Kolektivna dinamika fizioloških signala

Barić, Domjan

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Domjan Barić

Kolektivna dinamika fizioloških signala

Diplomski rad

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA; SMJER ISTRAŽIVAČKI

Domjan Barić

Diplomski rad

Kolektivna dinamika fizioloških signala

Voditelj diplomskog rada: izv. prof. dr. sc. Davor Horvatić

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2018.

Zahvaljujem se mentoru, izv. prof. dr. sc. Davoru Horvatiću , na savjetima i vođenju tijekom studija, te ponajviše na slobodi prilikom odabira teme diplomskog rada i načina njezine realizacije.

Sažetak

U ovom radu cilj je analizirati smo podatke dobivene snimanjem aktivnosti neurona transgeničnog miša (ekspresija GCaMP3, genetski enkodiran kalcij indikator) tijekom trčanja po pokretnoj traci. Koristeći Isingov model za analizu dinamike spinskog lanca, želimo pronaći vjerojatnost svakog stanja i razumjeti strukturu mreže neurona. Prvi korak je riješiti inverzni problem te pronaći parametre Isingovog modela. Osim egzaktne optimizacije parametara Isingovog modela, primjenjujemo nekoliko aproksimativnih metoda koje uspoređujemo s egzaktnom optimizacijom. U drugom dijelu rada opisujemo kako su metode strojnog učenja prirodno produženje inverznih problema u fizici. Koristeći strojno učenje možemo definirati metriku točnosti. Možemo predvidjeti u kojem će se slijedećem stanju neuron naći s točnošću od 92%. Opisan je algoritam koji predviđa stanje cijele mreže neurona, a ne samo jednog neurona. Za mrežu koja se sastoji od 37 neurona, možemo predvidjeti slijedeće stanje mreže s greškom manjom od četvrtine neurona po primjeru. Pomoću metoda strojnog učenja klasificiramo svako pojedino izmjereno stanje neurona u stanje rješavanja zadatka i stanje nagrade. Osim što s točnošću većom od 95% možemo klasificirati stanja, možemo odrediti aktivnost kojih neurona signalizira da se radi o izvršavanju nagrade ili zadatka.

Ključne riječi: fiziološka mjerenja, neuroni, Isingov model, inverzni problem, strojno učenje, boosting, stablo odluke, ansambli modela, neuronske mreže, SHAP.

Collective dynamics of physiological signals

Abstract

In this thesis, we analyzed neural activity data obtained from transgenic mice (expression GCaMP3, genetically encoded calcium indicator) during tasks/reward sessions. Using Ising's model for analysis of spin chain dynamics, we want to find the probability of each state and understand the structure of the network. The first step is to solve the inverse problem and find the parameters of the Ising model. In addition to the exact optimization of the Ising model parameters, we apply several approximate methods that we compare with exact optimization. In the second part, we describe how machine learning methods are a natural extension of inverse problems in physics. Using machine learning, we are able to not only answer various questions, but we can define a metric of precision. To the question in which state will neuron be in the next moment, we are able to answer with the precision of 92 %. We also describe an algorithm that predicts the state of the entire neuronal network, not just one neuron. For a network that consists of 37 neurons, we can predict the network state in the next moment with an average error of less than a quarter of the neurons per instance. Using the machine learning methods, we classify each individual measured neuronal state into the state of solving the task, or the state of reward. Not only that we can classify states with accuracy greater than 95%, but we can also understand the activity of which neurons signal that mouse is solving the task or receiving the reward.

Keywords: physiological measurements, neurons, Ising's model, inverse problem, machine learning, boosting, decision tree, model ensembles, neural networks, SHAP.

Sadržaj

1 Uvod

Isingov model, nazvan po fizičaru Ernstu Isingu, matematički je model feromagnetizma u statističkoj fizici. Model se sastoji od diskretnih varijabli koje predstavljaju magnetske momente spinova koji mogu biti u jednom od dva stanja (+1 ili -1). Spinovi su raspoređeni po mreži, koja omogućuje svakom centru da komunicira sa svojim susjedima. Model omogućuje prepoznavanje faznih prijelaza. Dvodimenzionalni Isingov model s kvadratnom rešetkom jedan je od najjednostavnijih statističkih modela za prikaz faznih prijelaza [1]. Međutim, Isingov model se može primijeniti na bilo koji problem u kojem su centri, koji međusobno interagiraju, raspoređeni po mreži. Neke od primjena su [2]:

- **Modeliranje aktivnosti neurona i rekonstrukcija neuronskih veza** - Neuroni mogu biti aktivni (stanje +1) ili neaktivni (stanje 0). Neuroni su povezani u neuronsku mrežu (dio mozga ili živčanog sustava). Cilj je razumjeti kako su neuroni povezani, što uzrokuje paljenje i gašenje neurona te sveobuhvatnu dinamiku mreže.

- **Određivanje strukture proteina** - Cilj je odrediti strukturu proteina. Na svakoj lokaciji u proteinu se može nalaziti jedna od 20 aminokiselina ili se može nalaziti praznina, tj. spin može imati dvadeset i jedno stanje (spin ne mora uvijek imati 2 stanja). Stanja s najmanjom energijom su ona koja očekujemo.

- **Određivanje vjerojatnosti reprodukcije gena** - Cilj je odrediti kako mutacije utječu na vjerojatnost da će se gen reproducirati. Svaka aminokiselina može biti u dva stanja: mutiranom (+1) ili nemutiranom (-1).

- **Liječenje kombinacijom antibiotika** - Zbog mutacija bakterije postaju otpornije na antibiotike. Jedna od strategija kako bi se smanjila otpornost je korištenje više antibiotika odjednom ili u rotaciji. Antibiotici mogu biti u stanju 0 ili 1, tj. jesu li primjenjeni ili ne.

- **Financijska tržišta** - Svakoj kompaniji je dodijeljeno stanje +1 (dionice će rasti) ili -1 (dionice će padati).

Primarni cilj statističke fizike je objasniti makroskopska svojstva tvari iz mikroskopskih zakona koji reguliraju najmanje dijelove sustava, npr. molekule ili atome. U

Isingovom modelu, početna točka je model koji opisuje interakcije između osnovnih magneta (spinova), cilj je izvesti opservable, kao što su magnetizacija spina i korelacije između spinova.
U inverznom problemu polazna točka su opažanja (mjerenja) nekog sustava čiji su mikroskopski parametri nepoznati. U inverznom Isingovom problemu, interakcija između spinova nije poznata, ali ih želimo odrediti iz magnetizacije, korelacija ili drugih opservabli. Općenito, cilj je zaključiti parametre koji opisuju sustav (na primjer, njegov Hamiltonijan) iz postojećih podataka. U tu svrhu, veza između mikroskopskih zakona i promatranih podataka je obrnuta.
U posljednja dva desetljeća pojavili su se inverzni statistički problemi u različitim kontekstima, što je nagnalo fizičare se okrenu metodama za analizu promatranja. Dva su ključna razloga zbog kojih se javljaju inverzni problemi:

1. Mikroskopske skale postaju eksperimentalno mjerljive
2. pohrana velikih količina podataka je postala jeftinija i jednostavnija.

Konkretno, biološke znanosti su stvorile nekoliko inverznih statističkih problema, uključujući rekonstrukciju neuronskih i genskih regulatornih mreža i određivanje trodimenzionalne strukture proteina. U svim primjerima, ”spin” opisuje mikroskopske stupnjeve slobode sustava, npr. stanje neurona u neuronskoj mreži.

Godine 2009. kreiran je ImageNet [3] - baza slika koja sadrži više od 14 milijuna slika. Ona je dizajnirana u svrhu znanstvenih istraživanja na području vizulanog prepoznavanja objekta. Za mnoge, ImageNet je bio katalizator za boom umjetne inteligencije u dvadeset i prvom stoljeću. Algoritmi strojnog učenja (preciznije: neuronske mreže) pokazali su bolje rezultate od standardnih algoritama računalnog vida, te su u 2015. imali veću točnost od čovjeka.
Strojno učenje je grana umjetne inteligencije koja se bavi oblikovanjem algoritama koji svoju učinkovitost poboljšavaju na temelju empirijskih podataka. Riječ ”učenje” se odnosi na to da algoritmi iz samih podataka pronađu parametre modela koji za dane podatke imaju najveću preciznost. Fizičari već dugi niz godina koriste strojno učenje, ali ga tako ne nazivaju. Linearne, polinomijalne i ostale regresije su najjednostavniji algoritmi strojnog učenja. Postupak traženja parametara za linearnu regresiju koji minimiziraju udaljenost predikcije i mjerenja na osi y je identičan optimizaciji

moderih algoritama strojnog učenja. Svaki algoritam (problem) strojnog učenja se sastoji od 4 ključna dijela:

1. Ulazni podatci **X** - Najčešće tablica u kojoj su stupci značajke, a redci primjeri (mjerenja). Za nestruktuirane podatke, npr. kategorizaciju slika to je sama slika.

2. Oznake **y** - Ono što želimo predvidjeti, npr. za slike ono što se nalazi na slici: pas, mačka, ptica itd.

3. Model **F** - Model koji služi za predikciju. Tražimo parametre *A* modela F takve da **F(X)**= $\hat{y}$ bude što sličnija izmjerenim vrijednostima.

4. Funkcija greške **L** - Funkcija koju minimiziramo, tj. cilj postupka je pronaći parametre *A* modela F takve da je funkcija greške, koju možemo promatrati kao udaljenost između predikcije i stvarne oznake, najmanja moguća: $\min_A L(F(X) = \hat{y}, y)$.

Klasični primjer za fiziku bi bio određivanje otpora vodiča: stupci ulaznih podataka **X** su struja i napon, a retci su mjerenja. Izmjerene vrijednosti otpora su **y**, model koji optimiziramo je

$$F(I, U) = a_1 \frac{U}{I} + a_0, \tag{1.1}$$

a funkcija greške je **L**($\hat{y}$,**y**)=$\sqrt{\frac{1}{N}\sum_i^N (\hat{y}_i - y_i)^2}$. Iako znamo da bi vrijednosti parametara a_1 i a_0 trebale biti 1 i 0, moramo provjeriti iz mjerenja da su one stvarno takve. Strojno učenje ubrzano raste u zadnjih nekoliko godina, te njegova primjena postaje sve češća:

- **Netflix** - Netflix koristi strojno učenje kako bi vam predložio sljedeći film ili seriju za gledanje. Također, koriste strojno učenje kako bi odabrali sliku koja će vas nagnati da nastavite gledati [4].

- **Apple: Siri** - Apple-ov pametni pomoćnik koristi strojno učenje kako bi iz vašeg glasa odredio pitanje/naredbu. Njegova preciznost raste što se više koristi, jer se algoritam uči i prilagođava vama [5].

- **PayPal** - PayPal koristi strojno učenje kako bi odredio koje su transakcije zapravo prevare [6].

Strojno učenje je optimalan pristup problemima u kojima imamo podatke ali zakonitost ponašanja nije poznata, što ih čini idealnim alatom za riješavanje inverznih problema u fizici. Primjer vrijedan spomena, u kojem su fizičari koristili strojno učenje, je CERN-ovo natjecanje objavljeno na web-stranici Kaggle pod naslovom: "Higgs Boson Machine Learning Challenge" [7]. Cilj natjecanja je bio identificirati $\tau - \tau$ raspad Higgsovog bozona. Na natjecanju je sudjelovalo više od 1700 sudionika. Ovo natjecanje je jedan od prvih trenutaka u kojem su fizičari primijenili sve alate strojnog učenja kako bi razumjeli fizikalne zakone iz svojih mjerenja.

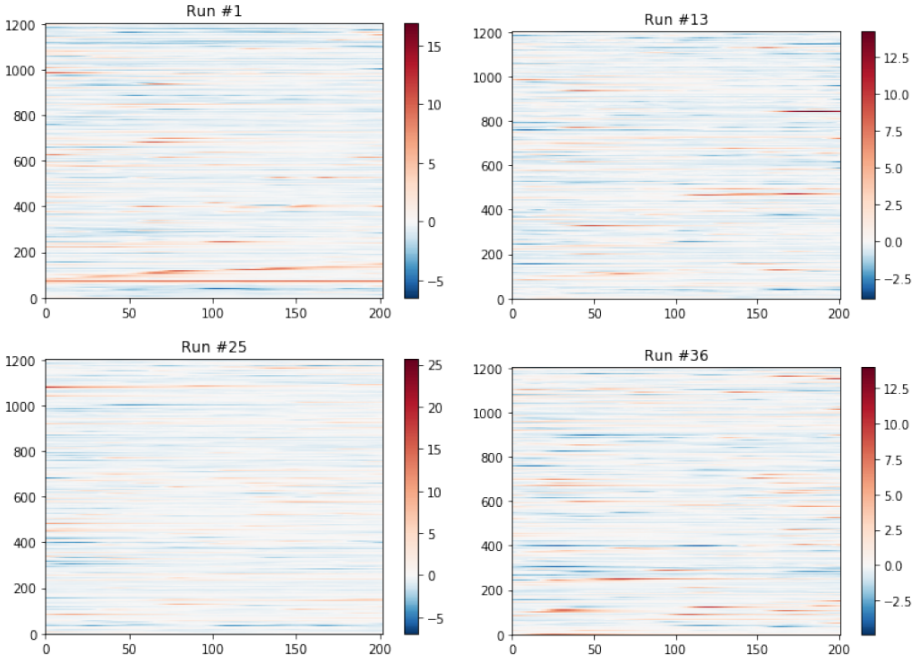
Danas sve popularnije neuronske mreže pripadaju algoritmima dubokog učenja što je grana strojnog učenja. Naziv "duboko" dolazi od toga što ovi algoritmi koriste nelinearne kombinacije ulaznih vrijednosti kako bi kreirali kompleksnije značajke koje su bolji prediktori od samih ulaznih vrijednosti. Strojno učenje se fokusira samo na treniranje algoritama, ono se ne bavi prikupljanjem, čišćenjem i vizualizacijom podataka. Svi ovi procesi su dio "Znanosti o podacima".

U ovom radu analiziramo signale s neurona miša dok rješava labirint. Nakon što riješi labirint dobije nagradu. Mjerene su koncentracije iona kalcija koje su povezane s aktivnosti neurona. Koncentracije kalcija su transformirane u vrijednosti 0 (neuron neaktivan) ili 1 (neuron aktivan). Postupak je opisan u poglavlju 2. Ovakvom definicijom uzorka upravo imamo Isingov problem. Ovo je zapravo inverzni Isingov problem zbog toga što imamo mjerenja i želimo saznati fizikalnu pozadinu aktivnosti neurona. U drugom poglavlju rada opisujemo podatke i proces pretvorbe iz koncentracije kalcija u aktivnost 0 ili 1. U trećem poglavlju analiziramo inverzni Isingov problem pomoću klasičnih fizikalnih metoda. U četvrtom poglavlju pristupamo problemu iz perspektive strojnog učenja.

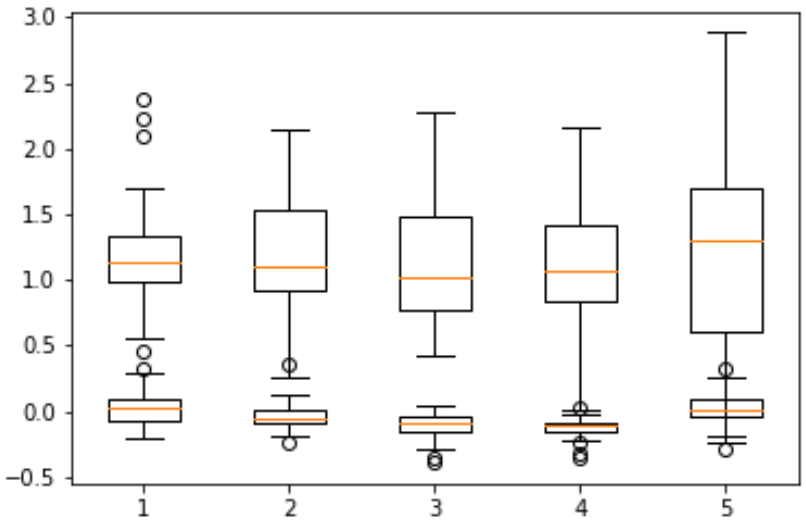
2 Opis i početna analiza podataka

U ovom radu analiziramo podatke dobivene snimanjem aktivnosti neurona transgeničnog miša (ekspresija GCaMP3, genetski enkodiran kalcij indikator) tijekom trčanja po pokretnoj traci. Duljina trajanja eksperimenta je 25 minuta i 42 sekunde. Frekvencija kojom snimamo stanje neurona je 20 Hz. Staza je kvadratnoga oblika s rasutim trakama za trčanje koje traju 10 sekundi. Nakon svake trake miš dobije nagradu. Očekujemo da će aktivnosti neurona na traci i van nje biti različite. Na stazi se nalazi 36 traka. Promatramo aktivnost 1202 neurona. Želimo modelirati aktivnost neurona, tako da iz stanja svih drugih neurona možemo odrediti u kojem stanju se i-ti neuron nalazi. Treba naglasiti da su miševi koje promatramo već upoznati sa stazom i procesom nagrađivanja. Zbog toga uzorci koje pronalazimo u neuronima su stvarno uzorci rješavanja staze, te uzorci očekivanja i dobivanja nagrade. Nepoznavanje staze i procesa nam ne unosi dodatni šum [8].

Na slici (2.1) vidimo aktivnost neurona za četiri različite trake. Bijela boja označava da neuron nije bio aktivan, crvena da je aktivnost pozitivna, a plava da je aktivnost negativna (negativna aktivnost znači da je koncentracija kalcija manja od uobičajene. Vidimo da je za većinu neurona aktivnost mala. Primjećujemo i da postoji naglašeno



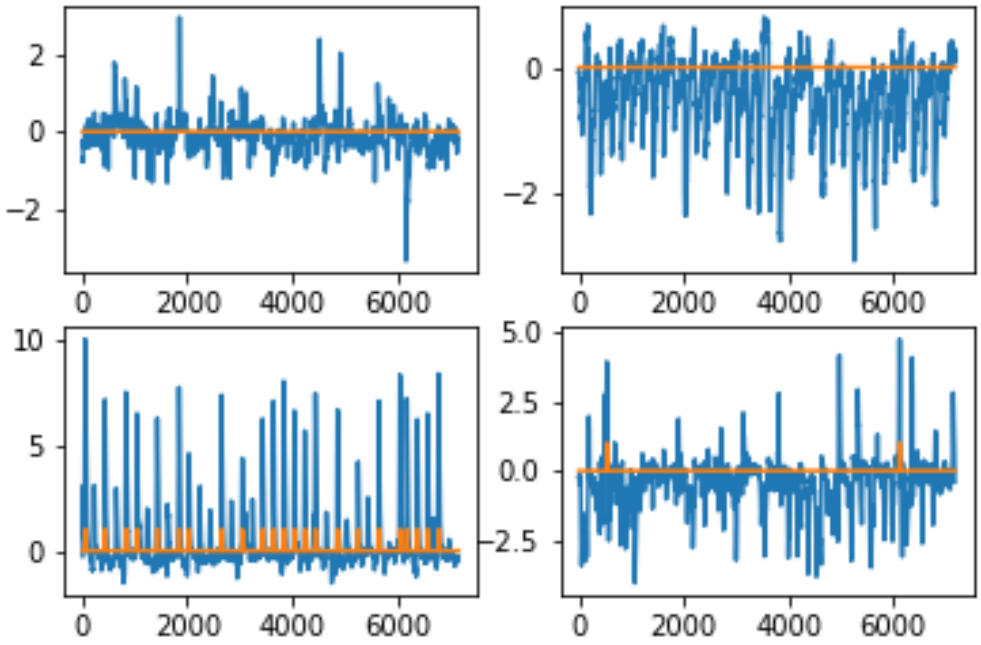
Slika 2.1: Snimka aktivnosti svih neurona za trake br. 1,13,25,36. Na y-osi je indeks neurona, a na x-osi trenutak snimanja. Boja opisuje aktivnost neurona. Izvor: obrada autora



Slika 2.2: Boxplot za 5 najaktivnijih neurona i 5 najmanje aktivnih neurona. Vidimo da aktivniji neuroni imaju više raspršene srednje vrijednosti aktivnosti po traci. Izvor: obrada autora

ponašanje u vremenu, tj. crvene crte su horizontalne. Neuron kad se aktivira, stoji aktivan neko vrijeme. Ne primjećujemo naglašene vertikalne crte, što znači da nema masivnoga simultanog aktiviranja neurona. Također, važno je napomenuti da indeksi neurona nemaju fizikalno značenje. Bilo koja permutacija neurona je jednako valjana. Za svaki neuron i za svaku traku možemo pronaći statističke vrijednosti poput: srednje vrijednosti, standardne devijacije, medijana, maksimalne vrijednosti i minimalne vrijednosti.

