Univerza
v Ljubljani
Fakulteta
za gradbeništvo
in geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si

Univerzitetni program Geodezija,
smer Geodezija

Kandidat:
Klemen Kregar

Analiza uporabe umetnih nevronskih mrež za potrebe klasifikacij v deformacijski analizi

Diplomska naloga št.: 815

Mentor:
izr. prof. dr. Bojan Stopar

Somentor:
doc. dr. Mitja Lakner

Ljubljana, 26. 11. 2009

IZJAVA O AVTORSTVU

Podpisani **KLEMEN KREGAR** izjavljam, da sem avtor diplomske naloge z naslovom »**Analiza uporabnosti umetne nevronske mreže za razvrščanje v deformacijski analizi**«.

Izjavljam, da se odpovedujem vsem materialnim pravicam iz dela za potrebe elektronske separatoteke FGG.

Volčji Potok, 14. 10. 2009

BIBLIOGRAFSKO – DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK: 004:528.2:551.2(043.2)
Avtor: Klemen Kregar
Mentor: izr. prof. dr. Bojan Stopar, univ. dipl. inž. geod.
Komentor: doc. dr. Mitja Lakner, univ. dipl. mat.
Naslov: Analiza uporabnosti umetne nevronske mreže za razvrščanje v deformacijski analizi
Obseg in oprema: 103 strani, 21 enačb, 7 tabel, 64 slik
Ključne besede: umetna nevronska mreža, vektor hitrosti geodinamičnega premika, grozdenje (clustering), tekmovalno učenje, Kohonenova samoorganizirajoča se mreža

Izvleček

V diplomski nalogi želim preizkusiti sposobnost umetne nevronske mreže za grozdenje (angl.: clustering) vektorjev. Naloga je praktičen poskus grozdenja vektorjev hitrosti geodinamičnih premikov, vendar pa rezultati niso namenjeni uporabi za potrebe geodinamičnih raziskav, ampak nam služijo le za ugotavljanje, kako umetna nevronska mreža deluje oziroma kaj zmore.

Grozdenje je postopek povezovanja objektov, ki so med seboj podobni, v skupine. V nalogi te objekte predstavljajo vektorji.

Nevronska mreža, ki smo jo uporabljali, je okrnjena različica Kohonenove samoorganizirajoče se mreže. Uporabili smo le en sloj nevronov s Kohonenovim tekmovalnim učilnim pravilom. Nevroni topološko urejeno niso organizirani.

Učenje je postopek, ko mreži v ponavljajočih se korakih predstavimo vhodne podatke. V vsakem koraku se nevronom spremenijo lastnosti. Pri tekmovalnem učenju vsi nevroni hkrati sprejmejo vhodni vektor, uteži pa se posodobijo le tistemu nevronu, ki mu je ta vektor najbolj podoben. Nevron se posodobi tako, da bo v prihodnje še bolj ustrezal vektorjem, podobnim njemu.

Računanje sem opravil s pomočjo *Matlabove* orodjarne N*eural network toolbox*, v kateri so določeni tipi mrež in učilni algoritmi že vgrajeni. Za predstavitev rezultatov grozdenja sem uporabljal program *AutoCAD 2005*.

Postopek sem preizkusil v štirih nalogah. Za prve tri sem podatke izdelal sam, za četrto pa sem uporabil dejanske vektorje hitrosti geodinamičnih premikov iz Kalifornije.

Ugotovil sem, da je nevronska mreža zelo zanimiv način grozdenja, ker pri njej, za razliko od klasičnih grozdilnih metod, ne moremo predvideti, kako bo delovala. Prav zaradi tega se mi zdi za naloge, pri katerih morajo biti odgovori točni in zanesljivi, neuporabna.

BIBLIOGRAPHIC – DOKUMENTALISTIC INFORMATION

UDC: 004:528.2:551.2(043.2)
Author: Klemen Kregar
Supervisor: assoc. prof. dr. Bojan Stopar, univ. dipl. inž. geod.
Cosupervisor: assist.prof. dr. Mitja Lakner, univ. dipl. mat.
Title: Analyse of artificial neural network clustering ability for deformation analysis
Notes: 103 p., 21 eq., 7 tab., 64 fig.
Key words.: artificial neural network, geodynamic velocity vector, clustering, competitive training, Kohonen selforganising map

Abstract

In this work the clustering abilities of artificial neural network are tested. So, this thesis is a practical test of clustering abilities of the geodynamic velocity vectors, however, the results are not meant to be used by geologists, but only for learning about the neural network workings and abilities.

