagođavanja. Najzastupljeniji oblik agregacije je računanje maksimuma u podmatricama 2×2 koje particionišu matricu na kojoj se ova operacija primenjuje. Tipično se uzima maksimalna vrednost, jer bi uprosečavanjem informacija bledela. Na slici 4.5 desno je prikazana agregacija računanjem maksimuma u podmatricama 2×2 . Ako konvolucija prepoznaje određene obrasce u nižim slojevima, korisno je da se ta informacija propagira u narednim slojevima. Na primer, ako je na slici prepoznat točak, to znači da je aktivirana visoka vrednost neurona filtera koji prepoznaje oblik točka. Ako u podmatrici 2×2 postoji visoka vrednost za pronalaženje točka, agregacijom pomoću maksimuma će ona opstati i u sledećem sloju. Na kraju, ako je poznato da je na slici točak, farovi, hauba, vrlo je verovatno da je na slici automobil. Kao što je pomenuto, agregacija nije uvek najbolji izbor sažimanja informacije. Primer za to je upravo rad na predviđanju funkcija proteina. Ako ukupnimo informaciju nekoliko segmenta proteinske sekvence, može se desiti da je baš u tom delu neka specifična mutacija koja će promeniti funkciju.

4.5.3 Dilatacijske konvolucije

Umesto standardnih konvolucija koje deluju na podmatrice, definišu se konvolucije sa *dilatacijom* odnosno razmakom između svaka dva susedna polja filtera [35]. Standardni operator konvolucije koji je definisan jednakošću 4.13 ima vrednost dilatacije 1. Veličina dilatacije



Slika 4.6: Jednodimenzionalne dilatacijske konvolucije kod modela *WaveNet* [36]

predstavlja hiperparametar neuronske mreže. Postupak kojim se dobija ekponencijalan rast receptivnog polja sastoji se od ekponencijalnog povećavanja dilatacije u svakom narednom sloju (obično se uzima stepen dvojke, tada su vrednosti dilatacije redom 1, 2, 4, 8, 16...). Glavna prednost korišćenja dilatacijskih konvolucija naspram agregacije je očuvanje rezolucije i pokrivenosti signala.

Uopštenje operatora konvolucije sa korakom dilatacije može se matematički zapisati na sledeći način:

$$X_{kl} = (M *_d K)_{kl} = \sum_{i=1}^n \sum_{j=1}^m K_{ij} \cdot M_{k+(i-1) \cdot d, l+(j-1) \cdot d} \quad (4.14)$$

pri čemu je filter i dalje matrica dimenzija $n \times m$, dok njegovo dejstvo zavisi od odabira vrednosti dilatacije d . Tipično se koristi isto d za vrste i kolone, pa je ovde to podrazumevano u definiciji. Neka su dimenzije matrice $\mathbf{M}$ označene sa $h \times w$, tada su k i l u jednakosti 4.14 iz skupova $k \in \{1, 2, \dots, h - (n - 1) \cdot d\}$ i $l \in \{1, 2, \dots, w - (m - 1) \cdot d\}$.

Jedan od reprezentativnih modela koji koristi dilatacijske konvolucije je *WaveNet* [36] koji je doprineo drastičnom poboljšanju prevođenja teksta u mašinski govor. Uz pomoć skupa za obučavanje koji može sadržati veliki broj rečenica koje je izgovorio čovek, moguće je oponašati govor na mašinski način koji zvuči prirodno prema oceni ispitanika. Takođe je moguće generisati klasičnu muziku na klaviru uz pomoć puno klavirskih dela datih modelu za obučavanje.

Wavenet predstavlja neuronsku mrežu koja radi sa sirovim signalom predviđajući narednu frekvenciju pomoću prethodnih. Mreža je u potpunosti konvolutivna sa dilatacionim faktorom koji se ekponencijalno povećava, stvarajući ekponencijalni rast receptivnog polja (videti sliku 4.6). Skup podataka za obučavanje sadrži rečenice za koje se formira zvučni signal snimljenog izgovora odabranog čoveka. Nakon toga se mogu generisati fragmenti govora (videti sliku 4.7), a uz dostavljeni tekst i oponašati način na koji to govornik izgovara.

Konvolutivna neuronska mreža je sastavljena od konvolutivnih slojeva koji mogu biti praćeni agregacijom, a na kraju se tipično nadovezuje jedan potpuno povezani sloj ili više njih kako bi



Slika 4.7: Grafik sekunde generisanog govora [36].

se atributi konstruisani konvolucijama upotreбили za dobijanje konačnog izlaza. Konvolucije ekstrahuju najbitnije informacije iz podataka u formi atributa koji predstavljaju podlogu za određen broj potpuno povezanih slojeva.

4.6 Grafovske neuronske mreže

Postoji mnoštvo vrsta podataka koji se mogu predstaviti računaru grafovskom strukturom. *Grafovske neuronske mreže* (engl. *graph neural networks*) ili skraćeno *GNNs* obrađuju upravo takve podatke. Njihove primene se sreću u hemiji, biologiji, medicini, društvenim mrežama, transportnim sistemima, u simulaciji saobraćaja i autonomnoj vožnji, u sistemima za preporučivanje sadržaja i svuda gde je prisutna grafovska struktura. *GNNs* se zasnivaju na ideji korišćenja veza između čvorova kako bi se dobile bolje performanse modela. Neki podaci imaju veoma pravilnu grafovsku strukturu, kao što su slike (svaki piksel ima 3, 5 ili 8 susednih piksela), dok je kod nekih graf veoma heterogen. Konvolutivne neuronske mreže su se pokazale efikasne kod slika, ali je potrebno proširiti metode i na kompleksnije grafove, što predstavlja još veći izazov.

Neka je dat graf za čije čvorove se na osnovu atributa vrši predviđanje ciljne promenljive. Na ulazu grafovske neuronske mreže data je matrica sa vrednostima atributa za svaki čvor (engl. *node features*) $\mathbf{F} \in \mathbb{R}^{m \times k}$, gde m predstavlja ukupan broj čvorova, a k fiksna broj atributa za svaki čvor. Početni atributi su obično dati za svaki čvor, međutim u ovom radu su naučeni neuronskom mrežom zbog specifičnosti problema predviđanja funkcije proteina (detaljnije u poglavlju 5). Data je i matrica susedstva $\mathbf{A} \in \mathbb{R}^{m \times m}$. Grane između čvorova ukazuju da postoji određena zavisnost između vrednosti ciljnih promenljivih za dva čvora koji dele granu. Radi jednostavnosti, uvodi se pretpostavka da *ivice grafa nisu otežane* i *graf nije usmeren* odnosno da je $\mathbf{A}$ simetrična matrica binarnih vrednosti. U zavisnosti da li je data vrednost ciljne promenljive za čvorove koji se nalaze u skupu za obučavanje razlikuju se problemi nadgledanog i nenadgledanog učenja.

Kod problema nadgledanog učenja, za sve čvorove koji su u skupu za obučavanje modela je data prava vrednost ciljne promenljive. Potrebno je kreirati model koji će predviđati vrednost ciljne promenljive za podatke izvan skupa za obučavanje.

Najjednostavniji pristup rešavanja zadatka mašinskog učenja kod podataka koji imaju grafovsku strukturu je potpuno zanemiravanje matrice susedstva. U tom slučaju, ne postoji korišćenje grafovskih neuronskih mreža već se koriste neki drugi modeli.

Za rad sa usmerenim acikličkim grafovima su na početku korišćene rekurzivne neuronske mreže (engl. *recursive neural networks*). Modeli *GNN* su uvedeni kao generalizacija rekurzivnih neuronskih mreža koji mogu da se primenjuju i na ostale vrste grafova. Oni se sastoje od iterativnog procesa kojim se propagiraju stanja čvorova do ravnotežnog stanja, nakon čega sledi

neuronska mreža koja na osnovu stanja generiše izlaz. Poseban pravac obuhvata skup metoda baziranih na uopštavanju konvolutivnih neuronskih mreža na grafove.

4.6.1 Grafovski atributi zasnovani na slučajnom hodu

Ideja slučajnog hoda se zasniva na tome da pridruži početnim atributima čvorova dodatne attribute (u nastavku: *grafovske attribute*) izvedene isključivo iz matrice susedstva. Pridruživanje se postiže nadovezivanjem grafovskih atributa i početnih atributa čvorova, u oznaci: $\vec{f}_i \parallel \vec{\Phi}_i$, gde $i = \overline{1, m}$. Rezultat ovog nadovezivanja predstavlja vektor atributa za svaki čvor i .