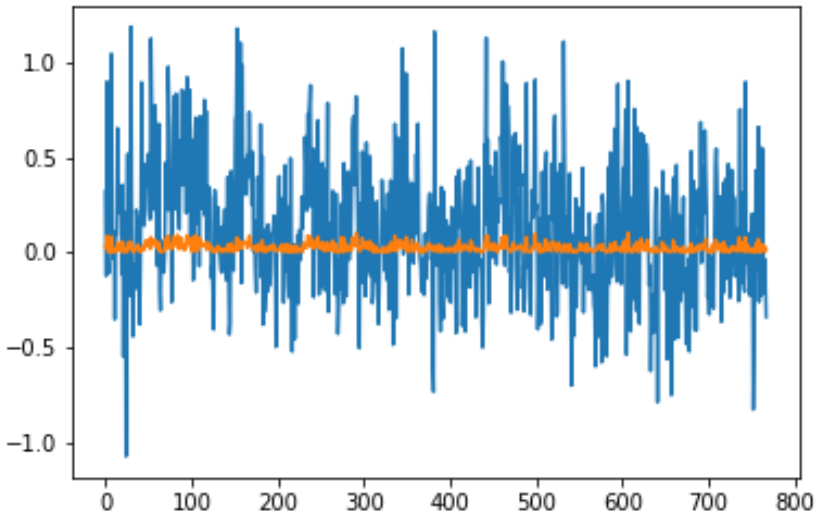
Na slici (2.2). je prikazan boxplot za 5 najaktivnijih i 5 najmanje aktivnih neurona u prosjeku. Primjećujemo da neuroni koji su slabo aktivni, slabo su aktivni za svaku traku. Neuroni koji imaju veće srednje vrijednosti aktivnosti, također imaju i veće standardne devijacije. Aktivnost neurona je nešto instantno, puls u trenutku koji traje kratko naspram vremena neaktivnosti. Želimo modelirati neurone poput spinova koristeći Isingov model. Stoga ćemo u svakom trenutku neurone koji su aktivni označiti s 1, a one koji su neaktivni s 0. Kažemo da je neuron aktivan u nekom trenutku t, ako je u tom trenutku njegova vrijednost veća od 3.5 standardnih devijacija, a neaktivan u suprotnom. Vrijednost standardne devijacije računamo po svakom trku na traci kao standardnu devijaciju aktivnosti svih neurona. Kada bi ju računali po neuronu, moglo bi se dogoditi da neuron koji je slabo aktivan, tj. oscilira oko nule, nakon binarizacije bude jako aktivan, jer mu je standardna devijacija jako mala. Stoga bi



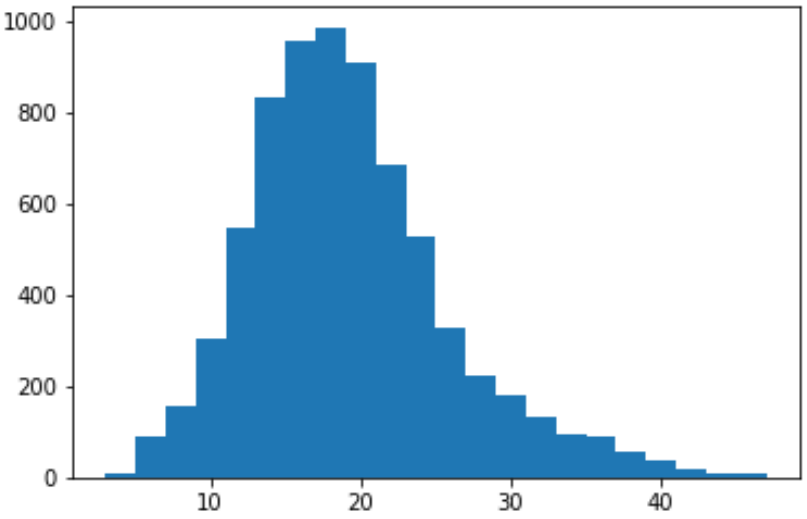
Slika 2.3: Na slici vidimo signal prije binarizacije (plavo) i poslije (narančasto) za neurone pod brojem 1,10,100,1000. Primjetimo različite skale na sva četiri grafa. Izvor: obrada autora

nam svako udaljavanje od nule izgledalo kao aktivnost, dok je u stvarnosti to zapravo samo šum.

Na slici (2.3). vidimo signal nakon binarizacije za neurone pod brojem 1,10,100 i 1000. Vidimo da tri od četiri neurona slabo aktivna. Važno je sjetiti se da za svaku traku binariziramo podatke posebno, što znači da ako je jedan signal jak na traci i, on može biti slab na traci j. Ovim želimo uhvatiti samo najvažnije događaje neurona. Kako bismo opravdali ovaj postupak pogledajmo na slici (2.4). srednje vrijednosti prije i poslije binariziranja. Očekivano je da će srednje vrijednosti nakon binariziranja podataka biti manje jer je maksimum za signal prije binariziranja 15 do 20 puta veći. Vidimo da binarizacija ne utječe drastično na srednju vrijednost što nam ukazuje da je ovaj postupak smislen. Ono što možemo promotriti je kolika je vjerojatnost da je K od N neurona aktivno, neovisno o tome koji su neuroni. Vjerojatnost je prikazana na slici (2.5) Vidimo da je očekivana vrijednost broja aktivnih neurona 19.



Slika 2.4: Slika prikazuje srednje vrijednosti prije i poslije binariziranja podataka. Plava boja označava početni signal, a narančasta vrijednosti za binarizirani signal. Izvor: obrada autora



Slika 2.5: Histogram broja aktivnih neurona. Za svaki trenutak je izbrojano koliko je neurona aktivno. Vidimo da se maksimum nalazi na 19 neurona. Izvor: obrada autora

3 Inverzni Isingov problem

Razmotrimo skup čvorova na mreži λ, svaki sa skupom susjednih čvorova koji tvori d-dimenzionalnu rešetku. Za svako mjesto rešetke $k \in \lambda$ postoji diskretna varijabla σ_k takva da $\sigma_k \in \{+1, -1\}$ što predstavlja spin čvora. Konfiguracija spina, $\boldsymbol{\sigma} = (\sigma_k)_{k\in\lambda}$ je dodjeljivanje vrijednosti spina svakom čvoru rešetke. Za dva susjedna mjesta $i, j \in \lambda$ postoji interakcija J_{ij}. Također za svaki čvor $i \in \lambda$ postoji vanjsko magnetsko polje h_i koje integrira s njim. Energija konfiguracije $\boldsymbol{\sigma}$ je dana Hamiltonijanom

$$H(\sigma) = -\sum_{\langle i,j\rangle} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i, \tag{3.1}$$

gdje je prva suma preko parova susjednih spinova (svaki par se broji samo jednom). Treba imati na umu da znak u drugom članu Hamiltonijana treba biti pozitivan jer je magnetski moment elektrona antiparalelan s njegovim spinom, ali negativni izraz se koristi konvencionalno. Oznaka $\langle i, j\rangle$ označava da su čvorovi i i j najbliži susjedi [9]. Ovako je prvotno definiran Isingov problem. Promatramo spinski lanac (ili mrežu). Svaki spin može biti u stanju gore (+1) ili dolje (-1). Isingov model primjenjen na dvodimenzionalnu kvadratnu mrežu je najjednostavniji statistički model koji ima fazni prijelaz.

U inverznom Isingovom problemu, kao i u svim inverznim problemima, polazimo od podataka (mjerenja). Priroda sustava nije poznata. Parametri modela su isti kao i u standardnom Isingovom problemu te je interakcija spinova definirana jednadžbom (3.1). Međutim parametri modela nisu poznati te ih iz različitih stanja spina želimo pronaći.

Mjerenja koncentracije kalcija u neuronima nakon binarizacije možemo promatrati kao lanac spinova koji se nalaze u stanju 0 (neaktivan neuron) ili stanju 1 (aktivan neuron). Međutim, važno je napomenuti da sada ne moramo promatrati samo spinove koji su najbliži susjedi, već je dozvoljena interakcija između bilo koja 2 spina. Ovako definiran graf je identičan inverznom Isingovom problemu, te možemo koristiti sve metode rješavanja inverznih Isingovih problema, ali i inverznih problema općenito.

Stohastička dinamika zasnovana na simetričnim vezama, podrazumijevaju ravnotežu sustava, tj. sustav se nalazi u stabilnom stanju koje je opisano Boltzmannovom dis-

tribucijom. Ako su veze asimetrične tada se sustav nalazi u neravnotežnom stanju mirovanja. Nemamo razloga zašto pretpostaviti da su veze u mreži neurona simetrične.

U poglavlju 3.1 tretiramo sustav kao da se nalazi u ravnoteži, tj. kao da su mu veze simetrične. U poglavlju 3.2 pretpostavljamo da se sustav ne mora nalaziti u ravnotežnom stanju.

3.1 Rekonstrukcija parametara u ravnoteži

Promatramo Isingov model s N binarnih spinova koji mogu poprimiti vrijednost 0 ili +1, $s_i = 0, 1, i = 1, ..., M$, gdje je M broj spinova. Vezanja parova (ili jačina vezanja parova) J_{ij} opisuje interakciju između parova spinova. Lokalna magnetska polja h_i djeluju na individualni spin. Energija spinske konfiguracije $\boldsymbol{s} \equiv \{s_i\}$ dana je s Hamiltonijanom:

$$H_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{s}) = -\sum_{i<j} J_{ij} s_i s_j - \sum_i h_i s_i. \tag{3.2}$$

Sustav Isingovog modela u ravnoteži opisan je Boltzmannovom distribucijom [10]

$$p(\boldsymbol{s}) = \frac{1}{Z} e^{-H_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{s})}, \tag{3.3}$$

gdje smo stavili da je $k_B T = 1$, tj. pretpostavljamo da je $\beta = 1$. Kada bi odabrali drugačiji T samo bi skalirali koeficijente J_{ij} i h_i, ali ponašanje sustava i vjerojatnosti pronalazaka bi ostale identične jer Boltzmannova distribucija ovisi samo o umnošcima βJ_{ij} i βh_i. Z označava particijsku funkciju:

$$Z(\boldsymbol{J}, \boldsymbol{h}) = \sum_{\boldsymbol{s}} e^{-H_{\boldsymbol{J},\boldsymbol{h}}(\boldsymbol{s})}. \tag{3.4}$$

U ovakvom statističkom opisu Isingovog modela, svaki je spin predstavljen slučajnom varijablom koju označujemo s σ, dok realizaciju spina označavamo sa s.

Ravnotežna magnetizacija spina je dana s $m_i \equiv \langle \sigma_i \rangle = \sum_{\boldsymbol{s}} p(\boldsymbol{s}) s_i$, a korelacija para s $\chi_{ij} \equiv \langle \sigma_i \sigma_j \rangle = \sum_{\boldsymbol{s}} p(\boldsymbol{s}) s_i s_j$. Također nas zanimaju i kovarijance parova koje su dane s: $C_{ij} \equiv \chi_{ij} - m_i m_j$.

Isingov sustav u ravnoteži je u potpunosti određen jačinom vezanja parova i magnetskim poljem koje djeluje na spinove. Oni zajedno čine parametre Isingovog modela.

U običnom (standardnom) Isingovom modelu cilj je odrediti opservable poput magnetizacije i korelacije parova iz Boltzmannove distribucije. Parametri modela su unaprijed poznati. U inverznom Isingovom modelu, parametri nisu poznati te ih treba odrediti iz opservacija stanja spina. Kada pričamo o inverznom Isingovom problemu u ravnoteži, tada pretpostavljamo da se stanja spina biraju neovisno o prethodnim stanjima. Broj primjera označavamo s N, a skup mjerenja s $D = \boldsymbol{s}^{\mu}$ za μ od 1 do N. Općenito, ni parametri modela ni struktura mreže nisu poznati. U većini standardnih Isingovih problema spinovi čine konačno dimenzionalnu rešetku, tj. postoji jasna udaljenost između spinovima. To nije slučaj s inverznim problemima. Vezanja su često opisana kao potpuno povezani graf, gdje svaki spin interagira sa svakim drugim spinom, i to sa svakim s različitom jačinom J_{ij}. Međutim, u nekim slučajevima, većina vezanja može biti jednaka nuli, i mreža može tvoriti (barem lokalno) stablastu strukturu. Različite konfiguracije mreže utječu na učinkovitost metode traženja parametara.

Inverzni Isingov problem je problem statističke inferencije [11]. Jedna od najčešćih metoda koje se koriste za riješavanje ovakih problema je *procjenitelj najveće izglednosti* (maximum likelihood estimator). Međutim, kompleksnost egzaktnog riješenja je 2^M, gdje je M broj spinova. Postoje mnoge aproksimacije koje pojednostavljuju traženje parametara te smanjuju kompleksnost algoritma.

Izračuni u statističkoj fizici se značajno pojednostave uvođenjem termodinamičkih potencijala. Moguće je pokazati da je procjenitelj najveće izglednosti zapravo samo transformacija termodinamičkog potencijala. Helmholtzova slobodna energija je najkorisniji termodinamički potencijal u standardnom Isingovom problemu, tj. kada su vezanja spinova i magnetska polja poznata $F(\boldsymbol{J}, \boldsymbol{h}) = -\ln Z(\boldsymbol{J}, \boldsymbol{h})$. Derivacije slobodne energije daju magnetizacije, korelacije te ostale opservable. Termodinamički potencijal koji je najkorisniji za inverzni Isingov problem, kada su dane korelacije $\boldsymbol{\chi}$ i magnetizacije $\boldsymbol{m}$, jest Legendreova transformacija Helmholtzove slobodne energije [12]

$$S(\boldsymbol{\chi}, \boldsymbol{m}) = \min_{\boldsymbol{J}, \boldsymbol{h}} \left[-\sum_i h_i m_i - \sum_{i<j} J_{ij}\chi_{ij} - F(\boldsymbol{J}, \boldsymbol{h}) \right]. \tag{3.5}$$

Primijetimo da je ovo upravo funkcija entropije, koja do na predznak daje najvjerojatnije parametre modela. Transformacija dana jednadžbom (3.5) povezuje inferenciju pomoću procjenitelja najveće izglednosti i statističke fizike Isingovog modela koja

je opisana Helmholtzovom energijom. Magnetska polja i vezanja spinova su dana deriviranjem potencijala

$$\begin{aligned} J_{ij} &= -\frac{\partial S}{\partial \chi_{ij}}(\boldsymbol{\chi}, \boldsymbol{m}) \\ h_i &= -\frac{\partial S}{\partial m_i}(\boldsymbol{\chi}, \boldsymbol{m}) \end{aligned}, \tag{3.6}$$

gdje su upotrijebljene uzoračke korelacije i magnetizacije. Ove relacije slijede iz inverzne transformacije

$$F(\boldsymbol{J}, \boldsymbol{h}) = \min_{\boldsymbol{J}, \boldsymbol{h}} \Big[-\sum_i h_i m_i - \sum_{i<j} J_{ij}\chi_{ij} - S(\boldsymbol{\chi}, \boldsymbol{m}) \Big]. \tag{3.7}$$

U praksi nije potrebno napraviti Legendreovu transformaciju slobodne energije s obzirom na vezanja i magnetska polja jer se derivacije slobodne energije po vezanjima mogu prikazati preko derivacija po magnetskom polju

$$\frac{\partial F}{\partial J_{ij}}(\boldsymbol{J}, \boldsymbol{h}) = \frac{\partial^2 F}{\partial h_i \partial h_j}(\boldsymbol{J}, \boldsymbol{h}) - \frac{\partial F}{\partial h_i}(\boldsymbol{J}, \boldsymbol{h}) \frac{\partial F}{\partial h_j}(\boldsymbol{J}, \boldsymbol{h}). \tag{3.8}$$

Stoga, termodinamika inverznog Isingovog problema se može opisati kao jednostruka transformacija slobodne energije, tj. Gibbsova energija:

$$G(\boldsymbol{J}, \boldsymbol{m}) = \max_{\boldsymbol{h}} \Big[\sum_i h_i m_i + F(\boldsymbol{J}, \boldsymbol{h}) \Big]. \tag{3.9}$$

Magnetska polja su dana prvom derivacijom Gibbsove energije

$$h_i = \frac{\partial G}{\partial m_i}(\boldsymbol{J}, \boldsymbol{m}). \tag{3.10}$$

Kako bi pronašli vezanja spinova promotrimo drugu derivaciju Gibbsovog potencijala

$$\frac{\partial^2 G}{\partial m_j \partial m_i}(\boldsymbol{J}, \boldsymbol{m}) = (\boldsymbol{C}^{-1})_{ij}, \tag{3.11}$$

gdje je $\boldsymbol{C}$ matrica povezanih korelacija $C_{ij} \equiv \chi_{ij} - m_i m_j$. Ovaj rezultat slijedi iz teorema o inverznim funkcijama

$$\left[\frac{\partial(h_1, ..., h_M)}{\partial(m_1, ..., m_M)} \right]_{ij} = \left[\Big(\frac{\partial(m_1, ..., m_M)}{\partial(h_1, ..., h_M)} \Big)^{-1} \right]_{ij} \tag{3.12}$$

i teorije linearnih odziva

$$C_{ij} = \frac{\partial m_j}{\partial h_i}(\boldsymbol{J}, \boldsymbol{h}) = -\frac{\partial^2 F}{\partial h_i \partial h_j}(\boldsymbol{J}, \boldsymbol{h}), \tag{3.13}$$

što povezuje susceptibilnost magnetizacije malim promijenama u magnetskom polju s korelacijama [13]. Jednadžba (3.11) je centralna jednadžba za mnoge metode rješavanja inverznog Isingovog modela. Lijeva strana jednadžbe je zapravo funkcija jakosti vezanja spinova J_{ij}. Ako Gibbsova energija moze biti izračunata ili aproksimirana, jednadžba (3.11) se može riješiti, te se mogu izračunati vezanja spinova J_{ij}. Slično tome, jednadžba (3.10) uz poznavanje magnetizacije m_i i vezanja spinova J_{ij} daje magnetska polja sustava. Time smo u potpunosti rekonstruirali parametre Isingovog modela

U slijedeća tri podpoglavlja ćemo opisati tri aproksimativne metode za određivanje parametara Isingovog modela. U četvrtom podpoglavlju ćemo opisati programski paket *conIII* koji služi za određivanje parametara inverznog Isingovog modela. U petom podpoglavlju ćemo usporediti rezultate različitih metoda.

3.1.1 Teorija srednjeg polja

Polazna točka je ansatz za Boltzmannovu distribuciju (3.3) [14]

$$p^{MF}(\boldsymbol{s}) = \prod_i \frac{1 + \widetilde{m}_i s_i}{2}, \tag{3.14}$$

tj. pretpostavljamo da su varijable spina statistički neovisne jedna o drugoj. Parametar $\widetilde{m}_i$ opisuje magnetizaciju spina. Svaki spin ima magnetizaciju koja je rezultat efektivnog magnetskog polja koji djeluje na taj spin. Ovo polje dolazi od lokalnog magnetskog polja h_i te od vezanja s drugim spinovima. Srednje polje je usrednjena vrijednost tipičnih konfiguracija efektivnog polja. Koristeći ansatz srednjeg polja možemo izračunati Gibbsovu slobodnu energiju. Minimizacijom Gibbsove slobodne energije možemo pronaći optimalne parametre. Potrebno je uvesti ograničenja na magnetizaciju $\mathbf{m}$, točnije $\tilde{\boldsymbol{m}} = \mathbf{m}$. Gibbsova energija je dana s

$$G^{MF}(\boldsymbol{m}, \boldsymbol{J}) = -\sum_{i<j} J_{ij} m_i m_j + \sum_i \left[\frac{1+m_i}{2} \ln \frac{1+m_i}{2} + \frac{1-m_i}{2} \ln \frac{1-m_i}{2} \right]. \tag{3.15}$$

Jednadžba za vezanja spinova dana je drugom derivacijom Gibbsove slobodne energije $G^{MF}(\boldsymbol{m}, \boldsymbol{J})$

$$(\boldsymbol{C}^{-1})_{ij} = -J_i^{MF} j, (i \neq j). \tag{3.16}$$

Rekonstrukcija magnetskog polja dolazi iz prve derivacije Gibbsove slobodne energije $G^{MF}(\boldsymbol{m}, \boldsymbol{J})$ po m_i

$$h_i^{MF} = -\sum_{i \neq j} J_{ij}^{MF} m_j + \operatorname{arctanh} m_i. \tag{3.17}$$

Ovaj rezultat izravno povezuje izmjerene korelacije i jačinu vezanja spinova. Kompleksnost algoritma je $O(M^3)$, što je puno manja složenost od eksponencijalne složenosti analitičkog riješena.