Clustering is a procedure of connecting similar objects to groups. In this work, the objects are represented by velocity vectors.

The network I was using is a simplified version of Kohonen self organising map. I am using a single layer of neurons with Kohonen competitive learning rule. Neurons are not topologically ordered.

Learning is a procedure of representing input data to the network. In each step the properties of neurons are adapted. Speaking for competitive training, all neurons at the same time receive input vector but parameters are adapted only to a neuron that fits vector best. The neuron is adapted in such a way that it will even better correspond when represented by similar vectors.

The computations were done by using Matlab's Neural network toolbox, where some types and learning rules are already contained. For representing results of clustering I used *AutoCad 2005*.

The procedure was tested by solving four tasks. I made data for first three by myself and the fourth was a practical test of method on real observations of geodynamic velocities from California.

I found that neural network is a very interesting way of clustering for we can not assume what the results will be like in common clustering methods. But at the same time I show that the method is unreliable, so we should not use it for tasks, where results have to be precise.

ZAHVALA

Zahvaljujem se vsem, ki so kakorkoli pripomogli k nastajanju diplomske naloge. To sta v prvi vrsti mentor, izr. prof. dr. Bojan Stopar in somentor, doc. dr. Mitja Lakner, tesno pa jima sledita doc. dr. Marko Vrabec in prof. dr. Jure Zupan, ki sta mi svetovala in pomagala pri izbiri literature - prvi v zvezi z geodinamiko in drugi v zvezi z nevronskimi mrežami.

Hvala tudi Kristini Jamšek za lektoriranje in Francetu Cerarju za pomoč pri urejanju moje diplomske naloge.

Na koncu se zelo zahvaljujem še svoji družini, zaročenki Nini ter vsem sošolcem in prijateljem, ki so me pri mojem delu podpirali in me razumeli.

KAZALO VSEBINE

KAZALO TABEL

KAZALO SLIK

1 UVOD

1.1 Namen in cilj

Namen diplomske naloge je oblikovanje skupin vektorjev podobnih premikov točk v geodinamiki z uporabo umetne nevronske mreže. Tema je zanimiva za deformacijsko analizo, kjer se sprašujemo katere točke objektov se premikajo skupaj in tako tvorijo območje znotraj katerega ni deformacije.

Premike točk v geodinamičnih raziskavah navadno ponazarjamo z vektorji hitrosti. Vektor je matematična entiteta, ki si jo lahko predstavljamo kot usmerjeno daljico (puščico $\rightarrow$). Vektor definirata dva podatka, in sicer njegova velikost in smer.

Za točke na zemeljskem površju znamo določiti, kako hitro in v katero smer se premikajo. Smiselno je sklepati, da točke, ki se premikajo z zelo podobno hitrostjo v zelo podobni smeri, tvorijo neko zaključeno homogeno celoto. Poleg podobnosti vektorjev premikov pa je za sklepanje o celoti nujno, da točke ene gmote ležijo na topološko zaključenem območju.

V nalogi so opazovanja premikov zgolj obdelana in to nima nobenega opravka s pridobivanjem takšnih podatkov.

Povezovanje podobnih vektorjev v topološko smiselna območja je delo, ki ga navadno opravlja človek; človeški možgani so namreč sposobni zaznati in prepoznati oblike ter podobne intuitivno povezati v skupine, pri tem pa znajo upoštevati tudi topološka načela bližine in sosedstva.

Glavni cilj te diplomske naloge je preizkusiti umetne nevronske mreže kot objektivno orodje, ki bo brez človeškega dejavnika optimalno formiralo skupine podobnih vektorjev. Težko je verjeti, da takšna metoda sploh obstaja, vsekakor pa je to lahko dober cilj, ki ga poskušamo doseči. Konec koncev bo rezultate

različnih analiz vedno pregledal človek in subjektivno izbral, po njegovem, najboljšo.