Metoda *slučajnih hodova* u grafu (engl. *random walks*) ima za cilj izvođenje informacije o čvorovima analiziranjem njihove grafovske strukture. Prvi metod koji se pokazao uspešnim u primeni slučajnih hodova, zasnovan na dubokim neuronskim mrežama je *DeepWalk* [37].

DeepWalk se sastoji iz dve etape. U prvoj, iz svakog temena u grafu polaze slučajni hodovi radi analiziranja lokalnih struktura u grafu odnosno susedstva za svaki čvor. U drugoj etapi, *DeepWalk* koristi algoritam *SkipGram* kako bi naučio reprezentacije za čvorove na osnovu rezultata iz prve etape. Reprezentacije su fiksne dužine k , pri čemu generalno pravilo kome se teži je da čvorovi koji su bliski u grafu (susedni čvorovi ili postoji kratak put između njih) imaju slične reprezentacije. Naslednici metode *DeepWalk* poput *node2vec* [38], *struc2vec* [39] i *LINE* [40] se zasnivaju na istoj ideji, ali različitom pristupu kreiranja slučajnih hodova.

Ukoliko informacija o početnim atributima čvorova ne postoji, *DeepWalk* je metoda koja je pogodna za probleme nenagledanog učenja. Ukoliko su poznati početni atributi, primer algoritma koji efikasnije radi je *Plantoid* [41].

4.6.2 Grafovske konvolutivne mreže

Ideja grafovskih konvolutivnih mreža [42] (engl. *graph convolutional networks*), ili skraćeno *GCN*, je ponikla iz želje da se proslavljeni konvolutivni pristup primeni na grafove. Za svaki čvor se iterativno ažuriraju reprezentacije odnosno atributi čvorova, u zavisnosti od grafovske strukture.

Analogno konvolutivnim slojevima kod *CNN* mreža, grafovski konvolutivni sloj se dobija kombinovanjem atributa susednih čvorova u reprezentaciju višeg nivoa. Susedstvo za čvor i je označeno sa $\mathcal{N}_i$. U susedstvo je uključen i sam čvor. Formalno $\mathcal{N}_i = \{j \mid i = j \vee A_{ij} \neq 0\}$. Neka je g funkcija transformacije primenjena na susedstvo:

$$\vec{h}'_i = g(h_x, h_y, h_z, \dots) \quad x, y, z \dots \in \mathcal{N}_i \quad (4.15)$$

Generalizacija filtera naspram signala poput slika, gde su pikseli povezani pravilnom grafovskom rešetkom, nije trivijalan zadatak. Jedan od jednostavnijih načina da se to uradi je množenje atributa čvorova matricom susedstva. Matrica $\mathbf{H}$ predstavlja objedinjena stanja susedstva za svaki čvor tj. $\mathbf{H} = \{\vec{h}_1, \vec{h}_2, \dots, \vec{h}_m\}$, $\vec{h}_i \in \mathbb{R}^n$, gde je m broj čvorova, dok je n fiksna broj atributa za svaki čvor. Transformacijom trenutnih stanja se dobija nova matrica $\mathbf{H}'$:

$$\mathbf{H}' = \sigma(\mathbf{A}\mathbf{H}\mathbf{W}) \quad (4.16)$$

gde je $\mathbf{W} \in n \times n$ linearna transformacija koja je deljena među čvorovima i sadrži parametre koji se uče, dok je σ aktivaciona funkcija zadužena za unošenje nelinearnosti. Množenje matricom

susedstva se agregiraju stanja čvorova. Kako se ne bi izgubila glavna informacija o stanju čvorova, kao što je i ranije pomenuto, dodaje se veza čvora sa samim sobom tako da matrica susedstva ima oblik $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_m$. Kako se usled množenja matricom $\mathbf{A}$ u sumi pojavljuje onoliko sabiraka koliko ima čvorova u susedstvu, vrši se *normalizacija* kako bi red veličine ostao isti:

$$\mathbf{H}' = \sigma(\tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}} \mathbf{H} \mathbf{W}) \quad (4.17)$$

gde je $\tilde{\mathbf{D}}$ dijagonalna matrica stepena čvorova, odnosno važi $\tilde{D}_{ii} = \sum_{j=1}^m \tilde{A}_{ij}$ dok je ostatak matrice popunjen nulama. Tako se dolazi do *agregiranja prosekom* (engl. *mean-pooling*) što se zapisuje za pojedinačni čvor i kao:

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_i|} \mathbf{W} \vec{h}_j\right) \quad (4.18)$$

Umesto oblika (4.17), kod *GCN* modela se koristi *simetrična normalizacija*:

$$\mathbf{H}' = \sigma\left(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{1/2} \mathbf{H} \mathbf{W}\right) \quad (4.19)$$

odnosno, zapisano za svaki čvor:

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i| |\mathcal{N}_j|}} \mathbf{W} \vec{h}_j\right) \quad (4.20)$$

Pomenuti oblik u jednačinama (4.19) i (4.20) se koristi za radi balansiranog agregiranja uticaja u zavisnosti od broja suseda. Na primer, ukoliko je čvor i povezan sa čvorom j koji ima mnogo suseda i sa čvorom k iz kojeg polazi tačno jedna grana, više uticaja na i ispoljava čvor k kako je to jedina grana iz koje on deluje. Ovaj vid agregacije povoljno utiče na problem prilagođavanja i vrlo je jednostavan. Jedna od mana grafovskih konvolutivnih mreža je što nisu primenljive na grafove koji se dinamički menjaju.

4.6.3 Neuronske mreže sa prenošenjem poruka

Neuronske mreže sa prenošenjem poruka (engl. *message-passing neural networks*) [43] ili skraćeno *MPNN* poboljšavaju *GCN* model koji indirektno podržava samo attribute ekstrahovane na osnovu grana u grafu. Umesto toga, moguće je koristiti attribute susednih čvorova ukombinovane sa težinama ili vektorom atributa grana kako bi se proizveo vektor poruke (engl. *message vector*) ka posmatranom čvoru. Tada svaki čvor agregira vektore poruka iz susedstva. Preciznije, definiše se:

$$\vec{m}_{ij} = f_e(\vec{h}_i, \vec{h}_j, \vec{e}_{ij}) \quad i, j \in \{1, 2, \dots, m\} \quad (4.21)$$

gde $\vec{m}_{ij}$ predstavlja vektor poruke iz i ka j koji se računa preko funkcije f_e koja za argumente prima redom attribute čvora iz kojeg se šalje uticaj, zatim attribute čvora koji prima uticaj i attribute grane između njih. Nakon toga se agregiraju svi vektori poruka pomoću funkcije f_v :

$$\vec{h}'_i = f_v\left(\vec{h}_i, \sum_{j \in \mathcal{N}_i} \vec{m}_{ji}\right) \quad (4.22)$$

Obično su f_e i f_v peceptroni ili male potpuno povezane neuronske mreže. Funkcija f_v kojom se dobijaju nova stanja vektora čvora se naziva *očitavajuća funkcija* (engl. *readout function*).

Neuronske mreže sa prenošenjem poruka imaju veću izražajnu moć od grafovskih konvolutivnih mreža jer osim atributa grana uključuju i attribute čvorova. Ipak, njihova mana je gomilanje parametara što povećava memorijsku zahtevnost i potrebu za optimizovanjem velikog broja parametara. U praksi se primenjuju na grafove manjih dimenzija od par stotina čvorova.

4.6.4 Grafovske mreže sa mehanizmom pažnje

S jedne strane *MPNN* predstavlja uopšteniji model od *GCN*-a, ali s druge strane pati od problema velikog broja parametara. *Grafovske neuronske mreže sa mehanizmom pažnje* [44] (engl. *graph attention networks*), skraćeno *GAT*, se nalaze na sredini pomenuta dva pristupa.

Novi pristup *GAT* arhitekture se zasniva na definisanju težina uticaja susedstva koje se uče tokom obučavanja modela. Prethodni metodi su uglavnom bili fokusirani na eksplicitnom zadavanju uticaja čvorova, na primer uprosečavanjem ili uzimanjem maksimuma. *GAT* je karakterisan mehanizmima unuturašnje pažnje (engl. *self-attention*). Prednost *GAT* modela je mogućnost rada sa varirajućom dužinom ulaza fokusiranjem na najbitnije delove za donošenje odluka. Pokazali su se jako dobro u mašinskom čitanju, učenju reprezentacije rečenica i mašinskom prevodenju. Rezultati su dali motivaciju za uvođenje arhitekture zasnovane na unutrašnjoj pažnji. Ideja se zasniva na računanju skrivenih reprezentacija svakog čvora u grafu kretanjem kroz susedstvo grafa primenjujući metodu unutrašnje pažnje. Glavne njene prednosti su mogućnost paralelizacije, mogućnost primene na čvorove različitih stepena, primenljivost u problemima gde se graf dinamički menja, uključujući probleme gde bi model trebalo da generalizuje na grafovima kompletno nepoznatim tokom obučavanja. Vremenska složenost je na istom nivou kao grafovske konvolutivne mreže.