3.1.2 TAP rekonstrukcija

Varijacijska procjena Gibbsove energije (3.15) može se poboljšati. Thouless, Anderson i Palmer (TAP) su 1977. godine u svom radu opisali dodatni član u Gibbsovoj energiji.

$$G^{TAP}(\boldsymbol{J}, \boldsymbol{m}) = G^{MF} - \frac{1}{2} \sum_{i<j} J_{ij}^2 (1 - m_i^2)(1 - m_j^2). \tag{3.18}$$

Ovaj član se može interpretirati kao efekt fluktuacije spina na magnetizaciju spina zbog utjecaja na susjedne spinove [15]. Naziva se Onsagerov član. Za standardni Isingov problem, ovaj član mijenja izraz za magnetizaciju u takozvanu TAP jednadžbu

$$m_i^{TAP} = \tanh \left(h_i + \sum_{i \neq j} J_{ij} m_j^{TAP} - m_i TAP \sum_j J_{ij}^2 (1 - (m_j^{TAP})^2) \right). \tag{3.19}$$

U inverznom problemu, s TAP slobodnom energijom ,jednadžba koja definira vezanja spinova J_{ij} je

$$(\boldsymbol{C}^{-1})_{ij} = -J_{ij}^{TAP} - 2(J_{ij}^{TAP})^2 m_i m_j. \tag{3.20}$$

Rješavanjem ove kvadratne jednadžbe dobivamo TAP rekonstrukciju [16]

$$J_{ij}^{TAP} = \frac{-2(\boldsymbol{C}^{-1})_{ij}}{1 + \sqrt{1 - 8(\boldsymbol{C}^{-1})_{ij} m_i m_j}}, \tag{3.21}$$

gdje smo izabrali riješenje koje se slaže s aproksimacijom srednjeg polja kada je magnetizacija nula. Magnetska polja su dana jednadžbom:

$$h_i = \operatorname{arctanh}(m_i) - \sum_{j \neq i} J_{ij}^{TAP} m_j + m_i \sum_{j \neq i} (J_{ij}^{TAP})^2 (1 - m_j^2). \tag{3.22}$$

3.1.3 Bethe–Peierls ansatz

Najveći problem kod egzaktnog pronalaženja procjenitelja najveće izglednosti jest što postoji mogućnost petlji u grafu. Zbog petlji vremenska kompleksnost problema može rasti brže nego eksponencijalno u ovisnosti o veličini sustava. Grafovi za koje se korelacije mogu pronaći i za velike sustave, u realnom vremenu, su aciklični grafovi, tj. stabla.

Promotrimo Isingov model čije korelacije tvore stablo. Graf se može sastojati od nekoliko nepovezanih dijelova, ili od jednog povezanog stabla, ali je važno da ne sadrži petlje. Skup vrhova stabla T označavamo s V_T, a skup bridova s E_T. Može se lako pokazati da se Boltzmannova distribucija za Isingov model može napisati kao [17, 18]

$$p_T(\boldsymbol{s}) = \prod_{i \in V_T} p_i(s_i) \prod_{(i,j) \in E_T} \frac{p_{ij}(s_i, s_j)}{p_i(s_i) p_j(s_j)} = \prod_{(ij) \in E_T} p_{ij}(s_i, s_j) \prod_{i \in V_T} p_i(s_i)^{1-|\partial i|}, \tag{3.23}$$

gdje smo s ∂i označili skup svih susjeda spina i, a s $|\partial i|$ smo označili broj susjeda spina i.

Ovu distriuciju možemo iskoristiti kao ansatz za Boltzmannovu distribucije. U ovom slučaju se distribucija naziva Bethe-Peierls ansatz. Uređeni par $(i, j) \in E$ ide preko svih parova spinova koji interagiraju, tj. preko svih bridova grafa. Moguće je parametrizirati distribucije p_i i p_j koristeći magnetizacijski parametar $\tilde{m}_i$ i korelacijski parametar $\tilde{C}_{ij}$.

$$\begin{aligned} p_i(s_i) &= \frac{1 + \tilde{m}_i s_i}{2} \\ p_{ij}(s_i, s_j) &= \frac{(1 + \tilde{m}_i s_i)(1 + \tilde{m}_j s_j) + \tilde{C}_{ij} s_i s_j}{4} \end{aligned} \tag{3.24}$$

uz uvjete

$$\begin{aligned} -1 \leq \tilde{m}_i \leq 1 \\ -1 + |\tilde{m}_i + \tilde{m}_j| \leq \tilde{C}_{ij} + m_i m_j \leq +1 - |\tilde{m}_i - \tilde{m}_j|. \end{aligned} \tag{3.25}$$

Bethe - Peierls ansatz se može usporediti s ansatzom srednjeg polja koji pridodjeljuje magnetizaciju svakom spinu. Bethe - Peierls ansatz ide korak dalje. On pridodaje svakom paru spinova korelaciju i magnetizaciju, koje su zatim određene samodosljedno. Važna karakteristika Bethe - Peierls ansatza je da se Shannonova entropija može razdvojiti u parove spinova

$$S[p^{BP}] = \sum_i S[p_i] + \sum_{i,j}(S[p_{ij}] - S[p_i] - S[p_j]). \tag{3.26}$$

Bethe - Peierls ansatz je dobro definiran i egzaktan kada struktura mreže ima oblik stabla. Kada graf sadrži petlje, distribucija vjerojatnosti nije normalizirana te ansatz nije dobro definiran.

Krećemo s pretpostavkom da mreža ne sadrži petlje. Kako bi riješili inverzni Isingov problem koristimo Bethe - Peierls anzatz kao varijacijski ansatz za minimizaciju Gibbsove energije uz uvjet $\tilde{\boldsymbol{m}} = \boldsymbol{m}$. Gibbsovu energiju minimiziramo preko $\tilde{\boldsymbol{C}}$, što daje

$$\begin{aligned} G^{BP}(\boldsymbol{J}, \boldsymbol{m}) = \sum_{(ij)} J_{ij}(C_{ij}^{BP} + m_i m_j) + \sum_i (1 - z_i) \_ s_i \frac{1 + m_i s_i}{2} \ln \frac{1 + m_i s_i}{2} + \\ \sum_{i,j} \sum_{s_i, s_j} \frac{(1 + m_i s_i)(1 + m_j s_j) + C_{ij}^{BP} s_i s_j}{4} \ln \frac{(1 + m_i s_i)(1 + m_j s_j) + C_{ij}^{BP} s_i s_j}{4}, \end{aligned} \tag{3.27}$$

gdje je $\boldsymbol{C}^{BP}$ optimalna vrijednost $\tilde{\boldsymbol{C}}$ i zadovoljava

$$J_{ij} = \sum_{s_i, s_j} \frac{s_i s_j}{4} \ln \frac{(1 + m_i s_i)(1 + m_j s_j) + C_{ij}^{BP} s_i s_j}{4}. \tag{3.28}$$

S z_i označavamo broj susjednih spinova spina i. Druga derivacija Gibbsove energije daje izraz za inverz korelacije

$$(\boldsymbol{C}^{-1}) = \frac{C_{ij}^{BP}}{(C_{ij}^{BP})^2 - (1 - m_i^2)(1 - m_j^2)} () j \neq i. \tag{3.29}$$

Riješenje ove kvadratne jednadžbe je

$$C_{ij}^{BP} = \frac{1}{2} \left\{ \frac{1}{(\boldsymbol{C}^{-1})_{ij}} - \sqrt{\frac{1}{(\boldsymbol{C}^{-1})_{ij}}^2 - 4(1 - m_i^2)^2 (1 - m_j^2)^2} \right\}, \tag{3.30}$$

Za $(\boldsymbol{C}^{-1})_{ij} \neq 0$. Za $(\boldsymbol{C}^{-1})_{ij} = 0$ riješenje je C_{ij}^{BP}. Uvrštavanjem riješenja u jednadžbu (3.15) te uz poseban slučaj $m_i = 0$ dobivamo

$$J_{ij}^{BP} = -\frac{1}{2}\operatorname{arcsinh}[2(\boldsymbol{C}^{-1})_i j], (j \neq i). \tag{3.31}$$

U teoriji grafova, ova formula je povezana s izrazom za udaljenost u stablu u kojem su težine dane korelacijom između spinova [19]. Magnetsko polje je dano prvom derivacijom Gibbsove energije

$$h_i^{BP} = (1 - z_i)\operatorname{arctanh} m_i - \sum_{j\in\partial i} J_{ij}^{BP} m_j + \sum_{j\in\partial i}\sum_{s_i,s_j} \frac{s_i + m_j s_i s_j}{4} \ln \frac{(1 + m_i s_i)(1 + m_j s_j) + C_{ij}^{BP} s_i s_j}{4}. \tag{3.32}$$

3.1.4 Programski paket: conIII

ConIII je programski paket u Pythonu koji služi za rješavanje modela maksimalne entropije, poglavito Isingovog modela [20, 21]. Modeli maksimalne entropije je naziv za skupinu modela koji su konzistenti sa statistikom sustava, ali van toga imaju što je moguće manje strukture. Krećemo od K važnih značajki koje dobijemo iz podataka $f_k(s)$, koje model kojeg razvijamo mora zadovoljiti. U našem slučaju su to magnetizacije neurona i međusobne korelacije. Najčešće se rješavaju preko Lagrangeovih multiplikatora. Model maksimalnog procjenitelja upravo pripada modelima maksimalne entropije.
Korištena su tri modela iz programskog paketa conIII:

1. **Model pseudo-vjerojatnosti (eng. Pseudolikelihood)**
   Model pseudo-vjerojatnosti je analitička aproksimacija vjerojatnosti koja značajno smanjuje kompleksnost traženja riješenja, te se približava egzaktnom riješenju kako broj neurona M raste, tj. kada $M \to \inf$ je jednaka egzaktnom riješenju.
   Izračunajmo uvjetnu vjerojatnost da se spin i nalazi u stanju s_i uz uvjet na ostala stanja sustava $\{s_{j\neq i}\}$

   $$p(s_i|\{s_{j\neq i}\}) = (1 + e^{-2s_i(h_i + \sum_{j\neq i} J_{ij} s_j)})^{-1}. \tag{3.33}$$

   Logaritmiranjem ovog izraza, te zbrajanjem po svim primjerima u podatcima,

dolazimo do izraza:

$$f(h_i, \{J_{ij}\}) = \sum_{r=1}^{N} \ln p(s_i^{(r)} | \{s_{j\neq i}\}^{(r)}). \tag{3.34}$$

U limesu kada su podatci dobra reprezentacija sustava, usrednjavanje po podacima može biti zamijenjeno usrednjavanjem po asamblu.:

$$f(h_i, \{J_{ij}\}) = \sum_{s} \ln p(s_i^{(r)} | \{s_{j\neq i}\}) p(s; h_i, \{J_{ij}\}). \tag{3.35}$$

Kako bi pronašli točku najveće izglednosti za spin i, moramo izračunati gradijent $\partial f / \partial J_{ij}$ i Hessian $\partial f / \partial J_{ij} \partial J_{i'j'}$ za Newtnovu metodu konjugiranog gradijentnog spusta. Nakon što maksimiziramo izglednost za svaki spin, parametri ne moraju zadovoljavati simetričnost: $J_{ij} = J_{ji}$. Simetriju namećemo tako da definiramo:

$$J'_{ij} = (J_{ij} + J_{ji})/2. \tag{3.36}$$

Model pseudo-vjerojatnosti je ekstremno brz i često iznenađujuće precizan. Kompleksnost računanja gradijenta je $O(NM^2)$, a Hessiana $O(NM^3)$.

2. **Model minimalnog protoka vjerojatnosti**

   Model minimalnog protoka vjerojatnosti, ili na eng. minimum probability flow (MPF) uključuje analitičku aprokskimaciju kako se distribucija vjerojatnosti mijenja kako mijenjamo konfiguracije sustava. Prvi korak MPF-a jest postavljanje skupa dinamika koje će izjednačiti distribucije podataka s distribucijom modela. Kada su ove distribucije jednake, kažemo da nema "toka vjerojatnosti" između njih.

   Krećemo s distribucijom stanja koja nam je dana iz mjerenja, te na njoj primjenjujemo Monte Carlo dinamiku. Kada su podatci i model različiti, vjerojatnost će teći između njih i indicirati da se parametri modela moraju promijeniti. Minimizirajući derivaciju Kullback-Leiber divergencije, mjerimo kako se razlika između modela i stanja u podatcima mijenja ako se dinamika vrti infinitezimalno mali dio vremena. Ideja je da je i ova derivacija isto minimizirana s optimalnim parametrima. Kompleksnost algorima je $O(NGM^2)$, gdje je G broj veza u sustavu. Za velike, u potpunosti povezane sustave je $G \approx 2^N$, pa je za

ovu metodu potrebno da je matrica rijetka (eng. sparse).

3. **Model prilagodljivog razvoja klastera**

   Prilagodljivi razvoj klustera iterativno računa članove u razvoju entropije S:

$$S - S_0 = \sum_{\Gamma} \Delta S_{\Gamma}, \tag{3.37}$$

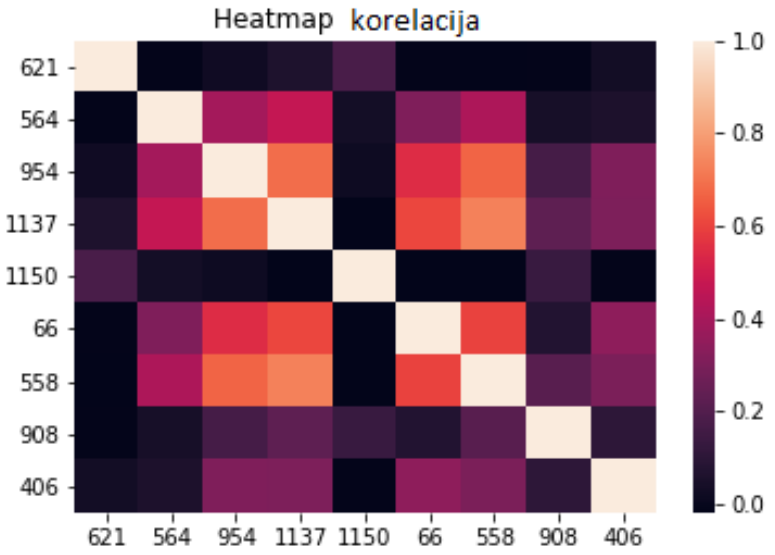
   gdje je suma po klasterima Γ. U egzaktnom slučaju imamo $2^N - 1$ mogućih nepraznih podskupova u sustavu. U najjednostavnijoj verziji razvoja, entropija se razvija oko vrijednosti $S_0 = 0$.

   Inverzni Isingov problem se može riješiti za svaki klaster zasebno i to egzaktno kada su klasteri mali. Zatim se ovi rezultati koriste kako bi se u potpunosti konstruirala matrica interakcije J_{ij}. Razvoj počinje s malim klasterima te se proširuje do sve većih klastera, zanemarujući bilo koji klaster kojem je doprinos entropije ΔS_{Γ} manji od zadanog praga. Kako bi se pronašlo najbolje riješenje koje nije prenaučeno, prag je inicijalno postavljen na veliku vrijednost, te se zatim smanjuje, postepeno dodavajući sve više klastera, sve dok ne pronađemo optimum.

   Kompleksnost algoritma ovisi o veličini klastera u razvoju entropije. Ako je razvoj ograničen na veličinu klastera m, složenost algorima je u najgorem slučaju $O(\binom{M}{m} 2^m)$.

3.1.5 Usporedbe metoda

Kako je samo za male sustave moguće egzaktno riješiti inverzni Isingov model, prvo ćemo promotriti sustav od devet neurona. Neurone smo odabrali tako da smo tražili skup neurona koji imaju najveću srednju vrijednost korelacije. Traženi skup je moguće pronaći egzaktno, koristeći algortime pretraživanja prostora poput pretraživanja u dubinu, pretraživanja u širinu, te A*. Međutim, kako bi pretražili sav prostor stanja, broj mogućih kombinacija je 5.1×10^{27}. Matrica korelacije je prikazana na slici (3.1). Mreža je vizualizirana na slici (3.2). Moguće je dodatno optimizirati pretraživanje prostora te smanjiti broj mogućih stanja potrebnih za pretražiti. Kako bi smanjili kompleknsot traženja skupa kojeg ćemo analizirati, izabrali smo 100,000,000 slučajnih skupova od 9 neurona, te smo od svih tih skupova izabrali onaj skup koji ima najveću

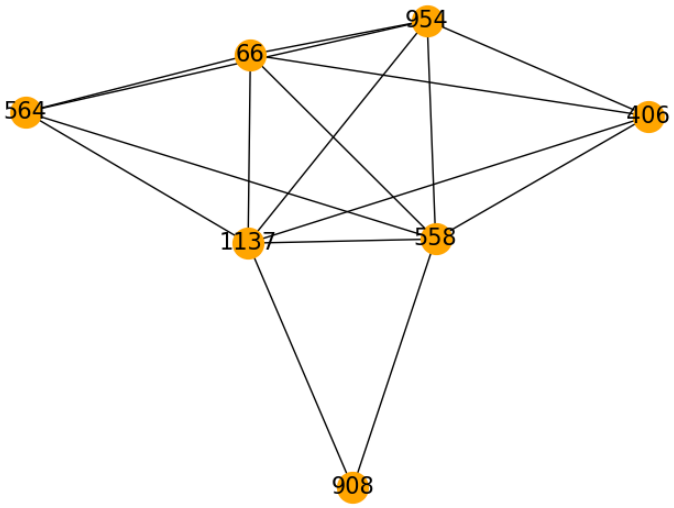


Slika 3.1: Korelacija neurona u skupu koji analiziramo. Izvor: obrada autora

srednju vrijednost korelacija. Izabrani skup ima srednju korelaciju jednaku 0.84.

Usporedba raznih aproksimativnih metoda s egzaktnim rješenjem je prikazana na slici (3.3). Možemo primijetiti da sve metode za većinu parametara aproksimiraju u dobrom rasponu, tj. aproksimirani parametri imaju isti red veličine kao i parametri dobiveni egzaktnom optimizacijom. Međutim, ni jedna metoda ne može dobro aproksimirati parametre koji su manji od -10. Na primjer, ekstremne vrijednosti parametara dobivenih TAP rekonstrukcijom i aproksimacijom srednjeg polja su pozitivne i veće od 200, odnosno 1400. Takvi parametri govore da neuroni žele biti što više aktivni, što nije slučaj. Ostale metode ne mogu aproksimirati parametre manje od -20. Najbolji rezultati su ostvareni koristeći Bethe-Peierls ansatz i metodu prilagodljivog razvoja klastera. Bethe-Peierls ansatz je primjenjiv pod pretpostavkom da je struktura mreže stablasta, tj. da ne sadrži petlje. Metoda prilagodljivog razvoja klastera pretpostavlja da je graf zapravo građen od mnogo podgrafova, koji su međusobno slabo povezani, ali su untar sebe snažno povezani. Mi preptostavljamo da su svi neuroni međusobno povezani. Stoga, možemo zaključiti da je mreža građena od mnogo manjih podgrafova, koji su zapravo stablaste strukture.