Umetne nevronske mreže (v nadaljevanju nevronske mreže) so računalniški sistemi, ki se zgledujejo po delovanju človeških možganov. Nevronske mreže so zelo močna orodja, ki jih v zadnjem desetletju vedno pogosteje uporabljamo. Sposobne so opravljati naloge, ki jih navadna računalniška logika ne zmore. Računalnik kljub svoji procesni sposobnosti, ki je veliko večja od človeške, nikoli ne more upoštevati toliko dejavnikov kot jih človek s svojimi možgani. Prav tako človek nikoli ne bo sposoben v tako kratkem času kot računalnik preračunati velike količine podatkov (Zupan, 2009).

Nevronske mreže pa so poskus združitve prednosti enega in drugega. »Če je za današnje računalnike značilna stroga odvisnost od programa oziroma algoritma, je značilnost nevronskega procesiranja v tem, da upošteva nakopičeno znanje v fazi učenja ter da odgovarja na vhodne podatke na način, ki je najbližji glede na izkušnje v učilni dobi.« (Dobnikar, 1990)

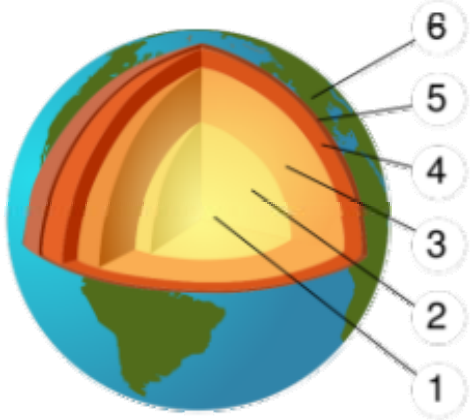
V okviru diplomskega dela želim potrditi, ovreči ali pa upravičiti odgovor »ne da se trditi« za naslednje hipoteze:

1. Nevronska mreža je sposobna razvrstiti objekte v skupine na podlagi njihovih lastnosti.
2. Vseh teoretično možnih nizov podatkov ni mogoče razvrstiti v skupine z enim samim algoritmom, vedno mora človek izbrati določeno metodo ali vsaj določiti vrednosti nekaterim parametrom.
3. Rezultat razvrščanja v skupine, ki ga pridobimo z uporabo umetne nevronske mreže, je lahko praktično uporaben na različnih področjih.

Pričujoča naloga je poskus reševanja klasičnega problema na nov način, z novo, objektivnejšo metodo. Uspešnost tega poskusa ne bo odvisna od uspešnosti preizkušane metode, ampak bo poskus sam v vsakem primeru prinesel nekaj novega znanja tako o geodinamiki kot tudi o klasifikacijskih sposobnostih nevronskih mrež.

1.2 Splošno o geodinamiki

Zaradi geodinamičnih procesov se površje Zemlje neprestano premika. Planet Zemlja je sestavljen iz več plasti, vendar se bomo osredotočili le na vrhnjo plast, na skorjo.



Plasti Zemlje:

1. notranje jedro
2. zunanje jedro
3. plašč
4. plašč (zgornji del)
5. skorja
6. površje

Plasti Zemlje (http://sl.wikipedia.org/wiki/Slika:Jordens_inre-numbers.svg)

Zemljina skorja ali litosfera je skupek »trdnih« plošč, ki plavajo na plasti staljene kamnine. Zaradi različnih endogenih sil se te plošče gibajo, pri tem pa (bolj ali manj gladko) drsijo ena mimo druge, se podrivajo ali gubajo.

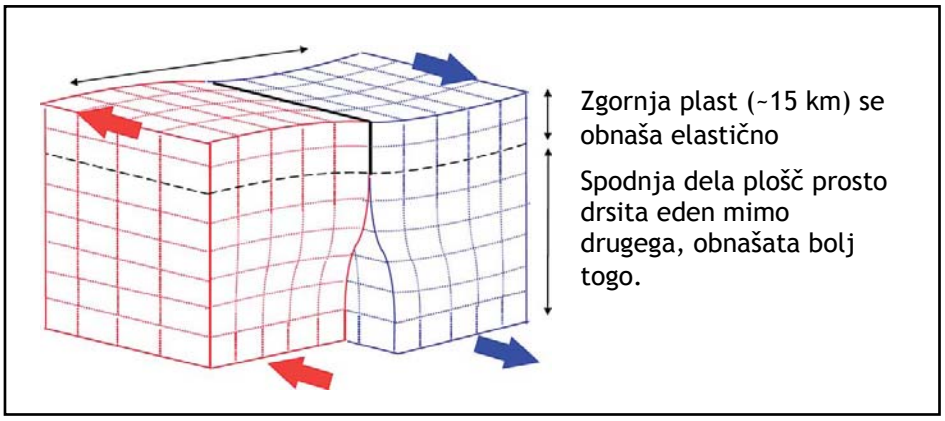
Lastnost litosferskih plošč je, da vedno zasedajo zaključeno zvezno območje. Ni mogoče, da bi imela plošča znotraj svojega območja otoke druge plošče. Sicer pa so oblike plošč lahko poljubne.