4.6.5 GAT sloj

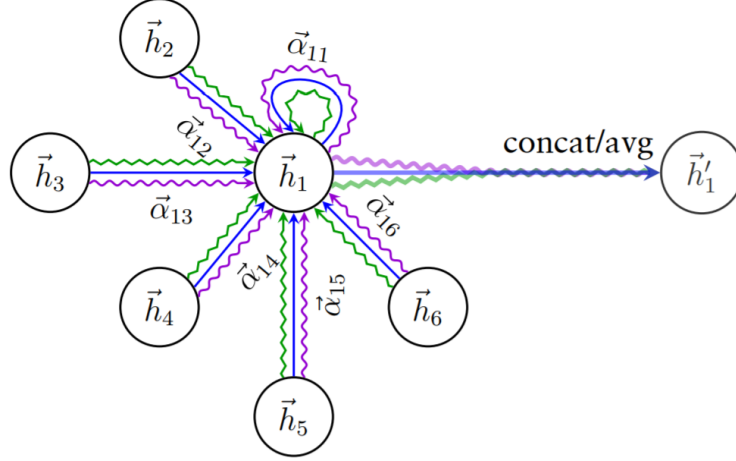
Ulaz u *GAT sloj* je skup atributa čvorova je matrica stanja čvorova $\mathbf{H}$. Na osnovu matrice $\mathbf{H}$ izvodi se novi skup atributa čvorova (potencijalno različite kardinalnosti) $\mathbf{H}' \in m \times n'$ koji predstavlja izlaz tog sloja.

Da bi se postigao efekat dubokog učenja, uvodi se linearna transformacija čiji se parametri uče tokom obučavanja modela. Za početak, neka je to matrica $\mathbf{W} \in \mathbb{R}^{n' \times n}$ koja je zajednička za sve čvorove. *Unutrašnji mehanizam pažnje* se definiše kao: $a : \mathbb{R}^{n'} \times \mathbb{R}^{n'} \rightarrow \mathbb{R}$ računanjem *koeficijenata pažnje* (engl. *attention coefficients*):

$$e_{ij} = a(\vec{h}_i, \vec{h}_j) \quad (4.23)$$

koji mere *relevantnost* atributa čvora j za čvor i . U najopštijoj formulaciji, model dozvoljava delovanje čvorova svaki sa svakim, zanemarujući sve uslove povezanosti. Grafovska strukutra se pridružuje mehanizmu sprovođenjem *maskiranih mehanizama pažnje*, odnosno računamo e_{ij} samo za j koji su iz susedstva i (sa oznakom $\mathcal{N}_i$). Koeficijenti se normalizuju mekim maksimumom kako bi red veličine bio isti sa koeficijentima drugih čvorova:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \quad (4.24)$$



Slika 4.8: Ilustracija višestrukog mehanizma pažnje primenjenog na susedstvo [44].

Mehanizam pažnje a koji se koristi je funkcija predstavljena potpuno povezanom neuronskom mrežom koja je parametrizovana vektorom $\vec{a} \in \mathbb{R}^{2n'}$, funkcija nelinearnosti je *LeakyReLU*. Tako se dobija:

$$\alpha_{ij} = \frac{\exp \left(\text{LeakyReLU} \left(\vec{a}^T [\mathbf{W} \vec{h}_i \parallel \mathbf{W} \vec{h}_j] \right) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left(\text{LeakyReLU} \left(\vec{a}^T [\mathbf{W} \vec{h}_i \parallel \mathbf{W} \vec{h}_k] \right) \right)} \quad (4.25)$$

gde $\parallel$ predstavlja operaciju konkatencije (nadovezivanja vektora). Nakon što su koeficijenti normalizovani, koriste se za računanje atributa čvorova u narednom sloju. To može biti izvedeno linearnom kombinacijom atributa čvorova sa težinama odgovarajućih koeficijenata na koju može biti primenjena nelinearna transformacija σ :

$$\vec{h}'_i = \sigma \left(\sum_{k \in \mathcal{N}_i} \alpha_{ik} \mathbf{W} \vec{h}_k \right) \quad (4.26)$$

Koeficijenti α_{ij} se računaju u skladu sa jednakošću (4.25) koristeći trenutne vrednosti parametara $\vec{a}$. Nakon što se vrednosti izlaza mreže uporede sa pravim vrednostima, formira se funkcija greške na osnovu koje se parametri $\vec{a}$ ažuriraju. Iterativni proces optimizuje istovremeno parametre matrice $\mathbf{W}$, kao i parametre $\vec{a}$.

Zarad stabilizacije procesa učenja, koristi se višestruki mehanizam pažnje (engl. *multi-head attention*). Preciznije, K nezavisnih mehanizama pažnje se izvršavaju u skladu sa jednakošću (4.26), a zatim se svi atributi nadovežu. Nadovezivanjem se za svaki čvor konstruiše Kn' atributa (umesto n' ako se ne koristi višestruki mehanizam pažnje). Ilustracija tri mehanizma pažnje označena različitim bojom prikazana je na slici 4.8, gde se agregacijom ili nadovezivanjem se dobija $\vec{h}'_1$. Specijalno, ako se višestruki mehanizam pažnje primenjuje na poslednjem sloju, nije potrebno vršiti nadovezivanje, već uprosečavanje, nakon čega se primenjuje aktivaciona funkcija.

$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{k \in \mathcal{N}_i} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \vec{h}_j \right) \quad (4.27)$$

Koeficijenti $\alpha_{ij}^{(k)}$ i matrica parametara $\mathbf{W}^{(k)}$ se računaju za svaki mehanizam $k = \overline{1, K}$ posebno. Regresija ili klasifikacija se može vršiti na osnovu tako dobijenih atributa i/ili uz njihovu agregaciju zavisno da li se ta vrsta predviđanja vrši nad pojedinačnim čvorovima ili celim grafom.

Poglavlje 5

Predložena arhitektura

U ovoj glavi je opisano kako je reprezentacija podataka neuronske mreže predstavljena u računarstvu, a zatim je predložena arhitektura koja obuhvata uopšten model kao i model prilagođen problemu predviđanja funkcije proteina. Na kraju su opisane biblioteke programskog jezika *Python* korišćene u implementaciji.

5.1 Reprezentacija podataka

Prilikom definisanja reprezentacija podataka tipično je potrebno formulirati ulaze koje mreža može da obrađuje. U ovom radu je takođe i reprezentacija izlaza netrivialna jer se u izlazima pojavljuje graf.

5.1.1 Ulaz neuronske mreže

Kao ulazni podatak je korišćena samo proteinska sekvenca zbog akcenta na primeni grafovskih neuronskih mreža odnosno poboljšavanju osnovnog modela koji ne koristi grafovsku strukturu. Ostali relevantni atributi nisu bili dostupni u jednostavnom obliku, pa je njihovo uključivanje ostavljeno za dalji rad na ovom problemu.

Svaka aminokiselina se incijalno preslikava u tzv. *one-hot* reprezentaciju od 21 atributa od kojih tačno jedan ima vrednost 1 (u zavisnosti od tog o kojoj aminokiselini se radi), a ostali 0.

5.1.2 Izlaz neuronske mreže

Pomenuto je da se skup funkcija nekog proteina predstavlja kao konzistentni podgraf u redukovanoj ontologiji (videti sliku 3.3). Nazivi svih 123 čvora su poređani u određenom redosledu tako da se izlaz neuronske mreže za dati protein može pogodno predstaviti kao vektor sačinjen od nula i jedinica, pri čemu se na svakoj od 123 pozicije nalazi 1 ako je odgovarajući čvor u skupu funkcija datog proteina, a 0 u suprotnom. Predviđanje neuronske mreže je vektor dimenzije 123 sa realnim vrednostima iz interвала $[0, 1]$ koje predstavljaju verovatnoće u zavisnosti od ocene pouzdanosti modela pri predviđanju.

5.2 Predložena arhitektura

U ovom odeljku će biti opisan uopšteni model koji se može primeniti i na druge probleme slične strukture, a zatim specifičnosti modela za problem predviđanja funkcije proteina.