Na slici (3.4) vidimo usporedbu kumulativnih histograma između egzaktno pronađenih parametara i parametara koje dobijemo metodom prilagodljivih klastera. Aproksimativna metoda je uhvatila trend parametara, tj. priroda interkacije između neurona je dobro aproksimirana. Ono što nije dobro aproksimirano jest iznos parametara,



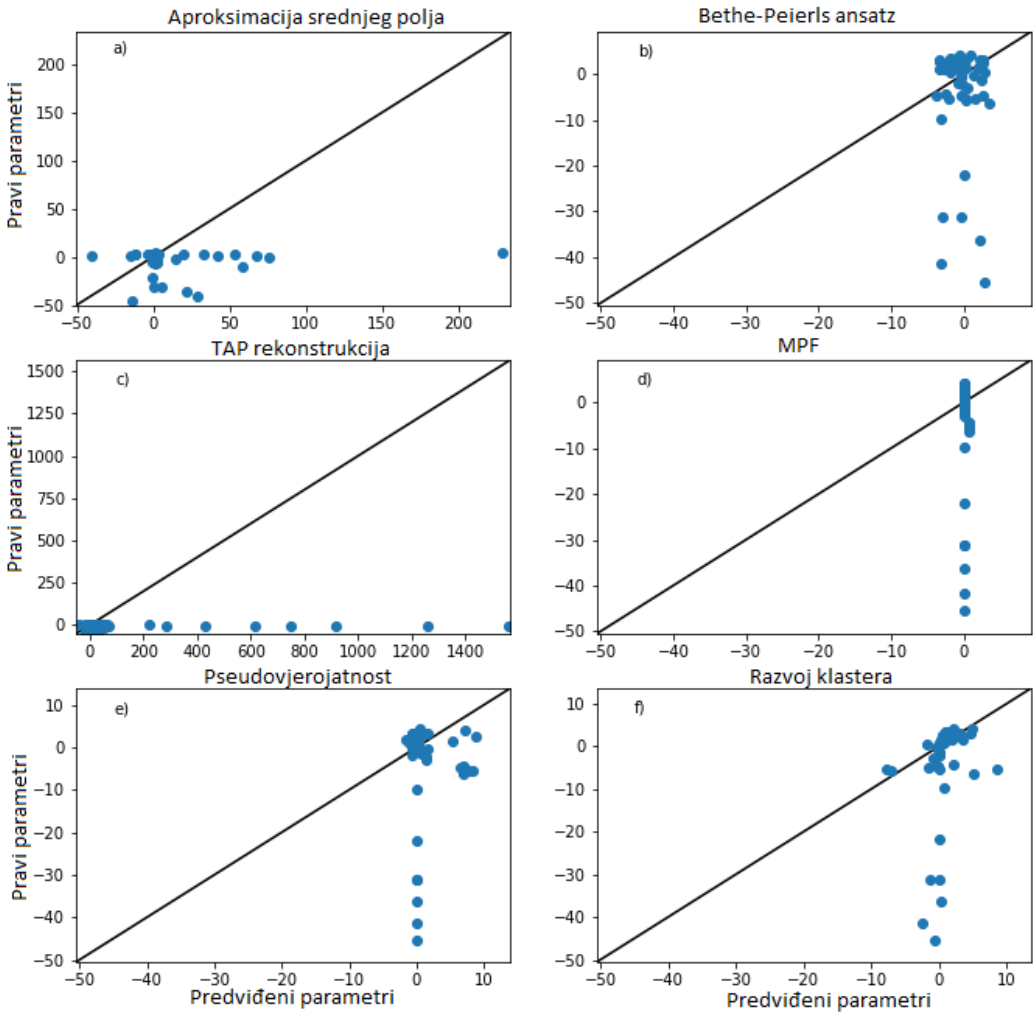
Slika 3.2: Vizualizacija mreže. Sve veze s težinom manjom od 0.2 su odbačene. Korelacija između neurona je korištena za definiranje veze u grafu. Izvor: obrada autora

aproksimirani parametri nikad nisu veći, po apsolutnoj vrijednosti, od 10.

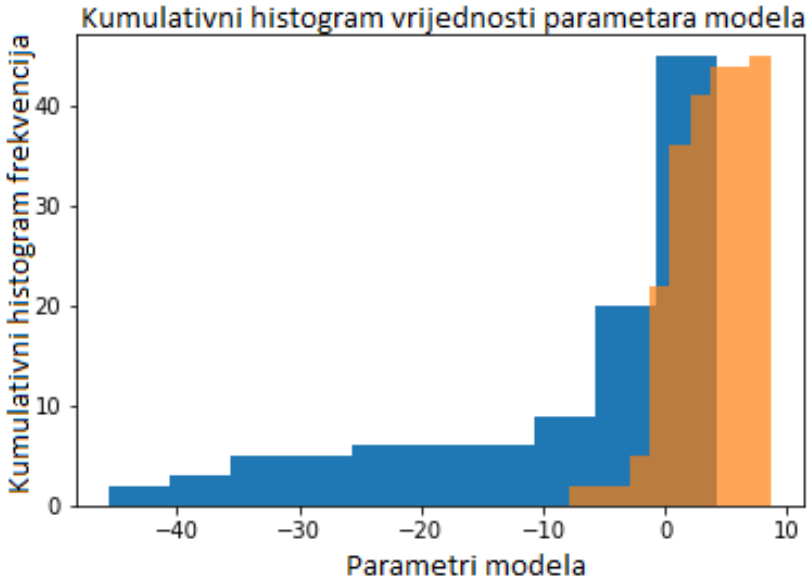
Slijedeće pitanje koje se postavlja jest koliko su razlikuju slučajno izabrana stanja koristeći egzaktne parametre i parametre koje smo dobili aproksimativnim metodama. Rezultati za deset najčešćih su dani u tablici (3.1). Izabrano je 10,000,000 slučajnih primjera. Vjerojatnost da se jedno stanje izabere su dobivene Boltzmannovom distribucijom koristeći egzaktne i aproksimativne parametre. Možemo odmah primjetiti da je najčešće stanje, stanje potpunog mirovanja, tj. vrijednosti svih neurona su 0. Također možemo primijetiti da su stanja s jednim aktivnim neuronom, dobivena egzaktnim parametrima, identična stanjima s jednim aktivnim neuronima, dobivenim aproksimativno metodom. Iz ovoga možemo zaključiti da su magnetizacije dobivene aproksimativnom metodom, valjana reprezentacija egzaktnih magnetizacija. Aproksimativna metoda pridaje manju vjerojatnost stanjima koja imaju više od dva neurona aktivna.

3.2 Rekonstrukcija parametara ako se sustav nalazi van ravnoteže

U prošlom poglavlju smo pretpostavljali da stanje mreže u trenutku t, $\boldsymbol{\sigma}^t$, ne ovisi o stanju mreže u trenutku $\boldsymbol{\sigma}^{t-1}$. Također smo pretpostavljali da su vezanja spinova simetrična, tj. $J_{ij} = J_{ji}$. Za neuronske mreže, ove pretpostavke nisu u potpunosti točne, pogotovo za mjerenja kalcija. Koncentracija kalcija se sporo mijenja u neuronima. Zbog toga se stanje u trenutku t slabo razlikuje od stanja u trenutku $t+1$,



Slika 3.3: Usporedba šest aproksimativnih metoda s parametrima koji su pronađeni egzaktnom optimizacijom: a) Aproksimacija srednjeg polja b) Bethe-Peierls ansatz c) TAP rekonstrukcija d) Model minimalnog protoka vjerojatnosti e) Pseudo-vjerojatnost f) Prilagodljiv razvoj klastera. Egzaktni parametri su manji od 10, te su po apsolutnoj vrijednosti manji od 50. Primijetimo da parametri koji su dobiveni metodama a i c imaju ekstremne pozitivne vrijednosti (a>200, c>1400), što je suprotno ponašanje od parametara koje smo dobili egzakntom optimizacijom. Parametri dobiveni metodama b,d,e i f su uvijek između -10 i 10. Parametri koje nisu dobro aproksimirani su parametri manji od -20. Pseudo-vjerojatnost i MPF, parametre koji su stvarno manji od -10 ne mogu aproksimirati. Metoda prilagodljivog razvoja klastera i Bethe-Peierls ansatz daju najbolje rezultate. Izvor: obrada autora



Slika 3.4: Kumulativni histogram parametara vezanja modela J i magnetizacije h. Plava boja označava parametre koji su dobiveni egzaktnom optimizacijom, a narančasta parametre koji su dobiveni aproksimacijom prilagodljivih klastera. Vidimo da je distribucija parametara slična. Najveća razlika je u tome sta parametri dobiveni aproksimativnom metodom nikad nisu manji od -10. Izvor: obrada autora

	Samples ACE	frequency	Samples exact optimisation	frequency
0	[0,0,0,0,0,0,0,0,0,0]	66.8%	[0,0,0,0,0,0,0,0,0,0]	91.7%
1	[1,0,0,0,0,0,0,0,0,0]	5.7%	[0,0,0,0,0,0,0,0,0,1]	1.4%
2	[0,1,0,0,0,0,0,0,0,0]	5.2%	[0,0,0,0,1,0,0,0,0]	0.9%
3	[0,0,0,0,0,0,0,0,1,0]	4.9%	[0,0,0,0,0,0,0,0,1,0]	0.8%
4	[0,0,0,0,0,1,0,0,0,0]	4.8%	[1,0,0,0,0,0,0,0,0,0]	0.8%
5	[0,0,0,0,1,0,0,0,0,0]	3.3%	[0,0,0,0,0,0,1,0,0]	0.4%
6	[0,0,1,0,0,0,0,0,0,0]	1.6%	[0,1,0,0,0,0,0,0,0,0]	0.4%
7	[1,0,0,0,1,0,0,0,0,0]	1.0%	[0,0,1,0,0,0,0,0,0,0]	0.4%
8	[0,0,0,0,1,0,0,1,0]	0.8%	[0,0,1,1,0,1,1,0,1]	0.3%
9	[0,1,0,0,1,0,0,0,0,0]	0.3%	[0,0,0,0,0,1,0,0,0]	0.3%

Tablica 3.1: Tablica deset stanja s najvećom vjerojatnosti. Prvi i drugi stupac su stanja i pripadne relativne frekvencije za stanja dobivena metodom prilagodljivog razvoja klastera. Treći i četvrti stupac opisuju egzaktna stanja i pripadne relativne frekvencije. Primjetimo da su stanja s jednim aktivnim neuronom najčešća. Također, za egzaktna stanja, stanje kompletnog mirovanja je najčešće stanje s udjelom od 91.7%. Za aproksimativnu metodu, stanje potpunog mirovanja, zauzima dvije trećine stanja. Promatrani skup podataka je uglavnom neaktivan, te se rijetko pojedini neuroni pale i postaju aktivni. Izvor: obrada autora

jer se koncentracija kalcija još nije stigla promijeniti. Također, ne postoji razlog zbog kojeg bi pretpostavljali da neuron i utječe na neuron j na isti način kao i neuron j na neuron i. Štoviše, ako sinaptička veza postoji s neurona i na neuron j, ne mora

značiti da ta veza uopće postoji u obratnom slučaju. Nadalje, rekonstrukcija parametara kada je sustav van ravnoteže se primjenjuje u slučajevima kada su dostupna mjerenja u vremenu, neovisno o tome je li sustav u ravnoteži ili ne. U ovom poglavlju ćemo se ukratko osvrnuti na jednu od metoda rekonstrukcije parametara van ravnoteže.

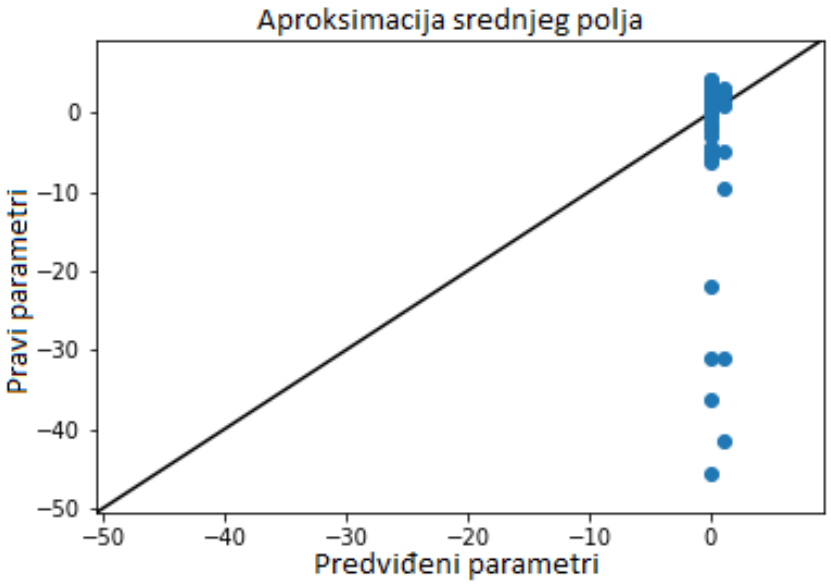
Jedna od aproksimacija koju je moguće primijeniti na vremenski ovisne podatke je teorija srednjeg polja. Definiramo vremensku korelaciju parova kao

$$\phi_{ij} = \langle \sigma^t \sigma^{t+1} \rangle. \tag{3.38}$$

Moguće je pokazati da razvojem vremenske korelacije oko srednjeg polja možemo dobiti izraz za parametre vezanja parova spina

$$\boldsymbol{J}^{MF} = \boldsymbol{A}^{-1}\boldsymbol{D}\boldsymbol{C}^{-1}, \tag{3.39}$$

gdje je $A_{ij} = \delta_{ij}(1 - m_i^2)$, $D_{ij} = \phi_{ij} - m_i m_j$ te $C_{ij} = \chi_{ij} - m_i m_j$ [22]. Na slici (3.5) vidimo usporedbu parametara dobivenih optimizacijom parametara koristeći aproksimaciju srednjeg polja van ravnoteže. Primjećujemo da su rezultati veoma slični onima koje smo dobili koristeći teoriju srednjeg polja u ravnoteži. Stoga možemo zaključiti da je za ovaj dani sustav promatranje u ravnoteži ili van nje manje bitno od korištenja metoda koje su prikladne za strukturu neuronske mreže. Zanimljivo je primijetiti da su parametri dobiveni aproksimativnom metodom grupirani oko 1 ili 0.



Slika 3.5: Usporedba parametara dobivenih egzaktnom optimizacijom sustava u ravnoteži s parametrima dobivenim koristeći teoriju srednjeg polja van ravnotežnog stanja. Možemo primijetiti da su parametri dobiveni aproksimativnom metodom grupirani oko 1 ili 0. Izvor: obrada autora

4 Primjena metoda strojnog učenja za rješavanje inverznog Isingovog problema

Iako koristeći Inverzni Isingov model možemo opisati i razumijeti dinamiku neurona postoji nekoliko problema koji ograničavaju njegovu primjenu:

- *Ne postoji način za provijeriti točnost modela osim za male sustave* ($\approx$ M=20).
  Egzaktno riješenje postoji, međutim njegova kompleksnost raste kao (2^M), s brojem neurona. Stoga, za visoko dimenzionalne sustave, ne možemo procijeniti točnost modela, pa ne možemo znati koliko su parametri modela pouzdani.

- *Model konvergira jedino ako postoji jaki signal u sustavu*.
  Model ne konvergira ako se pomno ne izaberu neuroni koji imaju značajnu interakciju. Npr. ako za sustav izaberemo najaktivnije neurone parametri modela se značajno razlikuju od onih koje dobijemo egzaktnom optimizacijom modela.

- *Prevelike pretpostavke na model*.
  Inverzni Isingov model pretpostavlja da su svi neuroni međusobno povezani (i to za ravnotežni sustav pretpostavljamo da su veze simetrične). Za neke od metoda rješavanja pretpostavljamo i arhitekturu sustava. Međutim, biološke neuronske veze su znatno kompliciranije: Signal putuje konačnom brzinom, mreža može sadržavati petlje, neuroni ne primaju i ne šalju signale simultano.

Inverzni Isingov problem je u svojoj suštini zapravo veoma jednostavan. Podatci o sustavu su poznati te želimo razumjeti ponašanje sustava. Ovako postavljen problem je standardni primjer strojnog učenja, samo je potrebno postaviti prava pitanja. Inverzni Isingov problem pretpostavlja da je sustav zapravo u potpunosti povezana mreža čije riješenje tražimo pomoću procjenitelja najveće izglednosti. Optimizacija procjenitelja je moguća egzaktno samo za male sustave. Za sustave veće od $\approx$ 20 neurona moramo raditi dodatne pretpostavke te sustav možemo optimizirati aproksimativno. Ovakav pristup je prerigorozan te se ograničavamo na malen podskup rješenja. Međutim, možemo koristiti napredne metode strojnog učenja koje su univerzalni aproksimator, nisu ograničeni veličinom sustava (red veličine $\approx 10^4$ neurona je još uvijek rješivo) te ih lako možemo testirati.
U ovom poglavlju planiram pokazati kako koristeći metode strojnog učenja možemo

razumjeti dinamiku sustava bez da uvodimo razne pretpostavke na dinamiku sustava. Koristeći algoritme i znanja strojnog učenja te znanosti o podacima možemo, ne samo izjednačiti znanje o sustavu koje dobijemo inverznim Isingovim modelom, već saznati puno više o sustavu. Ovakav pristup ima znatno manje ograničenje na veličinu sustava koje promatramo. Nadalje, jednostavnije je ocijeniti točnost modela koje je znatno veća od točnosti inverznog Isinogovg modela.

4.1 Koja pitanja možemo postaviti?

Rješavanjem ovog problema pomoću inverznog Isingovog modela u potpunosti definiramo sustav te možemo razumjeti kako su neuroni povezani, tražiti klasične fizikalne veličine koje u ovom sustavu možda nisu dobro definirane, te možemo tražiti stanja mreže. Kada modeliramo sustav s pretpostavkom da se sustav nalazi u ravnoteži ne možemo tražiti slijedeća stanja, jedino što možemo je birati veliki broj stanja po vjerojatnostima koje slijede iz modela. Odnosno, u ravnotežnom slučaju ne možemo ni odgovoriti u kojem stanju će se sustav naći u slijedećem trenutku. I to sve uz uvjet da je sustav mali. Za sustave veće od $M \approx 20$ neurona, ne možemo tražiti stanja sustava, jer je prostor stanja prevelik. Sva ova pitanja su veoma općenita, te je zbog toga teško testirati njihove odgovore.
Međutim, pitanja na koja možemo dati odgovore pomoću strojnog učenja su mnogo prirodnija i interpretabilnija. Također je odgovore vrlo jednostavno testirati.

1. **U kojem stanju će se neuron, ili cijela mreža, naći u slijedećem trenutku?**
   Na ovo pitanje donekle možemo odgovoriti s inverznim Isingovim modelom, ali ne u potpunosti. Pretpostavka Isingovog modela jest da buduće stanje ne ovisi o prošlim stanjima, ali znamo da to nije točno, jer miš ima točno predefiniran test koji se sastoji naizmjenice od zadataka i nagrada. Miš kojeg promatramo je već upoznat sa procesom, pa može očekivati što će se dogoditi. Također, mjerenja koja promatramo su mjerenja koncentracije kalcija koja se sporo mijenjaju u vremenu. Stoga, pretpostavka da slijedeće stanje ne ovisi o prošlom nije valjana.
   Prvo što možemo pitati je možemo li iz trenutnog stanja mreže neurona $\boldsymbol{\sigma}^t$ u trenutku t predvidjeti stanje neurona k u trenutku $t+1$, tj. možemo li predvidjeti σ_k^{t+1}? Ako možemo, s kolikom točnošću? Nadalje, možemo pitati: kako

točnost ovisi o broju neurona M koje uzimamo kao ulazne varijable? Možemo li predvidjeti stanje neurona ne samo u slijedećem trenutku nego i u trenucima $t + lag$, gdje je lag broj koraka koliko idemo u budućnost. Kako točnost modela ovisi o parametru lag?

Zatim možemo postaviti pitanje, ne samo možemo li predvidjeti stanje jednog neurona σ_k^t, već stanje cijele mreže $\boldsymbol{\sigma}^{t+1}$. Opet se možemo pitati kako ta točnost ovisi o parametru lag i o veličini sustava M kojeg uzimamo kao ulazne varijable. Posljednje pitanje koje ćemo postaviti jest možemo li predvidjeti iduće stanje neurona σ_k^{t+1} ako kao ulazne varijable koristimo prethodna stanja tog neurona $\sigma_k^t, \sigma_k^{t-1}, \sigma_k^{t-2}, ..., \sigma_k^{t-T}$. Ovo je jedino pitanje koje nećemo promatrati klasifikacijski, već regresijski, tj. nećemo normalizirati podatke na 0 i 1. Zanima nas koji je optimalni T.