Od pojava globalnih navigacijskih satelitskih sistemov je geodezija sposobna določati premike teh plošč v absolutnem smislu. V preteklosti so bile izhodiščne točke za kakršnakoli opazovanja vedno realizirane na zemeljskem površju, zato je bilo mogoče opazovati samo relativne premike ene plošče glede na drugo. GNSS koordinatni sistem realizirajo položaji satelitov na svojih tirnicah; te so odvisne samo od težnostnega polja Zemlje in od njenega težišča, niso pa neposredno odvisne od položajev litosferskih plošč. Tako lahko z GNSS opazovanji določimo absolutne premike teh plošč (Blewitt, 2007, str. 351).

Zemeljsko skorjo lahko obravnavamo na različnih nivojih. Glavnih litosferskih plošč je približno dvajset. Na mejnih območjih se pojavljajo tudi manjši objekti, ki jih imenujemo bloki. Stroga obravnava bi morala biti ločena, saj se bloki obnašajo drugače kot plošče.

Premikanje plošč po površini Zemlje opisujemo kot premike delov površja krogle. Kakršenkoli premik površine krogle lahko opišemo z Eulerjevim polom ter kotno hitrostjo (Blewitt, 2007, str. 377). Vektorji hitrosti premikov zemeljskega površja se ujemajo s temi modeli predvsem v notranjosti plošč, na mejnih območjih med ploščami pa prihaja do odstopanj. Vzrok za odstopanja je elastičnost plošč.

Na spodnji sliki je prikazana situacija drsenja plošč ene ob drugi. V globini plošči drsita ena mimo druge zvezno, na površju pa prihaja do trenja oziroma bolje rečeno lepljenja. Kadar zdrsne tudi vrhnja plast plošč, pride do potresa (Blewitt, 2007, str.383).



Prikaz gibanja litosferskih plošč (Blewitt, 2007, str.377)

Tudi bloki na mejnih območjih med ploščami se v principu obnašajo podobno, vendar je tam vpliv elastičnosti manjši, kar pri manjših površinah ne pride toliko do izraza.

Zaradi kompleksnosti mehanike pri geodinamičnih premikih ta diplomska naloga morda ne bo primerna za praktično uporabo v geologiji. V nalogi so prikazana prizadevanja, kako določiti enolične meje med območji, s tem pa zanemariti

elastičnost plošč. Za korektnejšo obravnavo dogajanja bi morali uporabiti Kohonenove samoorganizirajoče se mreže, s katerimi pa ni mogoče enolično potegniti meja med vektorji.

Geologija naj služi kot ozadje nalogi, oziroma kot možen primer uporabe. Cilj naloge je enostavno oblikovanje skupin podobnih objektov, nikakor pa ne izdelava orodja, ki zna popolnoma korektno izdelati model geodinamičnega premikanja.

1.3 Nevronske mreže

Nevronska mreža je v splošnem matematični model, ki v nekaterih pogledih skuša oponašati možgane. Gre za mrežo, sestavljeno iz »nevronov« in povezav med njimi. Nevron je osnovni element nevronske mreže. Najbolj splošno lahko trdimo, da je nevron element, ki sprejme nekaj podatkov, jih procesira in vrne en rezultat. Ta rezultat je lahko vhodni podatek za naslednje nevrone.