5.2.1 Uopšteni model

Neka su dati ulazi $x \in \mathcal{X}$ i vrednosti ciljnih promenljivih $y_i \in \mathcal{Y}_i$ (za i iz nekog skupa indeksa $\mathbb{L}$), gde je $\mathcal{Y}_i$ je odgovarajući kodomen. Data je matrica susedstva $\mathbf{A} \in \mathbb{R}^{|\mathbb{L}| \times |\mathbb{L}|}$ koja odgovara grafu povezanosti ciljnih promenljivih. Za matricu $\mathbf{A}$ se pretpostavlja da je binarna i simetrična (u opštoj varijanti se mogu postaviti težine grana), a proizvoljna grana $i \leftrightarrow j$ označava da pripadnost čvora i skupu funkcija datog proteina utiče na pripadnost j u skupu i obrnuto, odnosno ciljne promenljive y_i i y_j su korelisane. Neka je dato preslikavanje:

$$f : \mathcal{X} \rightarrow \mathcal{Z}_1 \times \mathcal{Z}_2 \times \cdots \times \mathcal{Z}_{|\mathbb{L}|} \quad (5.1)$$

tako da za svako $i \in \mathbb{L}$ funkcija f izračunava vektor reprezentacija $\vec{z}_i \in \mathcal{Z}_i$ na osnovu ulaza x , odnosno $f(x) = \mathbf{Z} = (\vec{z}_1, \dots, \vec{z}_{|\mathbb{L}|})$, čime se obezbeđuju reprezentacije za svaki čvor koje model može dalje da obrađuje. Specifično, mogu se uzeti k -dimenzioni vektori $\vec{z}_i$, tako da je $\mathcal{Z}_i = \mathbb{R}^k$.

Kako su ciljne promenljive strukturno povezane, sa $i \in \mathbb{L}$ su označeni čvorovi u grafu kojem je pridružena matrica susedstva $\mathbf{A}$. Pomenutim preslikavanjem f svaki čvor dobija reprezentacije na osnovu kojeg grafovska neuronska mreža g formira izlaze. Preciznije:

$$g : \mathbb{R}^{|\mathbb{L}| \times k} \times \mathbb{R}^{|\mathbb{L}| \times |\mathbb{L}|} \rightarrow \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_{|\mathbb{L}|} \quad (5.2)$$

Odnosno, konačni izlazi se dobijaju kao:

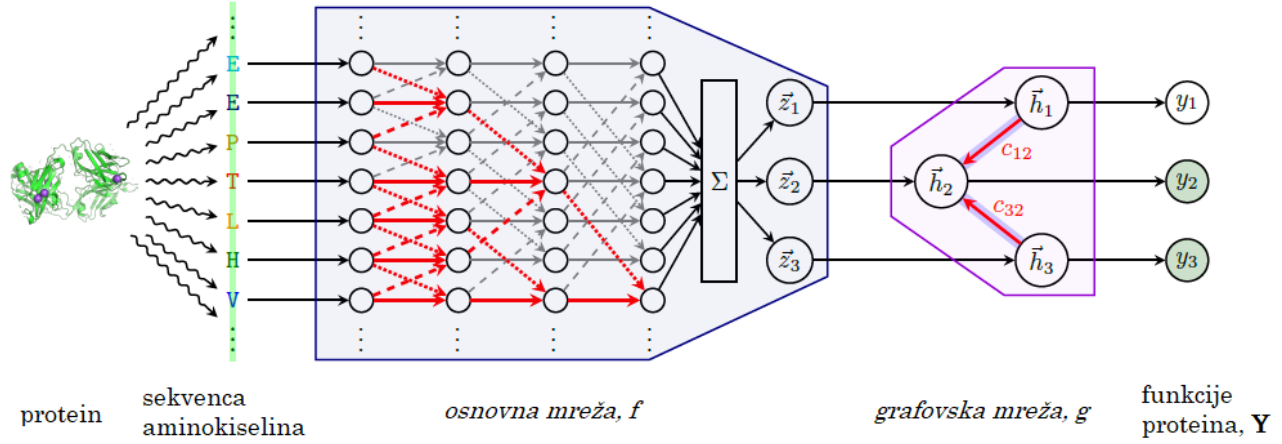
$$\mathbf{Y} = (y_1, \dots, y_{|\mathbb{L}|}) = g(\mathbf{Z}, \mathbf{A}) = g(f(x), \mathbf{A}) \quad (5.3)$$

Ukoliko su f i g diferencijabilne funkcije, kompozicija $g(f(x))$ koja predstavlja predviđanje višedimenzionalne ciljne promenljive za ulaz x se poredi sa njenim pravim vrednostima i na osnovu toga se formira funkcija greške koja se minimizuje tehnikama optimizacije (videti odeljak 4.4.2).

5.2.2 Model predviđanja funkcije proteina

Pomenuti model u prethodnom odeljku se može prilagoditi problemu predviđanja funkcije proteina [7]. Ulaz x je proteinska sekvenca za koju je poznat skup proteinskih funkcija. Svaki indeks i iz skupa indeksa $\mathbb{L}$ predstavlja čvor u ontologiji, odnosno svaki indeks je pridružen određenoj funkciji proteina.

Bitno opažanje je da se na osnovu simetričnosti matrice susedstva uvodi *pretpostavka da postoji veza u oba smera između roditeljskog čvora i njegovog direktnog potomka*. Jasno je da ukoliko je čvor u skupu funkcija za određeni protein, tada i roditelj tog čvora mora pripadati tom skupu zbog poštovanja pravila o konzistentnom podgrafu. Međutim, u prostoru svih proteinskih funkcija iz date ontologije, ukoliko je roditelj nekog čvora od strane modela označen kao funkcija proteina, to na neki način povećava verovatnoću da je i taj čvor u skupu funkcija datog proteina. Sledeći primer može učiniti ovu odluku intuitivnijom. Neka model predviđa vrednost blisku nuli



Slika 5.1: Predložena arhitektura grafovske neuronske mreže [7].

za neki čvor, odnosno da sa velikom pouzdanošću predviđa da taj čvor nije u skupu funkcija datog proteina. Posmatrajmo direktan potomak tog čvora za koji model tvrdi da jeste u skupu funkcija datog proteina sa malom pouzdanošću (na primer verovatnoća 0.51, dok je prag 0.5). Ukoliko model za još nekoliko roditelja od tog direktnog potomka predviđa verovatnoću blisku nuli, to implicira da je vrlo verovatno da je klasifikator pogrešio i da direktni potomak nije u skupu funkcija posmatranog proteina.

Osnovna mreža

Za preslikavanje f je uzeta potpuno konvolutivna mreža (bez agregacionih slojeva) sa dilatacijskim konvolucijama. U osnovnoj varijanti modela, koju ćemo nadalje zvati *osnovna mreža*, funkcija f direktno proizvodi izlaze:

$$\mathbf{Y} = (y_1, \dots, y_{|\mathbb{L}|}) = f(x) \quad (5.4)$$

Tada g nije definisana (ili se može reći da predstavlja identitet) što implicira da grafovske obrade nisu korišćene, odnosno matrica susedstva $\mathbf{A}$ je u potpunosti zanemarena. Kreiranjem *osnovne mreže* je određen početni model sa kojom će modeli koji koriste grafovske neuronske mreže moći da se uporede. Kako bi se pokazalo da grafovske neuronske mreže zaista donose poboljšanje, potrebno je pronaći što bolji model koji predstavlja *osnovnu mrežu* jer u suprotnom nadogradnja modela se može pokazati uspešnom samo zbog povećanog broja parametara. Detalji arhitekture koji su utvrđeni štimovanjem hiperparametara će biti obrađeni sledećem poglavlju.

Grafovska mreža

Sledeći korak u nadogradnji osnovnog modela je proširivanje izlaza preslikavanja f . Neka je u formuli (5.1) za sve $\mathcal{Z}_i = \mathbb{R}^k$ za svako $i = 1, \dots, |\mathbb{L}|$. Funkcija f i dalje predstavlja potpuno konvolutivnu neuronsku mrežu, ali se sada umesto skalara u izlaznom prostoru nalaze k - dimenzionalni vektori. Cilj ove neuronske mreže je da konstruiše reprezentacije za čvorove čime se postavlja temelj za primenu grafovske neuronske mreže.

Za funkciju g u ovom radu je predložena grafovska konvolutivna neuronska mreža GCN (videti odeljak 4.6.2), u kojoj za svaki čvor i postoje stanja $\vec{h}_i$ koja se ažuriraju tokom vremena sledećom formulom:

$$\vec{h}'_i = \text{ReLU} \left(\sum_{j \in \mathcal{N}_i} c_{ji} \mathbf{W} \vec{h}_j \right) \quad (5.5)$$

gde je $\mathcal{N}_i = \{j \mid j = i \vee A_{ij} = 1\}$ susedstvo koje uključuje i sam čvor, a $\mathbf{W}$ matrica parametara koje model treba da nauči. U zavisnosti od odabira koeficijenata c_{ji} dobijaju se različiti načini agregiranja stanja iz susedstva.