2. **Možemo li razlikovati ponašanje neurona za vrijeme rješavanja zadataka i za vrijeme nagrade?**

   Preciznije, možemo li odvojiti i klasificirati svako pojedino stanje $\boldsymbol{\sigma}^t$ u jednu od dvije kategorije: 1.) Rješavanje zadatka 2.) Nagrada.

   Krećemo od a priori pretpostavke da ne znamo ništa o sustavu i podacima, tj. da ne znamo da miš rješava zadatak i dobiva nagradu. Želimo vidjeti možemo li nenadziranim strojnim učenjem odvojiti skupine stanja. Znamo da postoje dva bitno različita stanja, no krenemo li od ove pretpostavke možemo pronaći druge kategorije stanja koje se bitno razlikuju, ako postoje. Možda se neuroni u trenutku dobivanja nagrade znatno razlikuju od trenutaka prije novog zadatka. Nadalje, želimo razviti model koji klasificira svako stanje $\boldsymbol{\sigma}^t$ u zadatak ili nagradu. Zanima nas točnost takvog modela i kako točnost ovisi o veličini sustava M. Osim toga, koristeći najnovije tehnike strojnog učenja, možemo odgovoriti na pitanje aktivnost kojeg neurona implicira da miš rješava zadatak, a kojeg da prima nagradu. Ovo je bitno drugačije od standardnih snimanja mozga koja nam govore koja su područja mozga aktivna za vrijeme koje radnje. Pomoću strojnog učenja možemo reći: "Ako je ovaj neuron aktivan, miš rješava zadatak". Ovu informaciju možemo saznati koristeći SHAP vrijednosti, koje su opisane u dodatku A.

Za svako od ovih pitanja možemo definirati preciznu metriku točnosti.

4.2 Predviđanje stanja neurona

4.2.1 Predviđanje stanja jednog neurona - klasifikacija

Kako bi testirali možemo li pomoću strojnog učenja odgovoriti na pitanje u kojem će se stanju naći neuron σ_k u slijedećem trenutku, tj. u trenutku $t+1$, koristimo 4% najaktivnijih neurona. Predviđamo stanja najaktivnijeg neurona. Sustav promatramo kao Markovljev lanac: promatramo samo kako stanje σ_k^{t+1} ovisi o stanjima $\boldsymbol{\sigma}^t$. Za sada ne promatramo utjecaj stanja sustava u trenutcima prije t ($t-1$,$t-2$,...). Također ne kreiramo nove značajke niti mijenjamo vrijednosti stanja neurona u ikojem pogledu. Sirova stanja neurona koristimo kao ulaz za naš model. Razlog ovome je što pretpostavljamo da stanje u kojem se neuron nalazi direktno utječe na buduća stanja drugih neurona. Naravno, možemo se pitati zašto učestalost mijenjanja stanja, ili srednja vrijednost stanja u kojem se neuron nalazi ne uzimamo kao varijablu za naš model? Prave neuronske mreže su mnogo kompleksnije: imaju petlje, signal putuje kroz njih konačno vrijeme, ne šalju se svi signali simultano. U kasnijem dijelu ovog podpoglavja ćemo promatrati kako prošla stanja neurona utječu na buduća stanja neurona. Za sada promatramo samo kako aktivnost neurona u idućem trenutku ovisi o trenutnom stanju svih ostalih neurona.

Ako neuron nije aktivan u idućem trenutku, on je označen s 0, a ako je aktivan s 1, tj. oznake se nisu promijenile. Ovaj problem u strojnom učenju nazivamo binarna klasifikacija: za svaki ulaz $\boldsymbol{X}$ predviđamo vjerojatnost da će oznaka biti $\boldsymbol{y}=1$. Naš model možemo napisati kao $\boldsymbol{F}(\boldsymbol{X}) = P(\boldsymbol{y}|\boldsymbol{X})$. Funkcija greške koja se koristi u ovim problemima je takozvana binarna unakrsna entropija [23], koja je dana s

$$\boldsymbol{L} = -\sum_i (y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)). \tag{4.1}$$

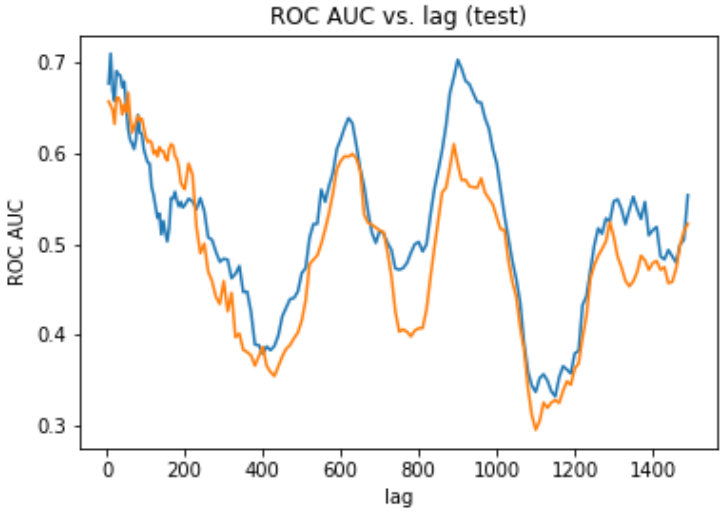
Svaki od modela treniramo na prve dvije trećine mjerenja, a testiramo na posljednoj trećini. Odvajanje na trećine s obzirom na vrijeme je važno kako bi izbjegli prenaučenost modela. Ako bi samo slučajno birali primjere za treniranje i testiranje, može se dogoditi da u skupu podataka za trening imamo primjer stanja $\boldsymbol{\sigma}^{t-1}$ i $\boldsymbol{\sigma}^{t+1}$ a u skupu podataka za testiranje imamo primjer $\boldsymbol{\sigma}^t$. Stanja su sporo promjenjiva, pa je stanje $\boldsymbol{\sigma}^t$ veoma slično susjednim stanjima, što rezultira curenjem informacija između skupa podataka za treniranje i testiranje, pa su rezultati nerealno veći. Re-

zultati osnovnih modela su dani u tablici (4.1). Primijetimo da algoritmi dubokog učenja, SVM i neuronske mreže imaju bolje rezultate od ostalih algoritama. Ovaj rezultat nije iznenađujuć jer su ti algoritmi upravo razvijeni po uzoru na biološke neuronske mreže.

	Name	Accuracy	ROC AUC
0	Nearest Neighbors	89.2%	69.6%
1	Linear SVM	92.1%	84.8%
2	RBF SVM	90.5%	61.7%
3	Decision Tree	89.1%	76.1%
4	Random Forest	90.7%	77.0%
5	Neural Net	92.0%	84.2%
6	AdaBoost	90.7%	78.1%
7	Naive Bayes	74.5%	74.9%
8	QDA	75.3%	81.1%
9	LightGBM	88.9%	81.2%

Tablica 4.1: Stupac *names* jest naziv korištenog algoritma, stupac *accuracy* prikazuje preciznost modela, tj. broj dobro klasificiranih primjera podijeljen s brojem svih primjera u skupu za testiranje. Stupac ROC AUC jest AUC metrika koja se često koristi u binarnim kaslifikacijskim problemima. Ako skup podataka koje predviđamo nije ravnomjeran, tj. ako jedna klasa dominira, preciznost modela i F1 score nisu metrike koje najbolje opisuju performanse modela. Na primjer, ako u skupu imamo 99% klase 1, a samo 1% klase 0, model može reći da su svi primjeri klase 1 i imati će preciznost 99%. ROC AUC metrika nema taj problem, jer mjeri kako su primjeri raspoređeni s obzirom na vjerojatnost da primjer pripada jednoj, odnosno drugoj klasi. Zbog vremenske i memorijske ograničenosti nekih od algoritama, algoritmi su trenirani i testirani samo na podacima za vrijeme rješavanja zadatka. Broj ulaznih neurona je 76. Izvor: obrada autora

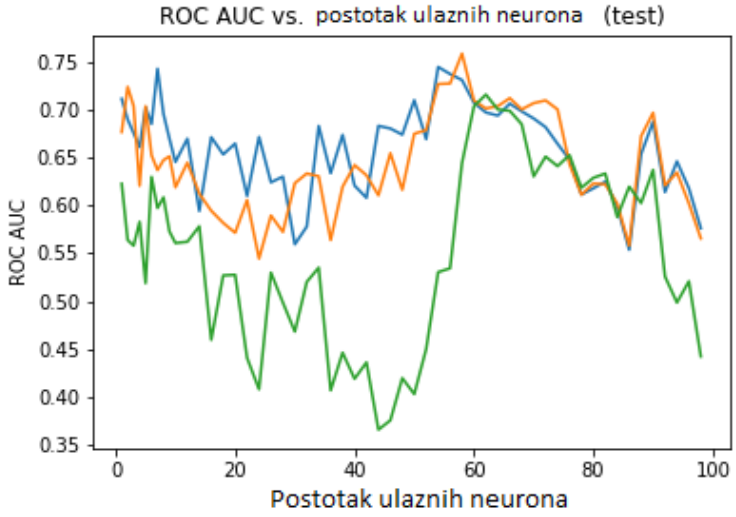
Kako bi odgovorili na pitanje kako točnost modela ovisi o broju neurona M koje koristimo kao ulaz za model, te o parametru lag koji opisuje koliko koraka u budućnost predviđamo stanje neurona, koristiti ćemo LightGBM algoritam [24, 25]. Osim što je algoritam koji konzistento daje najbolje rezultate na kaggle natjecanjima, on je algoritam koji se veoma brzo trenira. Puno brže od ostalih algoritama koji koriste stabla, poput slučajnih šuma, a pogotovo je znatno brži od neuronskih mreža. Iskoristiti ćemo brzinu treniranja algoritma za procjenu ovisnosti točnosti o parametrima M i lag. Za svaku vrijednost parametara moramo ponovno trenirati algoritam. Kako bi procijenili točnosti modela u ovisnosti o tome koliko koraka u budućnosti predviđamo stanje neurona, varijabla koju predviđamo je stanje najaktivnijeg ne-



Slika 4.1: Točnost modela za lag od 0 do 1500. Metrika kojom mjerimo točnost je ROC AUC. Plava linija prikazuje rezultat za 10% najaktivnijih neurona, narančasta za 4% najaktivnijih, na skupu podataka za testiranje. Primjetimo da je točnost modela koji ima više ulaznih varijabli uglavnom veća, osim na području $lag \approx 200$. Neočekivano je da točnost ne pada monotno s lag-om. Izvor: obrada autora

urona. Rezultati su prikazani na slici (4.1). Na slici (4.1), plava linija prikazuje rezultat ako za ulazne varijable koristimo preostalih 10% najaktivnijih neurona, a narančasta 4% najaktivnijih neurona. Vidimo da ako koristimo manji broj neurona kao ulazne varijable, točnost je uvijek nešto niža. Međutim, neočekivani rezultat je što točnost ne pada monotono u ovisnosti o lag-u, već oscilira, te ima područja u kojima je točnost jednaka onoj kada lag-a uopće nema. Na slici (4.2), vidimo kako točnost ovisi o postotku neurona, od ukupnog broja neurona, koje koristimo kao ulazne varijable za model. Plava linija pokazuje vrijednost ako je $lag = 1$, narančasta $lag = 10$, a zelena $lag = 100$. Vidimo da veći lag znači lošije rezultate. Očekivani rezultat bi bio da točnost raste s povećanjem broja ulaznih neurona, pogotovo zbog načina treniranje algoritama zasnovanih na stablima, međutim, više varijabli dovodi do pretreniranosti modela koja smanjuje njegovu točnost na skupu podataka za testiranje.

Veoma zanimljiv rezultat se javlja kada promatramo kako točnost modela za predviđanje aktivnosti najaktivnijeg neurona ovisi o parametru lag, ako ga treniramo i testiramo samo na mjerenjima za vrijeme rješavanja zadatka. Rezultat na skupu podataka za treniranje je prikazan na slici (4.3). Plava linija prikazuje rezultate ako za ulazne podatke koristimo 151 neurona (20% najaktivnijih), narančasta ako koristimo

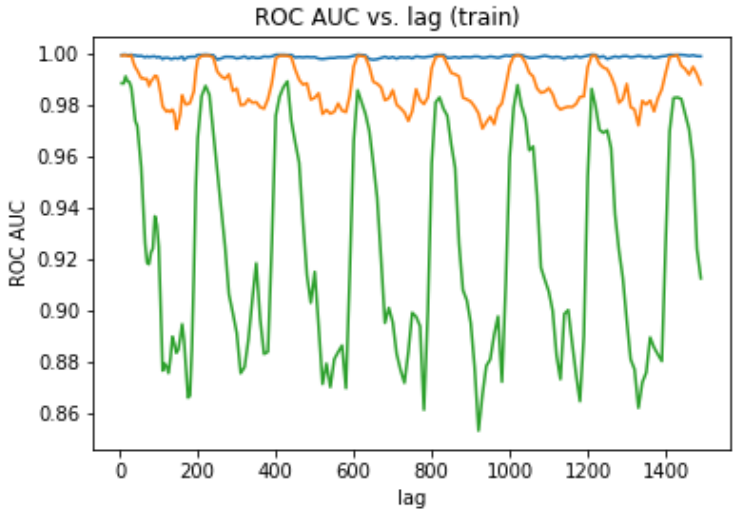


Slika 4.2: Točnost modela za postotak neurona koje koristimo kao ulazne varijable m od ukupnog broja neurona M. Plava linija prikazuje vrijednosti za lag jedank 1, narančasta 10, a zelena 100. Izvor: obrada autora

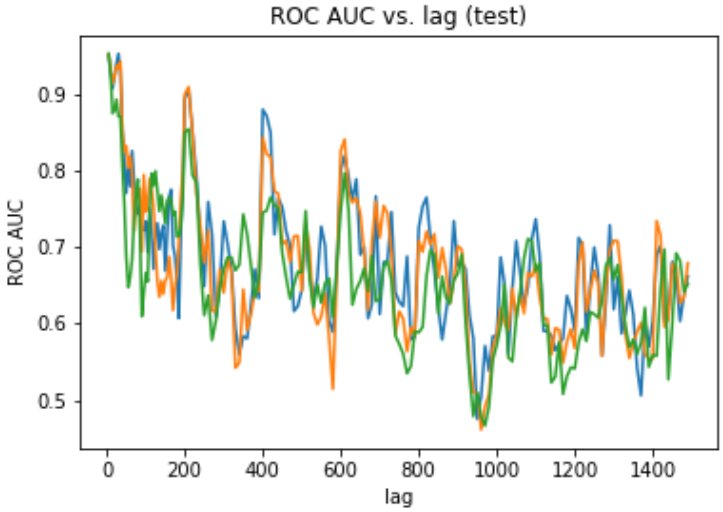
76 neurona (10% najaktivnijih), a zelena ako koristimo 30 neurona (30% najaktivnijih). Primijetimo da točnost modela na skupu podataka za trening praktički ne ovisi o tome koliko koraka u budućnosti predviđamo kada je broj ulaznih neurona velik. Smanjivanjem broja ulaznih neurona počinjemo primjećivati oscilacije u točnosti s parametrom lag. Kada koristimo samo 4% najaktivnijih neurona za predviđanje, oscilacije postaju velike, razlika između najveće i najmanje točnosti je 10%. Dakle, točnost modela je visoka za $lag = 1$, te se polako smanjuje kako se lag povećava. U trenutku kada lag pređe vrijednost 200, točnost modela na skupu za treniranje naglo poraste. Važno se prisjetiti da svaki zadatak traje 200 mjerenja, što je upravo period oscilacija točnosti. Točnost modela u ovom slučaju ne ovisi o lag-u, već samo koliko smo blizu "lokalno". Pod "lokalno" mislim koliko smo daleko unutar jednog zadatka od točke koju predviđamo. Preciznije, točnost je periodična s obzirom na duljinu trajanja mjerenja zadatka. Slika (4.4) pokazuje točnost za iste modela ali na skupu podataka za trening. Vidimo da se taj fenomen ne pojavljuje na skupu podataka za trening.

4.2.2 Predviđanje stanja jednog neurona - regresija

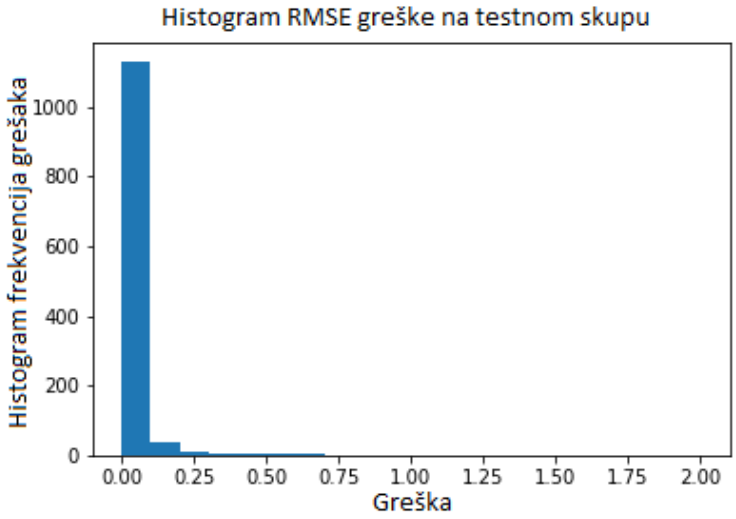
Slijedeće pitanje na koje ćemo pokušati odgovoriti jest koliko povijest svakog neurona utječe na njegovo stanje. Kako bi preciznije razmotrili problem, promatrati ćemo ga regresijski, odnosno, predviđamo koncentraciju kalcija u neuronu u trenutku t, ko-



Slika 4.3: Točnost modela za lag od 0 do 1500. Metrika kojom mjerimo točnost je ROC AUC. Plava linija prikazuje rezultat za 20% najaktivnijih neurona, narančasta za 10% najaktivnijih, a zelena za 4% najaktivnijih neurona, na skupu podataka za treniranje. Primjećujemo da kako smanjujemo broj neurona koje koristimo kao ulazne varijable u model, točnost modela počinje padati, te počinje oscilirati. Oscilacije su najveće za najmanji broj ulaznih neurona. Primijetimo da je period oscilacija jednak 200, što je upravo duljina trajanja jednog zadatka. Izvor: obrada autora



Slika 4.4: Točnost modela za lag od 0 do 1500. Metrika kojom mjerimo točnost je ROC AUC. Plava linija prikazuje rezultat za 20% najaktivnijih neurona, narančasta za 10% najaktivnijih, a zelena za 4% najaktivnijih neurona, na skupu podataka za testiranje. Primijetimo da fenomen koji postoji na skupu podataka za treniranje ne postoji u podacima za testiranje. Vidimo da točnost slabo ovisi o broju neurona koje koristimo. Također globalno točnost pada s lag-om, ali veoma oscilira. Izvor: obrada autora



Slika 4.5: Histogram grešaka. Primijetimo da je većina grešaka grupirana u području manjem od 0.25. Izvor: obrada autora

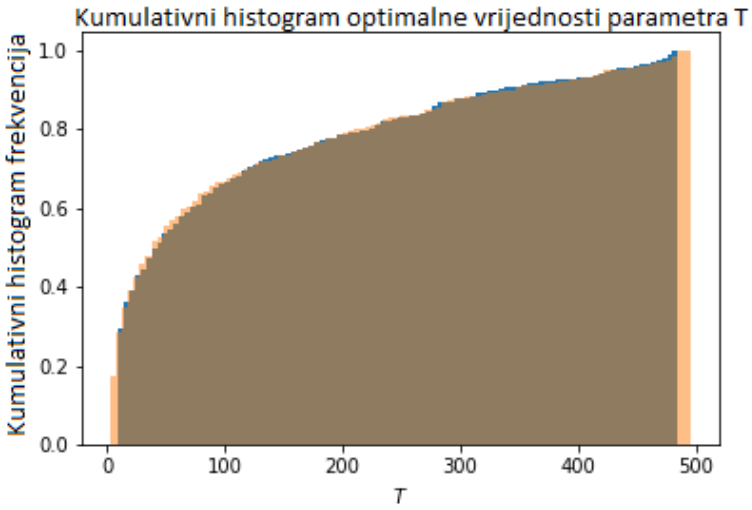
risteći koncentracije kalcija u trenucima $t-1, t-2, ..., t-T$. Naš model možemo opisati kao $\boldsymbol{F}(\sigma_k^{t-1}, \sigma_k^{t-2}, ..., \sigma_k^{t-T}) = \hat{\sigma}_k^t$. Funkcija greške i metrika točnosti koju promatramo je korijen srednje kvadratne greške [26]

$$\boldsymbol{L} = \sqrt{\frac{1}{N}\sum_i (\hat{y} - y)^2}. \tag{4.2}$$

Za svaki neuron treniramo model za svaku vrijednost parametra T od 0 do 500. Tražimo za koji T je greška najmanja. Na slici (4.5) vidimo histogram grešaka. Kako bi analiza bila robusnija na oscilacije, također tražimo optimalni T, ali tako da prvo zagladimo mjerenja točnosti modela s prozorom veličine 5. Usporedba optimalnog parametra T za zaglađene i nezaglađene podatke dana je na slici (4.6). Primijetimo da ne postoji značajna razlika između optimalnih parametara T u zaglađenom i nezaglađenom slučaju.