Nevronske mreže »... se po eni strani zgledujejo po strukturnih in funkcijskih lastnostih bioloških živčnih sistemov, po drugi strani pa upoštevajo stanje razvoja računalniške tehnike z vsemi njenimi omejitvami in pomanjkljivostmi.« (Dobnikar, 1990)

Bistvena lastnost nevronskih mrež je učenje. Mrežo moramo naučiti, preden jo lahko uporabljamo. Mrežo si predstavljajmo kot črno škatlo, v katero položimo podatke, vrne pa nam nek rezultat; ta rezultat pomeni odgovor na neko raziskovalno vprašanje. Za učenje je potreben dovolj velik niz podatkov s pripadajočimi rezultati, to pomeni, s takšnimi rezultati, kakršne bi želeli, da jih mreža vrne. Mreža na podlagi razlik med dobljenimi in želenimi rezultati prilagaja lastnosti svojih nevronov tako dolgo, da dobimo pravilne rezultate.

Mreže glede na postopek učenja delimo na mreže z nadzorovanim in nenadzorovanim učenjem; pri slednjem poznavanje želenih rezultatov ni potrebno.

Mreže, različne po funkcionalnosti, v glavnem opravljajo tri opravila: aproksimacije, klasifikacije in razvrščanje objektov v skupine.

Pri nenadzorovanem učenju mreži ne povemo, kakšne rezultate v kakšnih situacijah želimo, zato mora mreža sama »odkriti« značilnosti podatkov. Takšna mreža se ponavadi uporablja za razvrščanje objektov v skupine in prav takšna mreža je tudi uporabljena za reševanje raziskovalnega problema te naloge.

1.4 Terminologija

Pred začetkom pisanja naloge je treba razčistiti nekaj stvari v zvezi z izrazoslovjem; gre za nekaj osnovnih terminov, ki so pogosto uporabljeni in lahko zvenijo ali dvopomensko ali pa se nam zdijo tuji. Večino snovi je črpane iz angleških virov, zato se določene težave pojavljajo tudi pri prevajanju.

1. Besedna zveza »nevronska mreža« se v nalogi pojavlja zelo pogosto. Nevronska mreža je splošen izraz, ki lahko pomeni tako biološke možgane kot umetno nevronsko mrežo, ker je v resnici oboje mreža, sestavljena iz nevronov. Kadar je v nalogi uporabljena ta zveza, imam v mislih vedno umetno nevronsko mrežo oziroma njeno simulacijo. V začetku razvoja so bile nevronske mreže realizirane kot kompleksna elektronska vezja. Delovanje takšne mreže je moč simulirati z razvojem računalnikov. Besedna zveza »nevronska mreža« v nalogi pomeni računalniško simulacijo v mrežo povezanih podatkov in rezultatov.

2. Angleški besedi »cluster« in »clustering« sta v slovenskem prevodu samostalnik grupo, gručo, dobesedno grozd in glagol grupiranje, »gručenje« oziroma grozdenje. Praktično je celotna naloga posvečena prav postopku »clusteringa«, to je tvorjenju skupin oziroma razvrščanju objektov v skupine; zato je bilo treba za ti dve besedi izbrati ustrezen prevod in ni mi uspelo najti bolj ustreznih besed, kot sta grozd in grozdenje. Grupa se zdi

preveč neformalna in zastarela, z besedo gruča pa ne moremo tvoriti glagola.

Ponuja se še beseda »klasifikacija«, vendar ta beseda pomeni nekaj drugega. »Clustering« je postopek oblikovanja skupin podobnih elementov, medtem ko klasifikacija pomeni razvrščanje elementov v obstoječe ali predpostavljene skupine (razrede). Tudi praktično se postopka »clusteringa« in klasifikacije bistveno razlikujeta.

Beseda grozdenje je bolj uveljavljena na področju podjetništva in ekonomije. Tam ju prav tako uporabljajo kot prevod besede »clustering«, zato ni razloga, da ju ne bi uporabljali tudi v tehničnih strokah. Ker pa sta besedi »grozd« in »grozdenje« našim ušesom še tuji in včasih neintuitivni, bosta poleg njiju tudi besedi »skupine« in »razvrščanje v skupine«.

3. Tretja beseda je »vektor«. Vektor je po definiciji matematična količina, definirana z dolžino in smerjo.

V nalogi se vektorji pojavljajo v dveh oblikah. Prva oblika so geodinamični vektorji; to so usmerjene daljice, ki ponazarjajo premike ali hitrosti premikov točk na zemeljskem površju.