Tipovi agregacije stanja iz susedstva

U ovom radu, implementirane su sledeće vrste agregiranja informacije iz susednih čvorova:

- *sumiranje* (engl. *sum-pooling*) [45]
- *urosečavanje* (engl. *mean-pooling*) [3]
- *GAT agregacija* (engl. *graph attention pooling*) [44]
- *agregacija biranjem maksimuma* (engl. *max-pooling*)

Sumiranje se dobija odabirom $c_{ji} = 1$, dok se uprosečavanje konstruiše odabirom $c_{ji} = \frac{1}{|\mathcal{N}_i|}$. Kod *GAT agregacije* se uzima $c_{ji} = a(\vec{h}_i, \vec{h}_j)$ gde je a unutrašnji mehanizam pažnje (videti odeljak 4.6.4). Agregacija biranjem maksimuma se definiše izborom $c_{j'i} = 1$ za $j' = \arg \max_{j \in \mathcal{N}_i} \mathbf{W} \vec{h}_j$, dok je $c_{ji} = 0$ za ostale vrednosti $j \in \mathcal{N}_i, j \neq j'$.

Predložena arhitektura je prikazana na slici 5.1. Mreža obrađuje proteinsku sekvencu transformisanjem u reprezentaciju od 16 atributa, koja se prosleđuje potpuno konvolutivnoj mreži f . Nakon što f konstruiše reprezentacije dimenzije k za svaki čvor, grafovska neuronska mreža g vrši dalju obradu čiji je izlaz ocena verovatnoća za svaki čvor da li pripada skupu funkcija datog proteina.

Spektralni atributi

Još jedna ideja implementirana u sklopu grafovskih obrada je uključivanje matrice Laplasove matrice grafa (videti odeljak 2.3), odnosno njenih sopstvenih vektora [46]. *Spektralni atributi* se formiraju tako što svaki izabrani sopstveni vektor Laplasove matrice obezbeđuje po jedan dodatni atribut za svaki čvor. Sopstveni vektori koji odgovaraju većim sopstvenim vrednostima ukazuju na pravac delovanja Laplasove matrice grafa, što ih čini boljim za izbor u slučaju formiranja spektralnih atributa. Korišćenje svih sopstvenih vektora za dobijanje spektralnih atributa zahteva dodatne vremenske resurse i nepovoljno utiče na generalizaciju zbog preprilagođavanja grafovskoj strukturi. Zato je preporučljivo uzeti fiksiran broj sopstvenih vektora (u oznaci n_{spec}) koji će proizvesti tačno n_{spec} spektralnih atributa za svaki čvor. Broj spektralnih atributa n_{spec} je hiperparametar modela sa kojim se može eksperimentisati.

Tokom obučavanja modela, konvolutivna mreža f obezbeđuje reprezentaciju dimenzije k na koju se nadovezuje n_{spec} fiksnih spektralnih atributa. Dodatni fiksirani atributi izvedeni iz grafovske strukture mogu pomoći neuronskoj mreži u poboljšavanju prediktivnih performansi.

Izlaz iz mreže

Poslednji sloj mreže je potpuno povezani sloj sa sigmoidnom aktivacionom funkcijom, odnosno skup 123 neurona koji na osnovu prethodnog sloja grafovske neuronske mreže za svaki čvor $i \in \mathbb{L}$ daju verovatnoću pripadnosti skupu proteinskih funkcija za dati protein. Zatim se izlazi porede sa pravim vrednostima koristeći grešku binarne unakrsne entropije. Cela neuronska mreža se obučava pomenutim tehnikama optimizacije (videti odeljak 4.4.2).

5.3 Biblioteke korišćene u imlementaciji

Implementacija je napisana u programskom jeziku *Python*. Korišćena je javno dostupna platforma *Google Colaboratory* koja predstavlja platformu kompanije *Google* namenjenu korisnicima *Python*-a koji žele da koriste moćne hardverske resurse koje je *Google* učinio javno dostupnim. Jedna od velikih pogodnosti ovog sistema je mogućnost korišćenja grafičkih karti (*GPU*) za obimnija izračunavanja, što značajno može ubrzati proces obučavanja, što je vremenski najzahtevniji deo posla.



Slika 5.2: **Levo:** logo biblioteke *biopython* [47], **desno:** logo biblioteke *PyTorch* [48].

Biopython

Biopython (slika 5.2 levo) predstavlja biblioteku programskog jezika *Python*, namenjenu za obrađivanje bioloških podataka. Iskorišćena je za očitavanje sekvenci iz datoteke u *fasta* formatu uz pomoć identifikatora proteina.

PyTorch

PyTorch (slika 5.2 desno) je moderna biblioteka mašinskog učenja bazirana na *Torch* biblioteci opremljenoj alatima za tenzorska izračunavanja. Razvijena je od strane kompanije *Facebook* i veoma je popularna. Neki delovi su pisani u programskom jeziku C++ radi efikasnosti. *PyTorch* obezbeđuje dva veoma značana resursa za rad sa dubokim neuronskim mrežama:

- Korišćenje tenzora sa moćnim ubrzanjem pomoću grafičkih procesorskih jedinica (GPU)
- Automatizovan sistem računanja gradijanata i propagacije unazad.

SciKit-Learn

Biblioteka *SciKit-Learn* (slika 5.3 levo) je moderna biblioteka mašinskog učenja u programskom jeziku *Python* koja sadrži implementirane različite modele za klasifikaciju, regresiju, klasterovanje kao i implementaciju obrada podataka koje su potrebne u mašinskom učenju. U



Slika 5.3: **Levo:** logo biblioteke *SciKit-Learn* [49], **desno:** logo biblioteke *NumPy* [50].

ovom radu je iskorišćena za nasumičnu podelu skupa za obučavanje, validaciju i testiranje. Dizajnirana je tako da se lako povezuje sa ostalim bibliotekama.

NumPy

Biblioteka *NumPy* (slika 5.3 desno) u programskom jeziku *Python* podržava izračunavanja sa tenzorima. Njena funkcionalnost se ne vezuje striktno za mašinsko učenje, ali je u njemu veoma korišćena. Pomenuta biblioteka *Torch* jako dobro sarađuje sa bibliotekom *NumPy* (na primer, tenzori definisani u *Torch* biblioteci se prenose u *NumPy* tenzor samo čuvanjem pokazivača na memorijski prostor).

U ovom radu je korišćena tako što su tenzori koji predstavljaju ulaz i izlaz modela, kao i matrica susedstva skladišteni u jednoj datoteci kao *NumPy* tenzori. Sve obrade su izvršene lokalno, nakon čega su podaci skladišteni na disku *Google Drive* sa kojeg se podaci vrlo brzo prebacuju na *Google Colab*, što je obezbedilo fokus na implementaciji neuronske mreže, njeno obučavanje i evaluaciju.

Poglavlje 6

Eksperimentalna evaluacija

U ovoj glavi je prvo opisan skup podataka koji se koristi prilikom obučavanja, validacije i testiranja modela. Zatim je navedena statistika podataka i način na koji se obrađuju. Nakon toga je opisana specifikacija procesa obučavanja modela i na kraju eksperimentalna evaluacija sa rezultatima prikazanim u tabeli.

6.1 Skup podataka

Ceo skup podataka koji se koristi u ovom radu je preuzet sa takmičenja *CAFA3* [2]. Podaci su organizovani u dve datoteke:

- *uniprot_sprot_exp.txt* - svaki red sadrži jedinstveni identifikator proteina, funkciju proteina i oznaku ontologije. Proteinske funkcije koje se javljaju u ovoj datoteci predstavljaju listove u ontologiji, a preostali skup funkcija proteina se dobija tako što se kreira minimalni konzistentni podgraf u odgovarajućoj ontologiji koji sadrži date listove. Razmatrani su redovi datoteke koji su označeni slovom *F*, koje predstavlja obeležje ontologije molekularnih funkcija (videti odeljak 3.2.1). U datoteci se pojavljuje ukupno 34920 različitih proteina.
- *uniprot_sprot_exp.fasta* - za identifikatore proteina su obezbeđene proteinske sekvence u formi niske od 20 simbola za standardne aminokiseline i ponekih simbola nestandardnih aminokiselina.