4.2.3 Predviđanje stanja mreže neurona

Posljednje pitanje vezano uz predikciju stanja neurona, a koje ćemo pokušati odgovoriti jest možemo li, i s kojom preciznošću, predvidjeti stanje cijele mreže neurona. Pitanje je možemo li znajući stanje $\boldsymbol{\sigma}^t$ predvidjeti stanje $\boldsymbol{\sigma}^{t+1}$. Algoritam koji koristimo su neuronske mreže jer nam omogućuju da direktno predviđamo stanje cijele mreže. Drugi algoritmi nisu dovoljno fleksibilni u izboru izlazne vrijednosti modela.



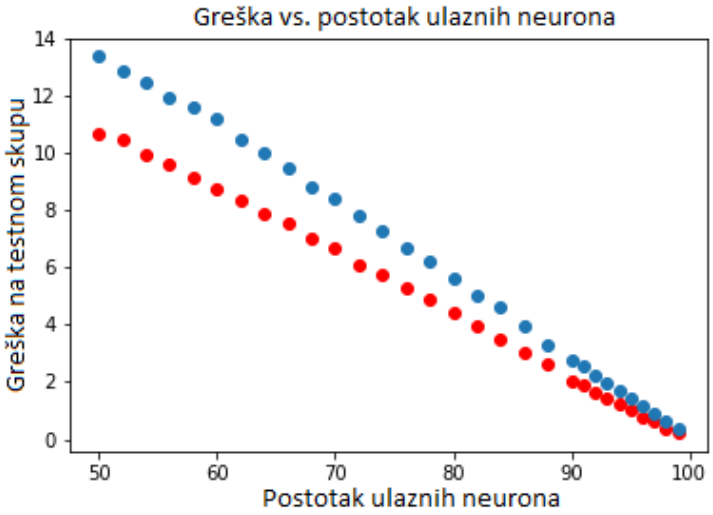
Slika 4.6: Kumulativni histogram optimalnog parametra T. Plavom bojom je označen histogram optimalnog T-a, a narančastom histogram optimalnog T-a, ali nakon što su rezultati zaglađeni prozorom veličine 5. Primijetimo da razlika između ova dva histograma zapravo ne postoji, tj. minimumi grešaka nisu samo slučajne oscilacije već su stvarno greške za optimalnu vrijednost parametra T. Vidimo da je za 40% neurona optimalni T manji od 50, a za njih 60% manji od 100. Za ostatak neurona, kumulativni histogram linearno raste od 100 do 500. Izvor: obrada autora

Kada bi koristili neki drugi algoritam bilo bi potrebno istrenirati poseban model za svaki neuron. Međutim, time pristup gubi smisao, jer ne predviđamo stanje cijele mreže već svakog neurona posebno. Moć neuronskih mreža je u tome da ih možemo optimizrati za proizvoljni zadatak.

Funkcija greške je binarna unakrsna entropija, koja je dana izrazom (4.1), za svaki neuron posebno. Kako ni jedna standardna metrika greške nije primjenjiva na ovaj problem, uvodimo vlastitu metriku greške:

$$\boldsymbol{L} = \frac{1}{N} \sum_i |y_i \oplus \hat{y}_i|, \tag{4.3}$$

gdje je $\oplus$ binarni logički operator ekskluzivno ili. Da pojasnim, ako je predikcija $\hat{y}_i$ a prava vrijednost y_i, prvo napravimo operator ekskluzivno ili između njih dvoje. Ova operacija nam daje informaciju na koliko se mjesta predikcija i stvarna vrijednost razlikuju. Zatim samo izbrojimo koliko ima neurona na kojima se predikcija i stvarna vrijednost razlikuju. Ovako definirana greška nam daje koliko smo neurona u prosjeku krivo procijenili. Na slici (4.7) vidimo kako ta greška ovisi o postotku neurona koje koristimo kao ulazne varijable u model. Plave točke su rezultati za pa-



Slika 4.7: Greška modela za predviđanje stanja cijele neuronske mreže u ovisnosti o veličine mreže koju predviđamo. Plave točke prikazuje predviđanje od 100 koraka unaprijed, a crvene za predviđanje od 10 koraka unaprijed. Zanimljivo je primjetiti savršenu linearnost između greške i veličine sustava. Izvor: obrada autora

rametar $lag = 100$, a crvene točke za $lag = 10$. Vidimo da što više koraka predviđamo u budućnost da je točnost modela manja. Također je zanimljivo primjetiti linearnu ovisnost između točnosti modela i broja neurona koje koristimo kao ulazne varijable. Na slici (4.8) vidimo točnost modela podijeljena s brojem neurona koje predviđamo (i uzimamo kao ulaz u model).

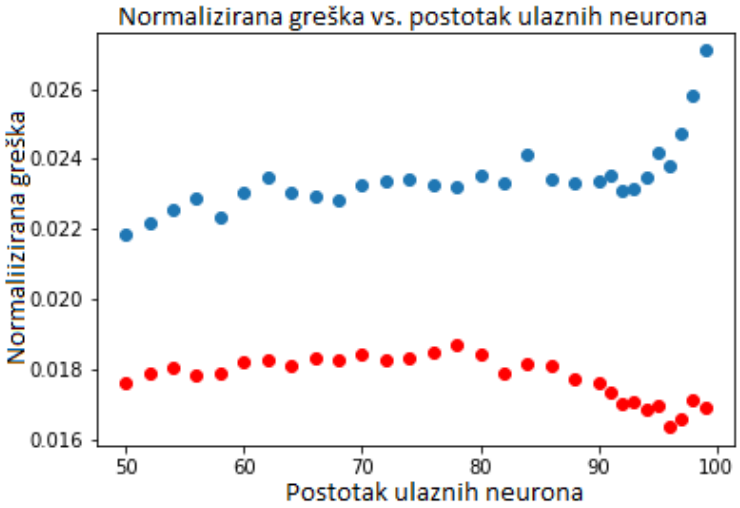
Na slici (4.9) vidimo kako točnost modela ovisi o parametru lag, veličina mreže koju predviđamo je 37. Primjetimo da mreža brzo dođe do greške od jednog neurona i tu zapravo ostaje. Za male lag-ove, mreža ima srednju grešku manju od pola neurona.
Korištena arhitektura jest

$$m - \text{Dense}(3 \times m) - \text{Dense}(m), \tag{4.4}$$

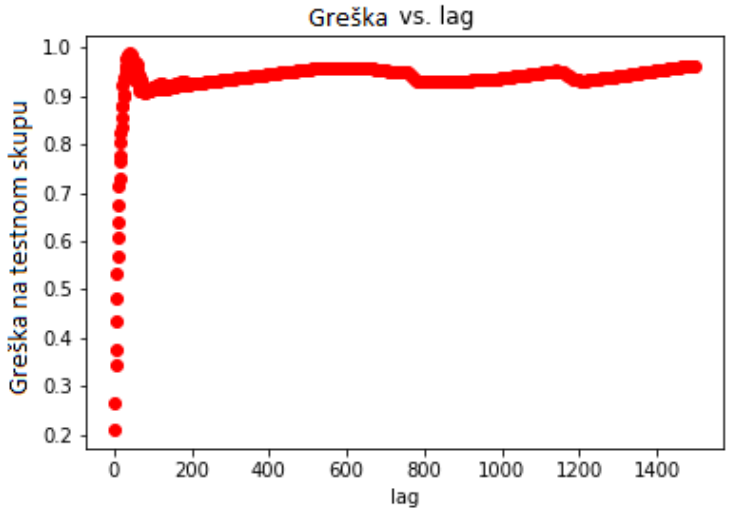
gdje je m veličina mreže koju predviđamo.

4.3 Zadatak ili nagrada

Zanima nas možemo li razlikovati stanja mreže neurona dok miš riješava zadatak od stanja mreže neurona dok miš prima nagradu. Krećemo od pretpostavke da ne



Slika 4.8: Greška modela za predviđanje stanja cijele neuronske mreže u ovisnosti o veličine mreže koju predviđamo. Greška je normalizirana s obzirom na veličinu mreže čije stanje predviđamo. Plave točke prikazuje predviđanje od 100 koraka unaprijed, a crvene za predviđanje od 10 koraka unaprijed. Možemo primjetiti da normalizirana greška zapravo ne ovisi o veličini sustava. Izvor: obrada autora



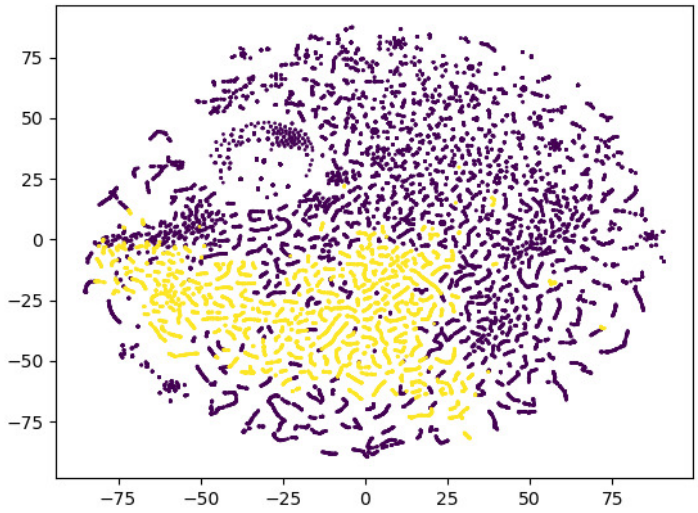
Slika 4.9: Greška modela za predviđanje stanja cijele neuronske mreže u ovisnosti o veličine mreže koju predviđamo. Greška brzo dođe do vrijednosti od 1 neurona te ne ide preko te vrijednosti. Izvor: obrada autora

znamo ništa, da nemamo labelirane podatke na zadatke i nagradu. Želimo pomoću nenadziranog strojnog učenja klasterirati podatke, te vidjeti koliko klastera možemo pronaći.

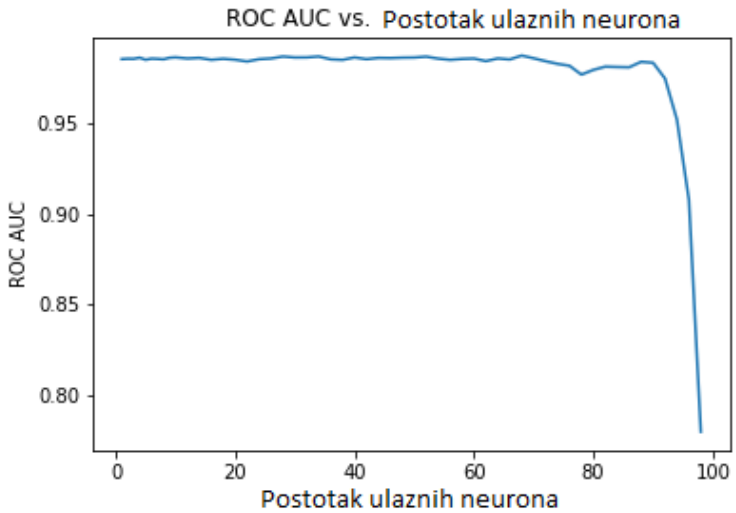
Koristimo sve neurone mreže kao vektor stanja koji želimo klasterirati. Kako je taj vektor visokodimenzionalan (1201) koristimi autoenkoder [27] kako bi smanjili dimenzionalnost vektora. Autoenkoder je posebna vrsta neuronske mreže koja se koristi za redukciju dimenzionalnosti problema. Autoenkoder se sastoji od dva dijela: enkodera i dekodera. Enkoder mapira visokodimenzionalni vektor u nižedimenzionalan prostor, a enkoder iz nižedimenzionalnog vektora pokušava rekonstruirati početni vektor. Srednji sloj autoenkodera se koristi kao "šifra" početnog vektora. Autoenkoder se trenira tako da je ulaz i izlaz modela isti vektor. Funkcija greške je binarna unakrsna entropija. Google koristi autoenkoder za svoju tražilicu slika. Nakon što za svako stanje pronađemo nižedimenzionalnu "šifru", koristimo algoritam TSNE [28], kako bi taj prostor mapirali u dvodimenzionalni prostor koji nam je jednostavno vizualizirati. Srednji sloj u korištenom autoenkoderu je bio veličine 100. Rezultati klasteriranja su prikazani na slici (4.10). Žutom bojom su označena mjerenja dok miš riješava labirint, a ljubičastom dok prima nagradu. Možemo primjetiti da su se stanja zadatka odvojila od stanja nagrade. Možemo zaključiti da su stanja neurona miša dok riješava zadatak i prima nagradu dovoljno različita da ih možemo raspoznavati.

Sada ćemo pokušati istrenirati model za klasifikaciju stanja na zadatak i nagradu. Ovaj problem je u svojoj suštini isti kao i predviđanje stanja jednog neurona. Opet imamo binarnu klasifikaciju, osim što sad ne klasificiramo mjerenja na aktivan/neaktivan već na zadatak/nagrada. Funkcija greške je binarna unakrsna entropija, a metrika greške je ROC AUC mjera. Opet ćemo koristiti LightGBM algoritam. Model predviđa vjerojatnost da se miš nalazi u stanju nagrade, tj. $\boldsymbol{F}(\boldsymbol{X}) = P(\boldsymbol{y}|\boldsymbol{X})$, gdje je sad $\boldsymbol{y}$ =stanje nagrade. Rezultati u ovisnosti o broju ulaznih neurona su prikazani na slici (4.11). Vidimo da se točnost modela brzo približi asimptotskoj vrijednosti od 99%, ali ju nikad ne dostigne. Također je zanimljivo da i s malim brojem ulaznih varijabli točnost modela ne ide ispod 80% ROC AUC-a.

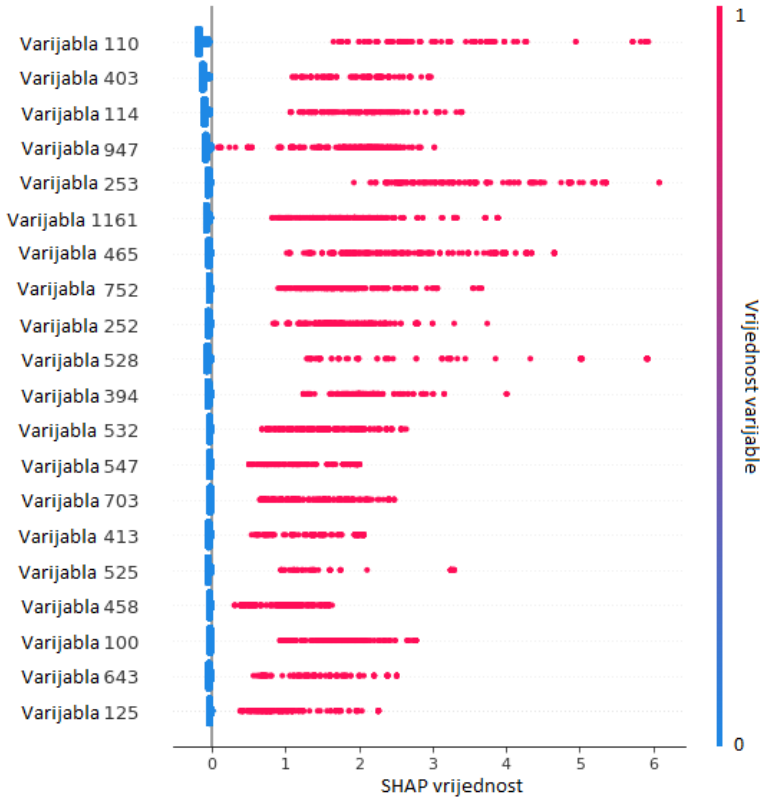
Slijedeće pitanje koje možemo postaviti je koji neuroni najviše utječu, te kako utječu na predikciju stanja. Kako bi odgovorili na ovo pitanje koristimo SHAP [29] vrijednosti. Na slici (4.12) vidimo SHAP vrijednosti modela koji koristi sve neurone kao



Slika 4.10: Vizualizacija klasteriranja pomoću autoenkodera i TSNE algortima. Žutom bojom su označena područja koja reprezentiraju zadatak, a ljubičastom nagradu. Vidimo da možemo odvojiti stanja nagrade i zadatka, iako granica možda nije strogo definirana. Međutim, algoritmi nenadziranog učenja uvijek imaju nešto lošije rezultate, te je već ovakav rezultat dovoljan da možemo biti sigurni da će algortimi nadziranog učenja biti primjenjivi. Izvor: obrada autora



Slika 4.11: ROC AUC mjera za klasifikaciju stanja na zadatak i nagradu u ovisnosti o broju ulaznih neurona u model. Vidimo da što više neurona koristimo kao varijable modela, to su rezulttati modela bolji što je i očekivano. Također je važno primjetiti da točnost modela ne ide ispod 80%. Izvor: obrada autora



Slika 4.12: SHAP graf za 20 najvažnijih varijabli. Na y-osi se nalazi redni broj neurona, a na x-osi su SHAP vrijednosti. Crvenom su obojana stanja kada je taj neuron aktivan, a plavom kada je neaktivn. Primjetimo da za najvažnijih 20 varijabli, njihova aktivnost implicira da se miš nalazi u stanju nagrade. Izvor: obrada autora

ulazne varijable. SHAP vrijednosti su izračunate na skupu podataka za testiranje. Na y-osi se nalazi redni broj neurona. Neuroni su predani po svojoj važnosti, tj. apsolutnoj sumu SHAP vrijednosti. Na x-osi su SHAP vrijednosti. Crvenom bojom su označene točke u kojima je taj neuron aktivan (1), a plavom one u kojima je neuron neaktivan (0). Vidimo da za 20 najaktivnijih neurona, njihova aktivnost upućuje da miš prima nagradu. Ovo je bitno drugačija informacija od standardnih mjerenja mozga. Informaciju koju tu možemo saznati jest koji su neuroni aktivni za vrijeme određene radnje. Pomoću SHAP vrijednosti možemo reći ako je neuron 110 aktivan, onda se miš nalazi u stanju nagrade. Iz aktivnosti neurona možemo reći što miš radi.

5 Zaključak

Cilj rada je bio istražiti, te opisati izmjerena stanja mreže neurona koji riješava zadatke, te potom dobije nagradu.

Isingov model je fizikalni model koji se koristi kako bi se opisala dinamika spinskog lanca. Izmjerena stanja smo modelirali pomoću spinskog lanca i Isingovog modela. Ovako definiran problem spada u skupinu tzv. inverznih problema. Inverzni problemi su oni u kojima postoje mjerenja, te je potrebno iz tih mjerenja pronaći parametre modela. Pomoću inverznog Isingovog modela smo uspijeli zaključiti kako izgleda struktura mreže, međutim ograničeni smo veličinom sustava kojeg želimo analizirati. Egzaktna metoda je jedino primjenjiva na male sustave (broj ulaznih neurona $M \leqslant 20$). Moguće je koristiti aproksimativne metode kako bi se riješavali sustavi veće dimenzionalnosti. Koristeći dobivene parametre modela, moguće je pronaći vjerojatnost svakog pojedinog stanja, te generirati nova slučajna stanja po izračunatim vjerojatnostima.

Koristeći metode strojnog učenja uspijeli smo odgovoriti na specifična pitanja, poput: kako će izgledati mreža neurona u idućem trenutku. Najvažnija informacija koju smo saznali koristeći standardne fizikalne metode jest biranje slučajnih stanja neurona. Međutim, stanja su birana pod pretpostavkom da ne postoji vremenska ovisnost, što znamo da nije valjana pretpostavka. Metode strojnog učenja nam daju odgovore na specificirana, precizna pitanja, koja su mnogo prirodnija za dane podatke. Kako izgleda mozak, tj. mreža neurona, za vrijeme određene radnje je nešto što se neurobiolozi pitaju odavno. Koristeći strojno učenje, ne samo da možemo odgovoriti na to, već smo otišli korak dalje, te možemo tvrditi da ako je ova skupina neurona aktivna, miš vrši ovu određenu radnju.