Druga oblika so vektorji kot oblika zapisa podatkov. V matematiki vektorje zapisujemo kot niz števil znotraj oglatega oklepaja, na primer: [2, 3, 5, 7]. Pri postopkih, ki so obravnavani v tej nalogi, se podatki o objektih vedno zapisujejo v obliki vektorjev. Torej *vektor geodinamičnega premika* zapišemo v obliki *matematičnega vektorja*, kot niz štirih števil, ki predstavljajo štiri lastnosti *vektorja geodinamičnega premika*.

Če kakršenkoli vektor imenujemo samo vektor, praktično ni velike razlike, vendar se zdi, da je razliko pomembno definirati, kajti v nalogi za boljšo preglednost izražanje pogosto poenostavljam.

1.5 Sestava diplomske naloge

Diplomska naloga je sestavljena iz šestih poglavij. V uvodu je zastavljen cilj in glavni namen naloge, obrazložitev nekaterih dejstev v zvezi z geodinamiko, opisano je, kaj nevronske mreže sploh so, poleg tega pa je definiranih nekaj pojmov, ki jih bomo v nalogi večkrat zasledili.

V drugem poglavju je predstavljenih nekaj klasičnih metod grozdenja. Pri vsaki skušano pokazati, zakaj ni primerna za reševanje mojega problema. Nekatere metode so precej kompleksne in tudi perspektivne. Če ne bi bila rdeča nit naloge nevronska mreža, bi se verjetno še bolj poglobil v spektralno grozdenje.

Tretje poglavje govori o nevronskih mrežah. V uvodnem delu pogledamo, kako delujejo človekovi možgani in kako so v zadnjem stoletju kot njihov produkt nastajale umetne nevronske mreže. Nato so opisana osnovna načela oziroma ideja umetne nevronske mreže. Ta načela so načeloma skupna vsem vrstam nevronskih mrež, ki so opisana v nadaljevanju. Poglavje se konča z opisom tekmovalnega učenja, ki je ključni element oziroma jedro mojega raziskovalnega dela.

Naslednje poglavje je praktično obarvano. Začne se z opisom podatkov, ki sem jih obdeloval, in opisom postopka dela. Sledi opis postopka dela ter stopnjevanje kompleksnosti problemov, ki sem jih reševal - najprej delo z desetimi in stotimi izmišljenimi vektorji, nato pa še delo na konkretnem primeru resničnih geodinamičnih premikov.

Sledi poglavje o ugotovitvah, v katerem so na kratko povzeta dejstva, ki sem se jih naučil ob delu, nato pa še zaključek, v katerem so zaključene teme, ki so v začetku uvoda še odprte. Poskusil bom tudi odgovoriti na raziskovalna vprašanja oziroma hipoteze.

2 O RAZVRŠČANJU OBJEKTOV V SKUPINE NA SPLOŠNO

Pri študiju postopkov, s katerimi bi se lahko lotil svojega problema, sem se najpogosteje srečeval s pojmoma »clustering« in »classification« oziroma grozdenje in klasifikacija.

Pojem grozdenje pomeni postopek, pri katerem se skupina podatkov razdeli na logične podmnožice, in to tako da v vsako podmnožico spadajo objekti s podobnimi lastnostmi.

Klasifikacija pa na drugi strani pomeni samó razvrščanje objektov v razrede, pri čemer moramo že vnaprej vedeti, kakšne lastnosti objektov zastopa kakšen razred.

Za moj problem so metode klasifikacije neuporabne. Uporabil bi jih lahko le v primeru, če bi že poznal homogena območja premikov, da bi vanje lahko umestil nova opazovanja z obmejnih območij. Zaenkrat se bom bolj osredotočili na postopke grozdenja.

2.1 Osnovni pojmi

Razvrščanje v skupine je postopek, pri katerem množico objektov razdelimo na podmnožice, *grozde* oziroma skupine, in to tako da so v vsaki skupini objekti s podobnimi lastnostmi. Gre torej za postopek iskanja podobnih objektov.

Obstaja več načinov, s katerimi bi lahko opisali hierarhijo metod grozdenja; ob tem pa menim, da na splošno velja naslednje: metode se delijo na hierarhične in nehierarhične; hierarhične metode se naprej delijo na združevalne in delilne (»agglomerative and divisive«), nehierarhične pa vsebujejo razdeljevalne in spektralne metode.