6.2 Statistika podataka i pretprocesiranje

Ontologija molekularnih funkcija (*MFO*) nije konačno utvrđena jer naučnici pronalaze nove nazive koje ubacuju kao čvorove, ali isto tako i odstranjuju neke postojeće. Čvorovi koji se eliminišu iz ontologije se označavaju kao *zastareli* (engl. *obsolete nodes*). Prema sajtu *Gene Ontology* [5] trenutno postoji 11118 čvorova u *MFO* koji predstavljaju postojeće funkcije proteina za ovu ontologiju. Podaci su dati u datoteci kao parovi:

identifikator proteina $\rightarrow$ list u ontologiji

Prva obrada podrazumeva da se objedine svi listovi koji pripadaju istom proteinu. Zatim se tom proteinu pridruže i preostale funkcije tako što se na osnovu listova formira minimalni konzistentni podgraf. Implementacija nalaženja minimalnog konzistentnog podgraфа koji sadrži date listove se vrši pomoću raščlanjivanja podataka o čvorovima korišćenjem modula *obo_parser* iz biblioteke *goatools*. Tako se formira usmeren aciklički graf *go* koji sadrži podatke o čvorovima i metode koje se nad njima mogu primenjivati. Za određivanje svih predaka za dati čvor postoji ugrađena funkcija *get_all_parents()*. Minimalni konzistentni podgraf koji predstavlja skup funkcija posmatranog proteina se može dobiti unijom listova i svih predaka listova.

Zbog ograničenih resursa, svi čvorovi koji se javljaju kao proteinske funkcije u manje od 500 proteina su odstranjeni. Na taj način je ontologija *MFO* redukovana na 123 čvora. Svi čvorovi koji su izbačeni, eliminisani su i iz skupova funkcija za bilo koji protein. Može se primetiti da pomenuta transformacija neće narušiti ontologiju u toj meri da se ne može formirati konzistentni podgraf zbog toga što je neki predak čvora isključen. Naime, svaki roditelj nekog čvora mora biti u skupu funkcija za više proteina nego sam taj čvor, jer ako je čvor u skupu funkcija datog proteina, moraju biti i njegovi roditelji. Time redukovana ontologija ostaje usmeren aciklički graf. Broj grana u redukovanoj ontologiji je 145. Nakon uvođenja redukovane ontologije skup funkcija za neke proteine se sveo na koreni čvor, pa je prirodno izbaciti sve takve proteine. Eliminirani su i proteini čija je sekvenca duža od 1000 aminokiselina. Broj takvih proteina je 3677. Razlog eliminisanja sekvenci dužih od 1000 je ušteda vremena koje je potrebno konvolutivnim slojevima da ekstrahuju atribut. Sve sekvence su proširene do dužine 1000 proizvoljnim simbolom, a kreirani su i *vektori maski* koji sadrže jedinice na svim pozicijama gde zaista postoje aminokiseline, a na preostalim mestima nule. *Vektori maski* služe kako bi se za sekvencu koja je proširena uzimale u obzir samo aminokiseline koje ona zaista sadrži. Proširivanje se vrši zbog konvolucija koje se lakše definišu nad ulazima fiksirane dužine. Nakon pretprocesiranja broj proteina koji je preostao u skupu podataka je 31243. Prosečan broj aminokiselina u sekvencama iznosi 431, dok je prosečan broj funkcija po proteinu jednak 7.

Ulazi su pripremljeni korišćenjem biblioteke *Biopython*, a zatim transformisani u tenzore koji se obrađuju uz pomoć biblioteke *Pytorch*. Izlazi su vektori koji su takođe specijalna vrsta tenzora. Pripremljeni podaci su sačuvani u obliku tenzora koji su prevedeni u *NumPy* strukture koje se mogu sačuvati u datoteci zahvaljujući pogodnostima ove biblioteke.

6.3 Specifikacije procesa obučavanja modela

Ceo skup podataka je podeljen na skup za obučavanje, validaciju i testiranje u procentualnom odnosu 68:17:15. Podela je vršena nasumično pomoću metode iz *SciKit-Learn* biblioteke. Brojanjem pojavljivanja svake od proteinskih funkcija u sva tri skupa provereno je da su skupovi dobro stratifikovani, odnosno raspodela broja proteina koji sadrži određenu funkciju po skupovima za obučavanje, validaciju i testiranje prati ugrubo proporciju 68:17:15 uz poneka variranja.

Mera kvaliteta modela koja je korišćena je F_1 skor, što je prirodan izbor kod problema sa neizbalansiranim klasama, a takođe se koristi i na *CAFA* takmičenjima.

Hiperparametri arhitekture su određeni na osnovu kriterijuma *ranog zaustavljanja* razmatrajući F_1 skor na validacijskom skupu. Nakon eksperimentisanja sa brojem konvolutivnih slojeva, primećeno je da *osnovna mreža* (opisana u odeljku 5.2.2) radi najbolje sa 6 dilatacijskih

konvolutivnih slojeva. Pojedinačne aminokiseline iz proteinske sekvence se inicijalno pretvaraju u *one-hot* reprezentaciju, nakon čega sledi potpuno povezani sloj od 16 neurona. Nakon toga slede dilatacijski konvolutivni slojevi koji imaju eksponencijalno brz rast receptivnog polja što omogućava korišćenje manjeg broja parametara za pokrivanje sekvence što ostavlja prostora za eksponencijalan rast izlaznih kanala. Korišćeni broj ovih kanala je: 16, 32, 64, 128, 256, 512 i 512. Više konvolutivnih slojeva od 6 prouzrokuje previše parametara što vodi ka problemu preprilagođavanja kao i značajnom trošenju vremenskih resursa. Korišćena je *regularizacija izostavljanjem* sa verovatnoćom 0.3. Početne vrednosti parametara su odabrane *normalizovanom inicijalizacijom* (videti odeljak 4.4.1). Između konvolutivnih slojeva je korišćena *ELU* aktivaciona funkcija.

Grafovska mreža (iz odeljka 5.2.2) podrazumeva konstruisanje reprezentacija čvorova koje se koriste kao temelj za grafovsku neuronsku mrežu. Nakon eksperimentisanja vrednosti za dužinu reprezentacije (od 2 do 10) fiksirana je vrednost $k = 9$. Broj spektralnih atributa je fiksiran $n_{spec} = 5$ i njihovo uključivanje je donelo poboljšanje F_1 skora.

Broj instanci koji se koristi za računanje gradijenta prilikom jednog koraka optimizacije je fiksiran na 32. Korišćen je metod optimizacije *Adam* sa stopom učenja 0.001. Model *osnovna mreža* proizvodi vektor dužine 123 sa verovatnoćama koje se porede sa istinitim vrednostima pomoću *binarne unakrsne entropije*.

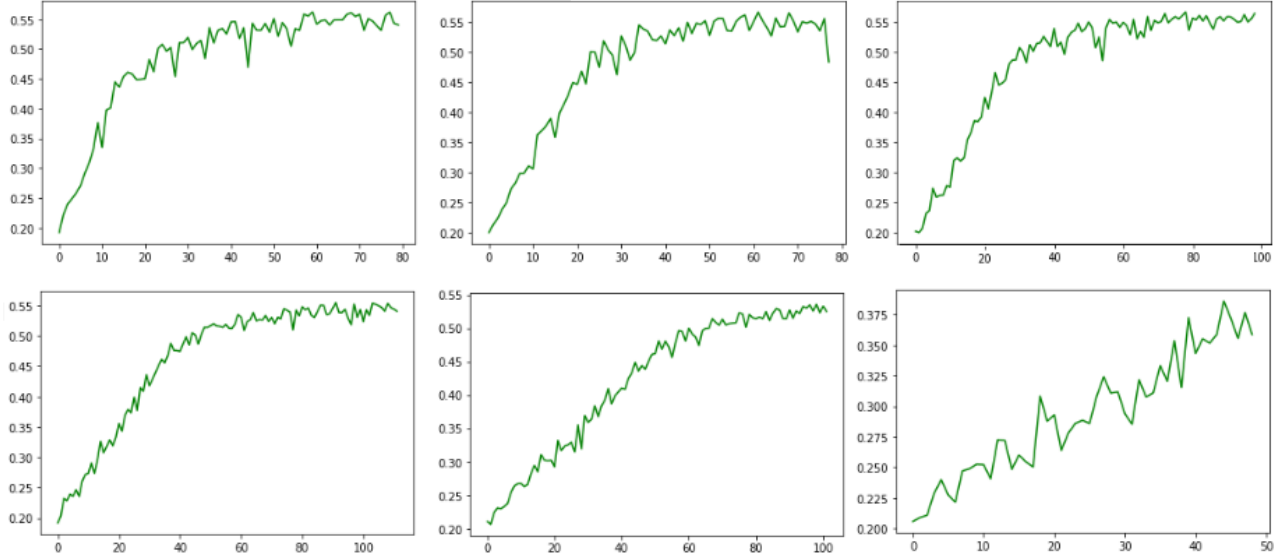
U implementaciji, načini agregacije stanja u susedstvu čvora, uključivanje spektralnih atributa i matrice povezanosti kontrolisani su indikatorima. Model *osnovna mreža* isključuje matricu susedstva u potpunosti, zatim se dobijaju različiti modeli odabirom agregacije i poslednja nadogradnja je inkorporacija spektralnih atributa. Tako su izvršena poređenja performansi različitih modela na skupu za validaciju i testiranje u odnosu na vrednosti F_1 skora.