Osim što možemo odgovoriti na razna pitanja, također možemo, što je nekad i važnije, precizno kvantificirati grešku modela. Jednostavno je definirati metriku koja ocjenjuje performanse modela. Tako smo u problemu s predviđanjem slijedećeg stanja neuronske mreže koristili sumu krivo klasificiranih primjera. Vidjeli smo da je za manje mreže ($M \approx 40$), greška modela manja od četvrtine neurona po primjeru. Ovako rigorozno testiranje modela nije moguće ako koristimo standardne metode.

Osim pronaženja odgovora na pitanja i računanje njegove greške, primjena strojnog učenja nam dopušta da s lakoćom možemo testirati kako preddefinirana točnost ovisi

o raznim parametrima. Treba spomenuti da se dani podatci ponašaju očekivano, tj. što više koraka u budućnost predviđamo, to je prediktivna moć modela manja. Također, što je broj ulaznih varijabi veći, to modelu raste točnost.

Naravno, prostor svih mogućih pitanja na koja bi strojno učenje moglo dati odgovor, u ovom radu nije u potpunosti istražen. Prvo od tih pitanja koje možemo postaviti jest možemo li generirati nova stanja neuronske mreže ali takva da oni poštuju empirijsku distribuciju vjerojatnosti stanja. Možemo li napraviti model koji bi u porpunosti zamijenio biranje slučajnih stanja preko klasičnih fizičarskih metoda, te kako taj model ovisi o veličini sustava. Polazna točka takvog modela bi bili sve popularniji GAN-ovi (eng. Generative Adversarial Networks).

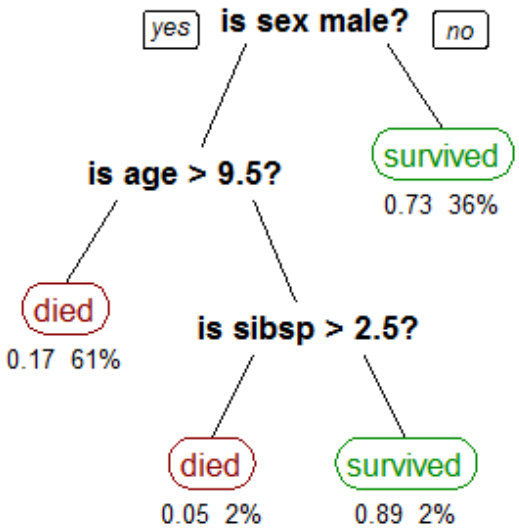
Dodaci

Dodatak A Algoritmi i principi strojnog učenja

A.1 Stablo odluke, Slučajna šuma i LightGBM

A.1.1 Stablo odluke

Učenje stabla odluke je uobičajena metoda u rudarenju podataka [30]. Cilj je stvoriti model koji predviđa vrijednost ciljane oznake y na temelju nekoliko ulaznih varijabli. Primjer je prikazan na slici (A.1). Svaki unutarnji čvor odgovara jednoj od ulaznih varijabli; za svaku vrijednost ulazne varijable postoji brid koji vodi do čvorova djece. Svaki list predstavlja vrijednost ciljne varijable s obzirom na vrijednosti ulaznih varijabli koje su predstavljene putem od korijena do lista.



Slika A.1: Stablo koje pokazuje opstanak putnika na Titanicu (”sibsp” je broj supružnika ili braće i sestara). Brojke ispod lišća pokazuju vjerojatnost preživljavanja i postotak primjera za taj list. Ukratko: Vaše šanse za preživljavanje bile su dobre ako biste bili (i) žena ili (ii) muškarac mlađi od 9,5 godina s manje od 2,5 braće i sestara. Slika preuzeta s `https://en.wikipedia.org/wiki/Decision_tree_learning`

Stablo odluke je jednostavna reprezentacija za primjere klasificiranja. Pretpostavimo da sve ulazne varijable imaju konačne diskretne domene, te postoji jedna ciljna

značajka pod nazivom ”klasifikacija”. Svaki element domene klasifikacije naziva se klasa. Stablo odluke ili stablo klasifikacije je stablo u kojem je svaki unutarnji (ne-list) čvor označen funkcijom unosa. Svaki list stabla obilježen je klasom ili raspodjelom vjerojatnosti po klasama.

Pri učenju stabla odluke koristi se funkcija nagrade, najčešće je to entropija, tj. smanjenje entropije. Stablo se gradi tako da za svaku varijablu prolazimo po svim mogućim vrijednostima te tražimo graničnu vrijednost v, takvu da je suma entropije u odvojenim skupovima (skup 1 $< v$, skup 2 $> v$) minimalna i manja od početne entropije. Ovaj postupak se primjenjuje za svaki čvor posebno. Ako gradimo novi čvor, za koji postoje roditeljski čvorovi, onda u traženje granične vrijednosti v i računanje entropije, ulazi samo podskup podataka koji zadovoljava uvjete roditeljskih čvorova.

A.1.2 Slučajna šuma

Slučajna šuma se sastoji od mnogo stabla odluke [31]. Kako bi pronašli vrijednost modela slučajne šume za neki ulazni vektor, trebmo pronaći vrijednost svakog stabla odluke za taj ulazni vektor. Zatim je rezultat slučajne šume, srednja vrijednost svih stabla odluke, ako je problem regresijski, odnosno, klasa s najviše glasova (stabala odluke) za klasifikacijske probleme.

Svako stablo se gradi na sljedeći način:

1. Ako je broj primjera u skupu za treniranje N, odaberimo N primjera, ali tako da se primjeri mogu ponavljati. Ovaj skup koristimo za treniranje jednog stabla.

2. Ako posotoji M ulaznih varijabli, odaberimo broj m takav da $m < M$. Biramo najbolju graničnu vrijednost v od m slučajnih varijabli za svaki od čvorova u stablu. Vrijednost m je konstantna za vrijeme treniranje modela

3. Svako stablo je izgrađeno do kraja, ne primjenjuje se skraćivanje (eng. pruning, metoda smanjivanja prenaučenosti stabla odluke).

Slučajna šuma pripada algoritmima koji se nazivaju ansambli. Ovaj naziv dolazi od činjenice da su ansambli samo više jednostavnih modela koji rade skupa.

A.1.3 LightGBM

LightGBM je jedna implementacija boosting algoritma primjenjenog na stabla odluke [24, 25]. Boosting je meta-algoritam strojnog učenja ansamblova, koji prvenstveno smanjuje pristranost, ali i varijancu. Boosting pripada algoritmima nadziranog učenja koji pretvaraju slabe modele u jake. Slab model se definira kao model koji je neznatno koreliran s ciljanom oznakom (može označiti primjere bolje od slučajnog nagađanja). Nasuprot tome, jak model je onaj koji je proizvoljno dobro koreliran s ciljanom oznakom. Boosting se temelji na pitanju koje su postavili Kearns i Valiant (1988, 1989): ”Može li niz slabih modela stvoriti jedan jak model?”.
Iako boosting nije algoritamski ograničen, većina boosting algoritama se sastoji od iterativnog učenja slabih modela koji se dodaju u konačni jaki model. Kada se dodaju, obično se ponderiraju na neki način koji je povezan s preciznošću slabog modela. Nakon što je slabi model dodan, težina ciljanih oznaka se prilagođavaju. Primjerima koji imaju veću grešku je pridodana veća težina, onima koji su točno kalsificrani je smanjena. Dakle, budući slabi modeli više se usredotočuju na primjere koji su prethodni slabi modeli pogrešno klasificirali.

A.2 Neuronske mreže

Umjetne neuronske mreže (eng. Artificial neural networks ili ANN) ili spojni sustavi, računalni su sustavi inspirirani biološkim neuronskim mrežama koje čine životinjski mozak [32]. Neuronska mreža sama po sebi nije algoritam, nego okvir za razne algoritme strojnog učenja koji rade zajedno i obrađuju složene podatke. Takvi sustavi ”uče” obavljanje zadataka uzimajući u obzir primjere, obično bez programiranja bilo kakvog pravila zadatka. Na primjer, u prepoznavanju slike mogu naučiti identificirati slike koje sadrže automobile analiziranjem primjera slika koje su ručno označene kao ”automobil” ili ”ne automobil”, a pomoću rezultata identificiraju automobile na drugim slikama. Oni to rade bez ikakvog prethodnog znanja o automobilima, npr. Da imaju četiri kotača, prozore, motor. Umjesto toga, automatski stvaraju identifikacijske osobine iz materijala za učenje koje obrađuju.
ANN se temelji na zbirci povezanih jedinica ili čvorova nazvanih umjetnim neuronima koji slabo modeliraju neurone u biološkom mozgu. Svaka veza, poput sinapsa u biološkom mozgu, može prenijeti signal iz jednog umjetnog neurona u drugi. Umjetni

neuron koji prima signal može ga obraditi i potom signalizirati dodatne umjetne neurone povezane s njim.
U standardnim ANN izvedbama signal pri povezivanju umjetnih neurona je realni broj, a izlaz svakog umjetnog neurona izračunat je nekom nelinearnom funkcijom zbroja njegovih ulaza. Veze između umjetnih neurona nazivaju se 'rubovi'. Umjetni neuroni i rubovi obično imaju težinu koja se prilagođava kroz proces učenja. Težina povećava ili smanjuje snagu signala na spoju. Umjetni neuroni mogu imati prag tako da se signal šalje samo ako agregirani signal prijeđe taj prag. Tipično, umjetni neuroni se skupljaju u slojeve. Različiti slojevi mogu izvoditi različite vrste transformacija na svojim ulazima. Signali kreću od prvog sloja (ulazni sloj), do posljednjeg sloja (izlazni sloj), eventualno nakon višestrukog prelaska slojeva.
Izvorni cilj ANN-a bio je rješavanje problema na isti način kao i ljudski mozak. Međutim, tijekom vremena se pozornost posvetila obavljanju određenih zadataka, što je dovelo do odstupanja od biologije. Umjetne neuronske mreže korištene su na različitim područjima, uključujući računalni vid, prepoznavanje govora, strojno prevođenje, filtriranje društvenih mreža, igranje društvenih igara i video igara, te medicinsku dijagnostiku.

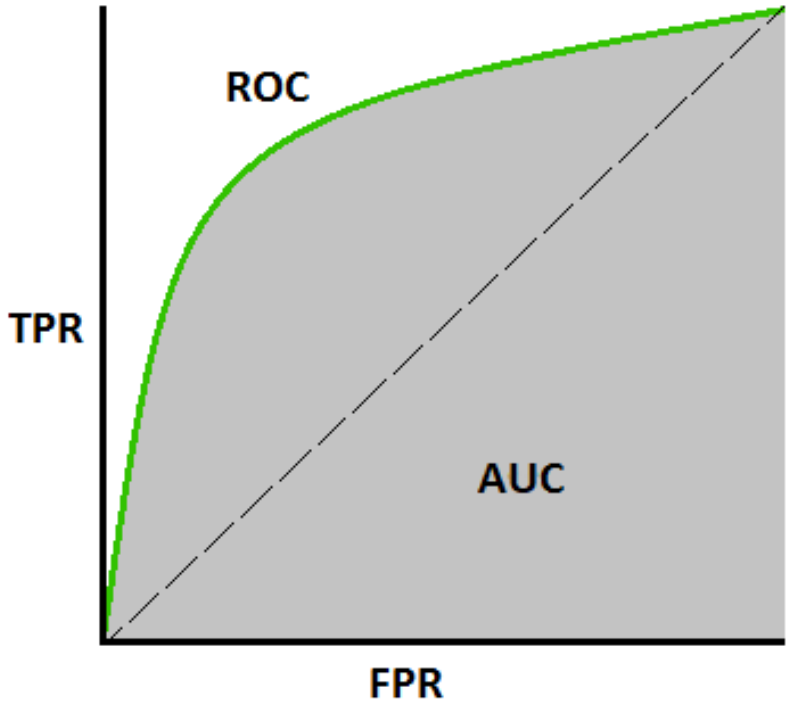
Neuronske mreže se treniraju koristeći takozvani algoritam propagiranja unatrag. Proces treniranja neuronske mreže se sastoji od 3 najvažnija koraka:

- Propagiranje unaprijed - Vektor stanja se koristi kao ulaz u neuronsku mrežu. Zatim se računa vrijednost u svakom neuronu u prvom sloju, te se pripadni signali šalju u neurone u drugom sloju. Ovaj proces se nastavlja do kraja neuronske mreže.

- Računanje greške - Koristeći funkciju greške izračunamo pogrešku za određeni primjer. Često se više primjera odjednom propagira, te se greška usrednji.

- Propagiranje unatrag - Izračunata greška se propagira unatrag te se pripadne težine veza mijenjaju s obzirom na grešku.

Ovaj proces se nastavlja sve dok se vrijednost funkcije greške na svim primjerima ne prestane smanjivati.



Slika A.2: Slika prikazuje ROC i AUC. Na x-osi je broj krivo klasificiranih jedinica, a na y-osi je broj točno klasificiranih jedinica. Slika preuzeta s `https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5`

A.3 AUC-ROC metrika

Krivulja AUC-ROC je metrika mjerenje uspješnosti klasifikacijskih modela na različitim pragovima [33]. ROC (eng. receiver operating characteristic) je krivulja vjerojatnosti, a AUC (eng. area under curve) predstavlja stupanj ili mjeru razdvojenosti. AUC govori koliko model može razlikovati klase. Što je AUC veći, model bolje predviđa nule kao nule i jedinice kao jedinice. Analogno, veći AUC znači da model bolje razlikuje između bolesnika s bolešću i bez bolesti. Na slici (A.2) vidimo definiciju AUC-ROC metrike. Na x-osi je broj krivo klasificiranih jedinica, a na y-osi je broj točno klasificiranih jedinica. AUC daje agregiranu mjeru uspješnosti za sve moguće pragove klasifikacije. Jedan od načina tumačenja AUC je kao vjerojatnost da model rangira slučajni pozitivan primjer (1) više nego slučajni negativni primjer (0). AUC je standardna mjera koja se koristi za ocjenu modela binarne klasifikacije u strojnom učenju.

A.4 Optimizacija hiperparametara

U strojnom učenju, optimizacija ili podešavanje hiperparametara je problem odabira skupa optimalnih hiperparametara za algoritam [34].

Ista vrsta modela strojnog učenja može zahtijevati različita ograničenja, težine ili stope učenja za generaliziranje različitih obrazaca podataka. Te se mjere nazivaju hiperparametri, i moraju biti podešene tako da model može optimalno riješiti problem strojnog učenja. Optimizacija hiperparametra pronalazi niz hiperparametara koji daje optimalni model koji minimalizira unaprijed definiranu funkciju greške na danim neovisnim podacima. Funkcija cilja uzima niz hiperparametara i vraća pripadnu grešku. Unakrsna validacija se često koristi za procjenu performansi modela. Važno je napomenuti da funkcija greške i funkcija cilja ne moraju biti iste. Funkcija cilja ne mora zadovoljavati razne uvjete koje funkcija greške mora, poput diferencijabilnosti. Funkcija cilja može biti proizvoljno komplicirana.

Dvije najčešće vrste optimizacije hiperparametara su:

1. Pretraživanje po mreži - Definira se mreža hiperparametara, te za svaki element mreže promatramo funkciju cilja.

2. Slučajno pretraživanje - Definiraju se rubovi prostora hiperparametara. Zatim odaberemo jedan slučajni element prostora hiperparametara te izračunamo pripadnu funkciju cilja. Ovaj postupak se ponovi proizvoljan broj puta te se zapamte optimalni parametri. Ovakav način optimizacije hiperparametara je u većini slučajeva bolji i brži od traženja hiperparametara po mreži.

A.5 SHAP

SHAP (Shapley Additive exPlanations) jedinstven je pristup objašnjavanju izlaza svakog modela strojnog učenja. SHAP povezuje teoriju igre s lokalnim objašnjenjima, kombinirajući nekoliko prethodnih metoda i predstavlja jedinu moguću dosljednu i lokalno preciznu metodu aditivnog pridodavanja varijabli koja se temlji na očekivanjima. Tehnika SHAP vrijednosti predložena je u nedavnim radovima Scott M. Lundberg sa Sveučilišta u Washingtonu [29]. Temelji se na Shapleyovim vrijednostima, tehnikama koje se koriste u teoriji igre kako bi se utvrdilo koliko je svaki suigrač u igri pridonio

uspjehu. U našem slučaju, svaka SHAP vrijednost mjeri koliko svaka varijabla našeg modela doprinosi, pozitivno ili negativno, na očekivani rezultat. To je slična ideja kao kad promatramo važnost varijabli u logističkoj regresiji, gdje možemo utvrditi utjecaj svake varijable gledanjem na veličinu pripadajućeg koeficijenta. Međutim, SHAP vrijednosti nude dvije važne prednosti. Prvo, SHAP vrijednosti mogu se izračunati za bilo koji model, pa nismo ograničeni na jednostavne, linearne - i stoga manje točne - logističke regresijske modele, već možemo izgraditi složene, nelinearne i točnije modele. Drugo, svaki pojedini primjer ima svoj vlastiti skup SHAP vrijednosti. Tradicionalni algoritmi važnosti značajki će nam reći koje su značajke najvažnije u čitavoj populaciji, ali to ne mora biti isto za svakog pojedinog korisnika. Varijabla koja je ključna za jednog klijenta može biti nebitna za drugog. Promatrajući samo globalne trendove, te se pojedinačne varijacije mogu izgubiti.