Preden si ogledamo metode razvrščanja v skupine, moramo spoznati dva pojma, ki ju potrebujemo pri večini postopkov. Gre za razdalje in pravila za povezovanje; z njima opisujemo podobnost objektov oziroma pripadnost objekta grozdu. Informacije o tem sem črpal s spletne strani *Statsof* (2007).

2.1.1 Razdalje

Kriterij podobnosti med objektoma je »razdalja«. Če sta dva objekta blizu skupaj, je razdalja med njima majhna, zato sta si podobna. Bolj kot sta si med seboj oddaljena, bolj sta si različna.

Vsak objekt je podan kot niz *n* števil, ki so podatki o tem objektu, ta števila pa lahko pomenijo tudi koordinate, ki določajo položaj objekta v *n*-dimenzionalnem prostoru. Objekti s podobnimi podatki se v tem prostoru pojavljajo blizu skupaj.

Razdalje med objekti, ki jih uporabljamo kot mero podobnosti, lahko definiramo na različne načine, izračunamo pa jih iz podatkov dveh objektov.

- Evklidska razdalja

  Ta je najbolj splošno uporabljana razdalja. Definirana je kot geometrična razdalja v multidimenzionalnem prostoru, kjer je vsaka točka podana z nizom koordinat

  $$A = (A_1, A_2, ..., A_n)$$

  $$d(A,B) = \sqrt{\sum_i (B_i - A_i)^2}$$

- Kvadratna evklidska razdalja

  Ta razdalja je podobna evklidski, le da so dolžine kvadrirane. To pomeni, da se razdalje pri oddaljevanju kvadratično povečujejo, kot na primer pri gravitacijski sili

  $$d(A,B) = \sum_i (B_i - A_i)^2$$

- Razdalja manhattan

  Razdaljo manhattan si zamišljamo kot potovanje po pravokotnih mestnih ulicah. Rezultat je vsota vseh koordinatnih razlik.

$$d(A,B) = \sum_i |B_i - A_i|$$

- Chebychevova razdalja

  To razdaljo uporabljamo, kadar želimo dva objekta označiti za različna, če se razlikujeta vsaj v enem od podatkov.

$$d(A,B) = \max|B_i - A_i|$$

- Potenčna razdalja

  S potenčno razdaljo lahko poljubno utežimo razdalje s tem, da izbiramo parametra *p* in *q*. Pri evklidski razdalji razlike kvadriramo in korenimo, tu pa potenciramo s *p* in korenimo s *q*.

$$d(A,B) = \sqrt[q]{\sum_i (B_i - A_i)^p}$$

Naštete so samo najbolj pogoste mere za razdalje.

2.1.2 Pravila za povezovanje

Z razdaljami znamo določiti podobnost med dvema objektoma; v postopkih razvrščanja v skupine pa bomo morali še večkrat ugotoviti, koliko so si skupine med seboj podobne oziroma kolikšna je pripadnost objekta skupini (razdalja med skupino in objektom). Odločiti se torej moramo, s katerimi razdaljami bomo opisovali podobnost skupin. Tudi za ta namen imamo na voljo več možnosti.

- Enojna povezava (nearest neighbour)

  Razdalja med dvema skupinama je definirana kot razdalja med najbližjima objektoma iz različnih skupin. Skupine se bodo tako povezovale v verige.

- Popolna povezava (furthest neighbour)

  Razdalja med skupinama je razdalja med dvema objektoma iz teh dveh skupin, ki sta najbolj oddaljena. Navadno se ta način obnese v primerih, pri katerih so podatki že po naravi razdeljeni v razrede. Če so objekti povezani v verige, je bolje uporabiti prvi način.

- Neuteženo povprečje parov

  Razdalja se izračuna kot srednja vrednost razdalj med vsemi možnimi pari objektov iz dveh različnih skupin.

- Uteženo povprečje parov

  Postopek je enak kot v prejšnjem primeru, le da se pri izračunu srednje vrednosti kot utež uporabi velikost skupine (to pomeni število objektov, ki jih objema). Pravilo je dobro uporabiti v primerih, pri katerih predvidevamo, da bodo velikosti skupin zelo različne.

- Neuteženi centroid

  Izračunamo središče skupine kot srednjo vrednost vseh njegovih objektov; imenujmo ga centroid. Razdalja med skupinama je definirana kot razdalja med centroidoma teh dveh območij.