6.4 Eksperimenti za određivanje hiperparametara

Hiperparametri se određuju jedan po jedan tako što se ispituju nekoliko vrednosti razmatranog hiperparametra dok je preostala arhitektura fiksirana. Korišćen je zbog jednostavnosti jer su eksperimenti vremenski dugi, ali postoje i bolji pristupi. Zatim se na osnovu F_1 skora na validacionom skupu bira vrednost koja se koristila u arhitekturi sa najboljim rezultatom. Nakon toga se razmatra sledeći hiperparametar istim postupkom i tako dalje. Razlog zbog čega se ne vrše eksperimenti za sve moguće konfiguracije je vremenska zahtevnost. Utvrđivanje nekog hiperparametra ne garantuje njegovu optimalnost. Prvi cilj je bio naći što bolji model *osnovne mreže*, ali je glavni cilj bio pokazati da *grafovska mreža* daje poboljšanje u odnosu na osnovni model.

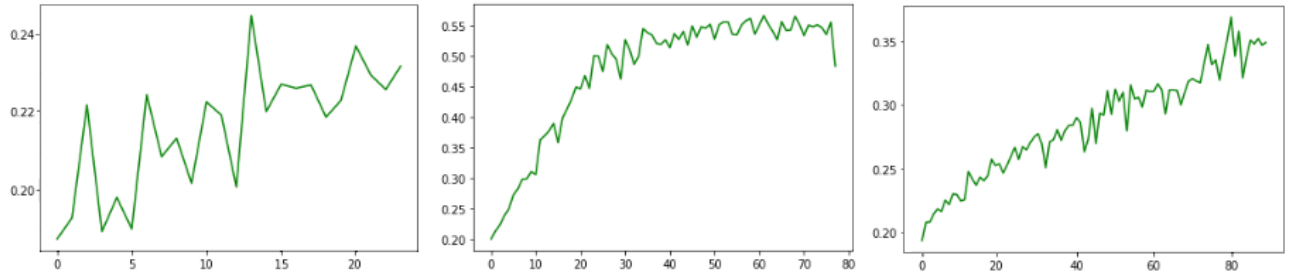
Prvi hiperparametar koji je razmatran je broj instanci koji se koristi za računanje stohastičke aproksimacije gradijenta prilikom jednog koraka optimizacije tkz. *batch size* (u oznaci bs). Grafički prikaz promene F_1 skora na validacionom skupu tokom epohe za različite vrednosti $bs \in \{16, 32, 64, 128, 256, 512\}$ je dat slikom 6.1. Najveći F_1 skor je dostignut za $bs = 32$ čime je fiksirana vrednost ovog hiperparametra.

Sledeći eksperimenti su pokretani za ispitivanje stope učenja (u oznaci lr). Ispitane su vrednosti 10^{-n} gde je $n \in \{2, 3, 4\}$. Grafici zavisnosti F_1 skora na validacionom skupu od rednog broja epohe dati su na slici 6.2. Primećeno je da za $lr = 0.01$ stopa učenja previše velika. Optimizacioni algoritam naizmenično naglo povećava i smanjuje vrednost F_1 skora na



Slika 6.1: Grafici predstavljaju prikaz F_1 skora na validacionom skupu u zavisnosti od rednog broja epohe. U prvom redu su grafici za vrednosti $bs \in \{16, 32, 64\}$, dok su u drugom redu grafici za vrednosti $bs \in \{128, 256, 512\}$.

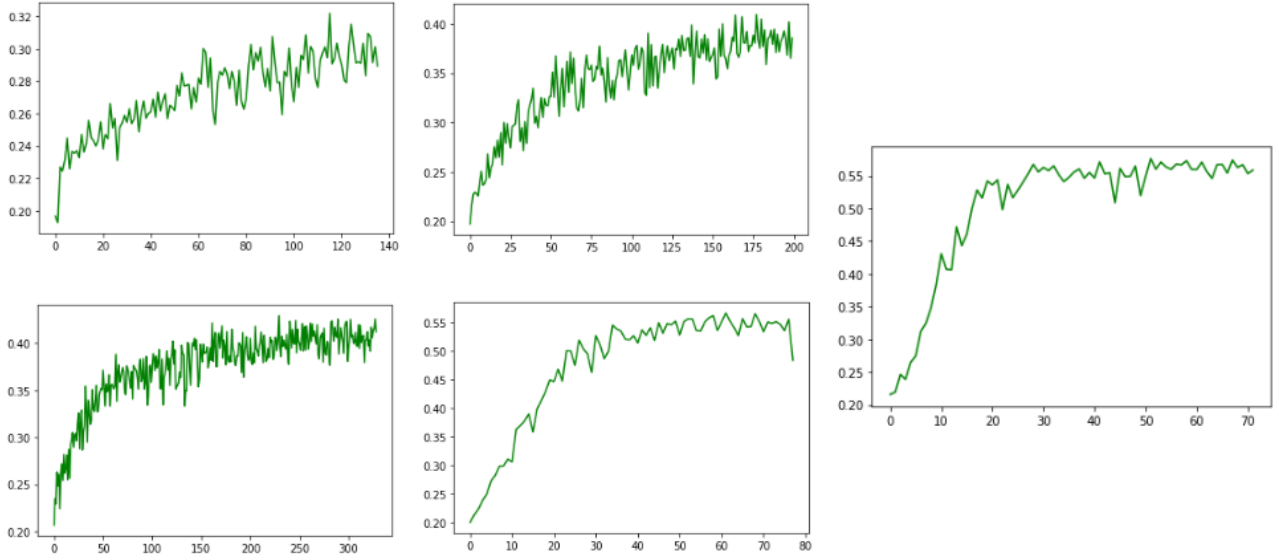
validacionom skupu. Iako je prisutan opšti trend rasta F_1 skora, ovakva nestabilnost tipično nije dobra jer optimizacioni proces ne može precizno prići lokalnom minimumu. Za vrednost $lr = 0.0001$ se može primetiti malo stabilniji rast, međutim F_1 skor se povećava jako sporo (primetiti i opseg vrednosti na y -osi). Na grafiku nisu prikazane sve epohe jer je proces prekinut zbog usporenog obučavanja. Naime, neki eksperimenti već na početku ukazuju da će proces učenja najverovatnije teći sporo ili nestabilno, zbog toga ih je moguće prekinuti zarad uštede vremena. Odabrana je vrednost $lr = 0.001$.



Slika 6.2: Grafici promenu F_1 skora na validacionom skupu tokom epohe za različite vrednosti stope učenja $lr \in \{0.01, 0.001, 0.0001\}$.

Sledeći hiperparametar koji je razmatran je broj konvolutivnih slojeva. Prethodni eksperimenti su pokretani koristeći arhitekturu *osnovne mreže* koja ima 5 konvolutivnih slojeva. Na slici 6.3 su prikazani grafici zavisnosti F_1 skora na validacionom skupu od rednog broja epohe za različiti broj konvolutivnih slojeva (od 2 do 6). Kako je 7 slojeva puno usporilo proces učenja, odlučeno je da arhitektura ima 6 konvolutivnih dilatacijskih slojeva i ona je pokazala najbolje rezultate. Za arhitekture koje imaju broj slojeva manjih od 6 isproban je i veći broj neurona u

poslednjem sloju kako bi se proširio skup parametara. Utvrđeno je da te arhitekture ne mogu da dosegnu vrednosti F_1 skora na validacionom skupu koje se postižu prilikom korišćenja 6 konvolutivnih slojeva.



Slika 6.3: Grafici predstavljaju prikaz F_1 skora na validacionom skupu u zavisnosti od rednog broja epohe. Skroz desno se nalazi grafik za 6 konvolutivnih slojeva. Preostala četiri grafika su raspoređena tako da su u prva dva reda broj konvolutivnih slojeva redom 2 i 3, dok su u drugom redu 4 i 5. Važno je uočiti i opseg vrednosti na y -osi.

Za ostale hiperparametre koji su uključeni u *grafovskoj mreži* nije prikazan grafik promene F_1 skora na validacionom skupu tokom epohe jer se uglavnom birala arhitektura koja dostigne njegovu najveću vrednost, a nije bilo potrebe za analiziranjem grafika.