Dodatak B Izvorni kod

Inicijalizacija podataka

```
def init_ising(signal_threshold=3.5,remove_non_active_neurons=True):
    trace= np.genfromtxt('trace.dat', delimiter=',')
    todayTreadmillLog_inds_run=[22,223,629,830,1639,1839,2358,2558,3052,3253,3857,
    4057,4590,4791,5331,5532,6007,6207,6646,6846,7404,7604,8123,8323,8813,9013,
    9549,9750,10285,10485,11175,11375,12019,12219,12909,13109,13788,13988,14552,
    14752,15432,15632,16317,16518,17601,17801,18414,18614,19881,20082,20620,20820,
    21755,21955,22888,23088,24099,24299,25229,25429,25719,25857,26201,26401,27184,
    27384,28265,28465,29394,29594,30026,30226]
    todayTreadmillLog_inds_prize=np.append(np.asarray([0]),
                    todayTreadmillLog_inds_run)
    todayTreadmillLog_inds_prize=np.append(todayTreadmillLog_inds_prize,
                    np.asarray([30691]))
    todayTreadmillLog_inds_run=np.reshape(todayTreadmillLog_inds_run,(-1,2))
    todayTreadmillLog_inds_prize=np.reshape(todayTreadmillLog_inds_prize,(-1,2))

    index_run=np.empty(0,dtype=int)
    for i in range(0,todayTreadmillLog_inds_run.shape[0]):
        index_run=np.append(index_run,np.asarray(range(
                todayTreadmillLog_inds_run[i,
                0],todayTreadmillLog_inds_run[i,1]+1)))

    index_prize=np.array(range(0,30691))
    index_prize=np.array(list(set(index_prize)-set(index_run)))

    trace_treadmill_run=np.copy(trace)
    for element in todayTreadmillLog_inds_run:
            standDev_run=trace_treadmill_run[:,element[0]:(element[1]+1)].std()
            trace_treadmill_run[:,element[0]:(element[1]+1)]
                    [trace_treadmill_run[:,
                    element[0]:(element[1]+1)]<3.5*standDev_run]=0
            trace_treadmill_run[:,element[0]:(element[1]+1)][
                    trace_treadmill_run[:,
```

```
                    element[0]:(element[1]+1)]>=3.5*standDev_run]=1

    trace_treadmill_prize=np.copy(trace)
    for element in todayTreadmillLog_inds_prize:
            standDev_prize=trace_treadmill_prize[:
                    ,element[0]:(element[1]+1)].std()
            trace_treadmill_prize[:,element[0]:(element[1]+1)][
                    trace_treadmill_prize[:,
                    element[0]:(element[1]+1)]<3.5*standDev_prize]=0
            trace_treadmill_prize[:,element[0]:(element[1]+1)][
                    trace_treadmill_prize[:,
                    element[0]:(element[1]+1)]>=3.5*standDev_prize]=1

    neural_data_run=trace_treadmill_run[:,index_run].copy()
    neural_data_prize=trace_treadmill_prize[:,index_prize].copy()
    if remove_non_active_neurons:
        neural_data_prize=neural_data_prize[
                (~np.all(neural_data_run==0,axis=1)),:]
        neural_data_run=neural_data_run[
                (~np.all(neural_data_run==0,axis=1)),:]

    return neural_data_run, neural_data_prize

def init(signal_threshold=3.5, remove_non_active_neurons=False):
    trace = np.genfromtxt('trace.dat', delimiter=',')
    st_dev=trace.std()
    trace[trace<signal_threshold*st_dev]=0
    trace[trace >= signal_threshold*st_dev] = 1
    if remove_non_active_neurons:
        trace=trace[~np.all(trace == 0, axis=1)]
    return trace

def init_continuous():
    trace= np.genfromtxt('trace.dat', delimiter=',')
    return trace
```

```
def substrings(n, x):
    return np.fromfunction(lambda i, j: x[i + j], (len(x) - n + 1, n), dtype=int)
```

Pomoćne funkcije i funkcije riješenja inverznog Isingovog problema

```
def get_percentile_of_active_neurons(neural_data,percentile=50):
    activity=neural_data.mean(axis=1)
    neural_data=neural_data[activity>np.percentile(activity,percentile),:]
    return neural_data

def xor_sum(y_predict,y_true,threshold=0.5):
    y_predict[y_predict>=threshold]=1
    y_predict[y_predict < threshold] = 0
    result= y_predict+y_true
    result[result == 2] = 0
    return result.sum(axis=1).mean()

def get_magnetisation_and_inverse_matrix_of_connected_correlations(neural_data):
    weights=np.ones(len(neural_data))/len(neural_data)
    connected_correlations=(neural_data.T.dot(neural_data*weights[:,None]))
    magnetisation=(neural_data*weights[:,None]).sum(0)
    connected_correlations=connected_correlations-
            np.outer(magnetisation,magnetisation)
    return magnetisation,np.linalg.inv(connected_correlations)


def mean_field_approx(neural_data):
    magnetisation,inverse_connected_correlations=
            get_magnetisation_and_inverse_matrix_of_connected_correlations(
                    neural_data)
    J=-inverse_connected_correlations
    np.fill_diagonal(J,0)
    h=J.dot(magnetisation)+np.arctanh(magnetisation)
    return h,J

def tap_reconstruction(neural_data):
```

```
    magnetisation,inverse_connected_correlations=
            get_magnetisation_and_inverse_matrix_of_connected_correlations(
                    neural_data)
    J=(-2*inverse_connected_correlations)/(1+np.sqrt(
            1-8*inverse_connected_correlations*np.outer(
                    magnetisation,magnetisation)))
    h=np.arctanh(magnetisation) + J.dot(magnetisation)+
            magnetisation*(J*J).dot(1-magnetisation*magnetisation)
    return h,J

def bethe_peierls_ansatz(neural_data):
    magnetisation,inverse_connected_correlations=
            get_magnetisation_and_inverse_matrix_of_connected_correlations(
                    neural_data)
    J=-0.5*np.arcsinh(2*inverse_connected_correlations)
    return J

def NESS_mean_field_approc(neural_data):
        mag,inverse_connected_correlations=
                get_magnetisation_and_matrix_of_connected_correlations(
                        neural_data)
        A=np.zeros((M,M))
        np.fill_diagonal(A,mag*mag)
        A=np.identity(M)-A
        A_inv=np.linalg.inv(A)
        D=[]
        for element_1 in neural_data.T:
            D_row=[]
            for element_2 in neural_data.T:
                D_row.append(np.mean(element_1*element_2))
            D.append(D_row)
        J=A_inv.dot(np.array(D).dot(inverse_connected_correlations))
```

Funkcija procjene točnosti više modela.

```
def score_multiple_models(data)
        neural_data=get_percentile_of_active_neurons(data,90).T
        lag=10
        activity=neural_data.mean(axis=0)
        target_neuron=np.argmax(activity)

        X=neural_data[:-lag,:]
        y=neural_data[lag:,:]
        y=y[:,target_neuron]
        X= pd.DataFrame(X).drop(columns=target_neuron)
        X_train,X_test,y_train,y_test=train_test_split(X,y,
                test_size=0.33,shuffle=False)

        classifiers = [
            KNeighborsClassifier(3),
            SVC(kernel="linear", C=0.025),
            SVC(gamma=2, C=1),
            DecisionTreeClassifier(max_depth=5),
            RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
            MLPClassifier(alpha=1),
            AdaBoostClassifier(),
            GaussianNB(),
            QuadraticDiscriminantAnalysis()]

           names = ["Nearest Neighbors", "Linear SVM", "RBF SVM",
                 "Decision Tree", "Random Forest", "Neural Net", "AdaBoost",
                 "Naive Bayes", "QDA"]
        accuracy=[]
        F_1=[]
        roc_auc=[]
        for clf,name in zip(classifiers,names):
            clf=clf.fit(X_train,y_train)
            if hasattr(clf, "decision_function"):
                    Z = clf.decision_function(X_test)
            else:
```

```
                Z = clf.predict_proba(X_test)[:, 1]
            roc_auc.append(roc_auc_score(y_true=y_test,y_score=Z))
            Z[Z<0.5]=0
            Z[Z>=0.5]=1
            accuracy.append(accuracy_score(y_true=y_test,y_pred=Z))
            F_1.append(f1_score(y_true=y_test,y_pred=Z))
            print(name)
            results={'names':names,'accuracy':accuracy,'F_1':F_1,
                    'roc_auc':roc_auc}
            pickle.dump(results,open('results_class_razni_prize.p','wb'))
        pickle.dump(results,open('results_class_razni_prize.p','wb'))
```

Funkcija biranja slučajnih skupova neurona.

```
def sample_neurons(corr_matrix,numb_of_neurons=769,size=9,
                numb_of_samples=10000000,check_point=10000)
        random_samples=np.random.randint(0,
                numb_of_neurons,(numb_of_samples,size))
        best_array=[]
        best_score=-1
        for i,sample in enumerate(random_samples):
            if np.unique(sample).__len__()==sample.__len__():
                temp_score=(np.sum(np.abs(corr_matrix.iloc[sample,
                        sample].values))-size)/(2*size)
                if temp_score>best_score:
                    best_score=temp_score
                    best_array=sample
                if i\%check_point==0:
                    print(i)
            else:
                continue
        pickle.dump(best_array,open("sample.p","wb"))
```

Funkcija procjene točnosti modela za predviđanje idućeg stanja neurona u ovisnosti o parametru *lag*

```
def stress_test_lag(data):
    neural_data=get_percentile_of_active_neurons(data,90).T
    lags=[x for x in range(1,200) if x%5==0]
    lags=lags+[x for x in range(200,1500) if x%10==0]
    params = {
        'task': 'train',
        'boosting_type': 'gbdt',
        'objective': 'binary',
        'metric': {'l2', 'auc'},
        'num_leaves': 31,
        'learning_rate': 0.05,
        'feature_fraction': 0.9,
        'bagging_fraction': 0.8,
        'bagging_freq': 5,
        'verbose': 0
    }
    activity=neural_data.mean(axis=0)
    target_neuron=np.argmax(activity)
    scores3_test=[]
    for lag in lags:
        X=neural_data[:-lag,:]
        y=neural_data[lag:,:]
        y=y[:,target_neuron]
        X= pd.DataFrame(X).drop(columns=target_neuron)
        X_train,X_test,y_train,y_test=train_test_split(X,y,
                test_size=0.33,shuffle=False)
        lgb_train = lightgbm.Dataset(X_train, y_train)
        model=lightgbm.train(params,lgb_train)
        scores3_test.append(roc_auc_score(y_true=y_train,
                y_score=model.predict(X_train)))
        if lag%100==0:
            print(lag)
```

Funkcija procjene točnosti modela za predviđanje idućeg stanja neurona u ovisnosti o broju ulaznih neurona *M*

```
def stress_test_perc_of_neurons(data):
    lag=100
    percentile=[x for x in range(1,10)]
    percentile=percentile+[x for x in range(10,99) if x%2==0]
    params = {
        'task': 'train',
        'boosting_type': 'gbdt',
        'objective': 'binary',
        'metric': {'l2', 'auc'},
        'num_leaves': 31,
        'learning_rate': 0.05,
        'feature_fraction': 0.9,
        'bagging_fraction': 0.8,
        'bagging_freq': 5,
        'verbose': 0
    }
    scores_test=[]
    for perc in percentile:
        neural_data=get_percentile_of_active_neurons(data,perc).T
        activity=neural_data.mean(axis=0)
        target_neuron=np.argmax(activity)
        X=neural_data[:-lag,:]
        y=neural_data[lag:,:]
        y=y[:,target_neuron]
        X= pd.DataFrame(X).drop(columns=target_neuron)
        X_train,X_test,y_train,y_test=train_test_split(X,y,
                test_size=0.33,shuffle=False)
        lgb_train = lightgbm.Dataset(X_train, y_train)
        model=lightgbm.train(params,lgb_train)
        scores_test.append(roc_auc_score(y_true=y_test,
                y_score=model.predict(X_test)))
        print(perc)
```

Funkcija pronalska optimalnog parametra T za regresijsko predviđanje idućeg stanja neurona.

```
def score_neuron_continious():
        neural_data = init_continuous()
    neural_data = neural_data.T

    activity = np.mean(neural_data, axis=0)
    neurons = np.array([x for x in range(0, activity.__len__())])

    assert neurons.__len__() == activity.__len__()

    indices = (-activity).argsort()
    neurons = neurons[indices]
    activity = activity[indices]

    time_windows = [i for i in range(2, 200) if i \% 3 == 0]
    time_windows=time_windows+[i for i in range(200,500) if i \% 5 == 0]

    params = {
        'task': 'train',
        'boosting_type': 'gbdt',
        'objective': 'regression',
        'metric': {'l2_root'},
        'num_leaves': 31,
        'learning_rate': 0.05,
        'feature_fraction': 0.9,
        'bagging_fraction': 0.8,
        'bagging_freq': 5,
        'verbose': 0
    }

    results = dict()
    for neuron in neurons:
        scores = []
        for window in time_windows:
            y = neural_data[window:-(window - 1), neuron]
            X = pd.DataFrame(substrings(window, neural_data[:, neuron]))
```

```
            X = X.drop(X.tail(window).index)
            X_train, X_test = X.iloc[:21000, :], X.iloc[21000:, :]
            y_train, y_test = y[:21000], y[21000:]
            lgb_train = lightgbm.Dataset(X_train, y_train)
            model = lightgbm.train(params, lgb_train)
            scores.append(mean_squared_error(model.predict(X_test), y_test))
        results[neuron] = scores
    pickle.dump(results, open("neurons.p", "wb"))
```

Funkcija procjene točnosti modela za predviđanje idućeg stanja mreže neurona u ovisnosti o parametru *lag*

```
def stress_neural_networks_lag()
        neural_data_raw=utilities.init()

    neural_data=utilities.get_percentile_of_active_neurons(neural_data_raw,97)

    neural_data=neural_data.T
    lags=[i for i in range(2,200)]
    lags=lags+[i for i in range(200,1500) if i%5==0]
    results=np.empty(0)
    for lag in lags:

        X=neural_data[:-lag]
        y=neural_data[lag:]

        model=Sequential()

        model.add(Dense(3*X.shape[1],activation='sigmoid'))
        model.add(Dense(X.shape[1],activation='sigmoid'))

        model.compile(loss='binary_crossentropy', optimizer='adam')

        X_train,X_test=X[:21000],X[21000:]
        y_train,y_test=y[:21000],y[21000:]
```

```
        callbacks = [
            EarlyStopping(monitor='val_loss', min_delta=0.00001, verbose=1),
            ModelCheckpoint(filepath='./tmp/weights.hdf5',
                      verbose=1, save_best_only=True)
        ]

        model.fit(X_train, y_train,batch_size=32, nb_epoch=10, verbose=1,
                validation_split=0.2,callbacks=callbacks)
        results=np.append(results,xor_sum(model.predict(X_test),y_test))

    pickle.dump(results,open("results_lag_dense.p","wb"))
```

Funkcija procjene točnosti modela za predviđanje idućeg stanja mreže neurona u ovisnosti o broju ulaznih neurona *M*

```
def stress_neural_networks_percents():
        neural_data_raw=utilities.init()

    percents=[i for i in range(90,100)]
    percents=percents+[i for i in range(50,90) if i%2==0]
    lag=10
    results=np.empty(0)
    for percent in percents:

        neural_data = utilities.get_percentile_of_active_neurons(
                neural_data_raw, percent)

        neural_data = neural_data.T
        X=neural_data[:-lag]
        y=neural_data[lag:]

        model=Sequential()

        model.add(Dense(3*X.shape[1],activation='sigmoid'))
        model.add(Dense(X.shape[1],activation='sigmoid'))
```

```
        model.compile(loss='binary_crossentropy', optimizer='adam')

        X_train,X_test=X[:21000],X[21000:]
        y_train,y_test=y[:21000],y[21000:]

        callbacks = [
            EarlyStopping(monitor='val_loss', min_delta=0.00001, verbose=1),
            ModelCheckpoint(filepath='./tmp/weights.hdf5',
                      verbose=1, save_best_only=True)
        ]

        model.fit(X_train, y_train,batch_size=32, nb_epoch=10,
                  verbose=1,validation_split=0.2,callbacks=callbacks)
        results=np.append(results,xor_sum(model.predict(X_test),
                y_test)/X_train.shape[1])
        pickle.dump(results, open("results_percent_dense_10_normal.p", "wb"))

    pickle.dump(results,open("results_percent_dense_10_normal.p","wb"))
```

Funkcija klasteriranja pomoću autoenkodera i TSNE algoritma

```
def autoencoder_and_TSNE():
        neural_data_run,neural_data_prize=utilities.init()
    data=np.concatenate((neural_data_run,neural_data_prize),axis=-1).T
    encoding_dim = 128

    input_img = Input(shape=(769,))
    temp = Dense(128, activation='relu')(input_img)
    encoded = Dense(encoding_dim, activation='sigmoid')(temp)
    temp2=Dense(128, activation='relu')(encoded)
    decoded = Dense(769, activation='sigmoid')(temp2)
    autoencoder = Model(input_img, decoded)

    encoder = Model(input_img, encoded)
    encoded_input = Input(shape=(encoding_dim,))
```

```
decoder_layer1 = autoencoder.layers[-1]
decoder_layer2 = autoencoder.layers[-2]

decoder = Model(encoded_input, decoder_layer1(decoder_layer2(encoded_input)))
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')

x_train, x_test= train_test_split(data, test_size=0.2, random_state=42)
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))

autoencoder.fit(x_train, x_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))

encoded_imgs = encoder.predict(data)

    X_embedded = TSNE(n_components=2).fit_transform(data)
    plt.scatter(X_embedded[:, 0], X_embedded[:, 1], c=boja,s=1)
    plt.show()
```

Bibliography

[1] Gallavotti, G. Statistical Mechanics. 1st ed. Berlin: Springer, 1999.

[2] Nguyen, H. C.; Zecchina, R.; Berg, J.:Inverse statistical problems: from the inverse Ising problem to data science // Advances in Physics. Vol. 66, 3(2017), str. 197-261.

[3] ”From not working to neural networking”, (25.6.2016), The Economist, https://www.economist.com/special-report/2016/06/25/from-not-working-to-neural-networking, (7.11.2018).

[4] ”The Subliminal Trick Netflix Uses to Get You to Watch Its Movies & Shows”, (26.10.2018), Thrillist, https://www.thrillist.com/entertainment/nation/how-new-netflix-recommendation-algorithm-works, (16.11.2018)

[5] ”Finding Local Destinations with Siri’s Regionally Specific Language Models for Speech Recognition”, (10.8.2018), Apple Machine Learning Journal, https://machinelearning.apple.com/2018/08/09/regionally-specific-language-models.html, (16.11.2018)

[6] ”How PayPal beats the bad guys with machine learning”, (13.4.2015), InfoWorld, https://www.infoworld.com/article/2907877/machine-learning/how-paypal-reduces-fraud-with-machine-learning.html, (16.11.2018)

[7] ”Higgs Boson Machine Learning Challenge”, (12.5.2014), Kaggle, https://www.kaggle.com/c/higgs-boson, (7.11.2018).

[8] Theoretical Cognitive Neuroscience Lab, Center for Memory and Brain, Boston University

[9] ”Ising model”, (1.11.2018), Wikipedia, https://en.wikipedia.org/wiki/Ising_model#CITEREFGallavotti1999, (7.11.2018)

[10] P. L. Krapivsky, S. Redner, and E. Ben-Naim. A kinetic view of statistical physics. Cambridge University Press, 2010.

[11] C. M. Bishop. Pattern recognition and machine learning. Springer, 2006.

[12] S. Cocco and R. Monasson. Adaptive cluster expansion for the inverse Ising problem: convergence, algorithm and tests. J. Stat. Phys., 147(2):252–314, 2012.

[13] H. E. Stanley. Introduction to phase transitions and critical phenomena. Oxford University Press, 1987

[14] M. Opper and D. Saad, editors. Advanced Mean-field Methods: Theory and Practice. The MIT Press, 2001

[15] D. J. Thouless, P. W. Anderson, and R. G. Palmer. Solution of a 'solvable model of a spin glass'. Phil. Mag., 35:593, 1977.

[16]] H. J. Kappen and F. Rodriguez. Boltzmann machine learning using mean field theory and linear response correction. Advances in Neural Information Processing Systems, pages 280–286, 1998.

[17] H. Bethe. Statistical theory of superlattices. In Proc. Roy. Soc. London A, volume 150, pages 552–575, 1935

[18] R. Peierls. On Ising's model of ferromagnetism. In Mathematical Proceedings of the Cambridge Philosophical Society, volume 32, pages 477–481. Cambridge Univ Press, 1936.

[19] R. Bapat, S. J. Kirkland, and M. Neumann. On distance matrices and Laplacians. Linear Algebra Appl., 401(0):193–209, 2005.

[20] Convenient Interface to Inverse Ising, (31.10.2018), `https://github.com/eltrompetero/coniii`, (7.11.2018)

[21] Lee, Edward D. and Daniels, Bryan C. Convenient Interface to Inverse Ising (ConIII): A Python package for solving maximum entropy models. arXiv preprint:1801.08216 (2018).

[22] H.-L. Zeng, E. Aurell, M. Alava, and H. Mahmoudi. Network inference using asynchronously updated kinetic Ising model. Phys. Rev. E, 83(4):041135, 2011

[23] Goodfellow, I.; Bengio, Y.; Courville, A. Deep Learning, MIT Press, (2016), `http://www.deeplearningbook.org/`, (7.11.2018)

[24] LightGBM, Light Gradient Boosting Machine, (7.11.2018), https://github.com/Microsoft/LightGBM, (7.11.2018)

[25] Ke, Guolin and Meng, Qi and Finley, Thomas and Wang, Taifeng and Chen, Wei and Ma, Weidong and Ye, Qiwei and Liu, Tie-Yan: LightGBM: A Highly Efficient Gradient Boosting Decision Tree, Advances in Neural Information Processing Systems 30, Curran Associates, Inc., (2017), str. 3146-3154

[26] Hyndman, Rob J.; Koehler, Anne B. (2006). "Another look at measures of forecast accuracy". International Journal of Forecasting. 22 (4): 679–688.

[27] Baldi, P.. (2012). Autoencoders, Unsupervised Learning, and Deep Architectures. Proceedings of ICML Workshop on Unsupervised and Transfer Learning, in PMLR 27:37-49

[28] van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE . Journal of Machine Learning Research, 9, 2579–2605.

[29] Lundberg, Scott M. et al. "Consistent Individualized Feature Attribution for Tree Ensembles." CoRR abs/1802.03888 (2018): n. pag.

[30] Rokach, Lior; Maimon, O. (2008). Data mining with decision trees: theory and applications. World Scientific Pub Co Inc.

[31] Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

[32] Gerven, M.A.J. van ; Bohte, S.M.: Artificial neural networks as models of neural information processing, (2017), (7.11.2018)

[33] "Receiver operating characteristic", (28.10.2018), Wikipedia,https://en.wikipedia.org/wiki/Receiver_operating_characteristic#cite_note-matlab-1, (7.11.2018)

[34] Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13, 281-305.