6.5 Evaluacija i rezultati

Prilikom obučavanja, na kraju svake epohe se pamte modeli čija ocena na validacionom skupu premašuje do tog trenutka najbolji F_1 skor. Ukoliko se F_1 skor na validacionom skupu ne poboljšava tokom uzastopnih 20 epoha, proces obučavanja se zaustavlja, a za najbolji model se uzima poslednji sa najvećim F_1 skorom. Globalno ograničenje je 200 epoha.

Eksperimenti koji su korišćeni za podešavanje hiperparametara su pokretani po jedanput. Nakon što su hiperparametri fiksirani, različite arhitekture koje podrazumevaju *osnovnu mrežu*, *grafovsku mrežu* sa različitim tipovima agregacije, kao i *grafovsku mrežu* sa spektralnim atributima su evaluirane 5 puta korišćenjem procesa nasumičnosti prilikom inicijalizacije. Na osnovu tih 5 eksperimenata se računaju proseki i standardna devijacija.

Evaluacija obučenog modela se vrši na test skupu. Podaci iz test skupa nisu uticali ni na koji način na proces obučavanja i upotrebljeni su tek na samom kraju. Rezultati evaluacije različitih arhitektura su prikazani u tabeli 6.1.

Tabela 6.1: Vrednosti F_1 skora na validacionom i test skupu za sve razmatrane arhitekture, sa izračunatim prosekom i standardnom devijacijom za 5 eksperimenata.

Model	Validacioni F_1	Test F_1
Osnovna mreža	0.582 ± 0.003	0.584 ± 0.003
Grafovska mreža - uprosečavanje	0.583 ± 0.006	0.586 ± 0.004
Grafovska mreža - GAT	0.582 ± 0.004	0.587 ± 0.005
Grafovska mreža - biranje maksimuma	0.581 ± 0.002	0.585 ± 0.004
Grafovska mreža - sumiranje	0.596 ± 0.003	0.600 ± 0.003
Grafovska mreža - sumiranje (bez spektralnih atributa)	0.587 ± 0.007	0.590 ± 0.008

Tumačenje rezultata

Na osnovu rezultata u tabeli 6.1, može se zaključiti da model dobro generalizuje jer je opseg F_1 skora sličan između validacionog i test skupa. Arhitekture koje uključuju grafovsku mrežu su postigle bolje rezultate u odnosu na osnovni model. Najbolji model za izabranu meru kvaliteta je grafovska mreža sa agregacijom sumiranja. Arhitektura najboljeg modela postiže dobre rezultate i zahvaljujući uključivanju spektralnih atributa, kao što se može videti u tabeli. Na takmičenju *CAFA3*, F_1 skor koji su postigli najboljih 10 metoda se nalazio u opsegu $0.50 - 0.60$. Naravno, direktno poređenje sa ovim rezultatima nije moguće jer je korišćena redukovana ontologija. Ipak, ova opservacija je korisna jer pomenuti opseg vrednosti ukazuje da iako vrednost F_1 skora od 0.6 ne deluje velika, nalazi se u okviru razumnog raspona u odnosu na ovaj problem.

Poglavlje 7

Zaključak i dalji rad

Razvoj grafovskih neuronskih mreža je uzeo maha u poslednjih nekoliko godina, šireći spektar metoda mašinskog učenja i problema na kojima se one mogu primeniti. S druge strane, bioinformatika je takođe veoma aktivna naučna oblast koja doprinosi automatizovanom izučavanju biologije i biomedicine.

Grafovska struktura u problemu predviđanja funkcija proteina se pojavljuje u izlazu mreže, što do sada nije rešavano grafovskim neuronskim mrežama prema saznanjima autora. Pokazano je da osnovni model koji sadrži samo konvolutivne slojeve i zanemaruje grafovsku strukturu među funkcijama proteina, ostvaruje slabije rezultate u poređenju sa arhitekturama koje koriste grafovsku neuronsku mrežu kao nagradnju osnovnog modela. Među tipovima agregacije uprosečavanjem, pozornim mrežama, biranjem maksimuma i sumiranjem, najbolji rezultat postiže arhitektura koja stanja iz susedstva agregira sumiranjem. Spektralni atributi dobijeni iz Laplasijana grafa takođe doprinose boljim performansama predviđanja modela.

Postoji više pravaca za dalji rad i istraživanje. Osim proteinske sekvence kao ulaznog podatka, vrlo korisni atributi mogu biti i fizičko-hemijska svojstva proteina, sekundarna, tercijarna i kvatenarna struktura, interakcije sa ostalim proteinima i ostali relevantni dopunski atributi. Skup podataka može biti obogaćen sa novim proteinima i njihovim svojstvima. Redukovanu ontologija *MFO* se može proširiti na ceo skup čvorova, što otvara mogućnost direktnog poređenja sa ostalim metodama. Osim ontologije molekularnih funkcija, model se može kreirati i za preostale dve ontologije (*BPO* i *CCO*).

Među grafovskim neuronskim mrežama postoji još metoda koje se mogu upotrebiti za rešavanje problema funkcije proteina. Uz dodatne podatke, attribute i ublažavanje problema ograničenih resursa, postoji potencijal za još efikasnije rešavanje ovog problema.

Literatura

- [1] https://www.biofunctionprediction.org/cafa-targets/Introduction_to_protein_prediction.pdf.
- [2] *The CAFA Challenge*. <https://www.biofunctionprediction.org/cafa/>.
- [3] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
- [4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- [5] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.
- [6] Naihui Zhou, Yuxiang Jiang, Timothy R Bergquist, Alexandra J Lee, Balint Z Kacsoh, Alex W Crocker, Kimberley A Lewis, George Georghiou, Huy N Nguyen, Md Nafiz Hamid, et al. The cafa challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome biology*, 20(1):1–23, 2019.
- [7] Stefan Spalević, Petar Veličković, Jovana Kovačević, and Mladen Nikolić. Hierachial protein function prediction with tails-gnns. *ICML 2020 - Workshops - Graph RepresGraph Representation Learning and Beyond (GRL+)*, *arXiv preprint arXiv:2007.12804*, 2020.
- [8] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, NJ, 1996.
- [9] <https://medium.com/kriptapp/guide-what-is-directed-acyclic-graph-364c04662609>.
- [10] <https://github.com/HQarroum/directed-graph>.
- [11] <https://www.javatpoint.com/graph-representation>.
- [12] Cvetković Dragos and Slobodan Simić. Towards a spectral theory of graphs based on the signless laplacian, iii. *Applicable Analysis and Discrete Mathematics*, 4:156–166, 03 2010.

- [13] Albert L Lehninger, David L Nelson, Michael M Cox, et al. *Lehninger principles of biochemistry*. Macmillan, 2005.
- [14] Akif Uzman. Molecular biology of the cell (4th ed.): Alberts, b., johnson, a., lewis, j., raff, m., roberts, k., and walter, p. *Biochemistry and Molecular Biology Education*, 31:212 – 214, 07 2003.
- [15] <https://www.shutterstock.com/image-vector/levels-protein-structure-amino-acids-complex-1427667899>.
- [16] Dragoslav Marinković, Marko Anđelković, Ana Savić, and Vukosava Diklić. *Biologija za III razred medicinske i veterinarske škole*. Zavod za udžbenike i nastavna sredstva, 2016.
- [17] Gordana Supic, Katarina Zeljic, and Zvonko Magic. *HUMANA GENETIKA - Priručnik za praktičnu nastavu*. 10 2016.
- [18] <https://www.pinterest.com/pin/584201382893190538/>.
- [19] Anders Esberg. *Functional aspects of wobble uridine modifications in yeast tRNA*. PhD thesis, Molekylärbilogi (Teknisk-naturvetenskaplig fakultet), 2007.
- [20] <http://kiharalab.org/web/navigo/views/goset.php>.
- [21] <https://www.rcsb.org/3d-view/>.
- [22] <https://www.ebi.ac.uk/QuickGO/>.
- [23] Mladen Nikolić and Anđelka Zečević. *Mašinsko učenje*. 2019.
- [24] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [25] <https://cs231n.github.io/neural-networks-1/>.
- [26] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- [27] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [29] Zorica Stanimirović. *Nelinearno programiranje*. Univerzitet u Beogradu - Matematički fakultet, 2014.
- [30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

- [31] Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.
- [32] Petar Veličković. *The resurgence of structure in deep neural networks*. PhD thesis, University of Cambridge, 2019.
- [33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [34] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- [35] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [36] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [37] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.
- [38] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.
- [39] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 385–394, 2017.
- [40] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.
- [41] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48, 2016.
- [42] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [43] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.
- [44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

- [46] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- [47] <https://biopython.org/>.
- [48] <https://pytorch.org/>.
- [49] <https://scikit-learn.org/stable/>.
- [50] <https://numpy.org/>.
- [51] The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research*, 47(D1):D330–D338, 11 2018.