- Uteženi centroid

  Način je podoben prejšnjemu, le da bolj utežimo večjo skupino. Tudi ta metoda je primernejša, kadar sumimo, da velikost skupin ni enaka.

- Wardova metoda

  Pri tem načinu za določitev razdalje med skupinami uporabljamo variančno analizo. Gre za iskanje najboljšega para po metodi najmanjših kvadratov. Združita se tisti skupini, katerih unija bo imela najmanjšo varianco. Metoda naj bi bila zelo učinkovita, vendar so rezultat ponavadi zelo majhne skupine.

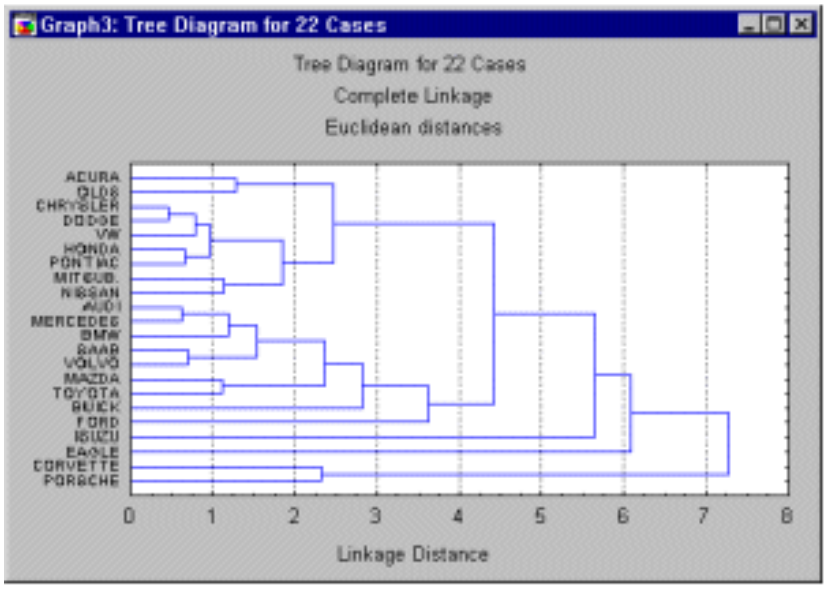
2.2 Hierarhično grozdenje

Hierarhično grozdenje je postopek, v katerem končni rezultat ni enolična razdelitev objektov v skupine, ampak hierarhično drevo ali dendrogram. Ta pravzaprav predstavlja, kako se v postopku združevanja posamezni objekti prek skupin združujejo v celoto oziroma kako z razdeljevanjem celota prek skupin razpade v posamezne objekte. Opisane postopke sem črpal s strani *Statsoft* (2007).

2.2.1 Združevanje (*tree clustering)*

Namen metode združevanja je povezovati med seboj podobne objekte v večje skupine. Rezultat te metode je hierarhično drevo.

Zamislimo si ležeče hierarhično drevo.



Hierarhično drevo - Dendrogram (Statsoft, Inc. 2007)

Na levi strani začnemo s tem, da vsak objekt predstavlja svojo skupino. Potem začnemo počasi dvigovati kriterij, ki pomeni, koliko oddaljeni objekti se lahko povežejo v skupino. Tako se začnejo do tedaj posamezni objekti združevati v skupine po dva. Ko kriterij še povečujemo , se pari začnejo povezovati med sabo

in nove skupine spet med sabo (dogajanje na sliki, če jo opazujemo od leve proti desni). Nazadnje se povsem na desni združita še zadnja dva grozda.

Proces lahko ustavimo tudi pred združitvijo vseh skupin v celoto. Za to imamo na voljo kriterij razdalje ali številski kriterij; prvi pomeni, da določimo maksimalno razdaljo, pri kateri se grozdi še lahko združijo. Če so si med seboj še bolj oddaljeni, se proces ustavi, preden se združijo.

Številski kriterij pomeni, da postopek teče, dokler se ne ustvari želeno število skupin.

Na abscisni osi grafa lahko za vsako skupino, ki se je formirala iz prejšnjih dveh, razberemo, pri katerem kriteriju razdalje se je to zgodilo. Končno je mogoče vsako vejo hierarhičnega drevesa obravnavati kot svoj grozd.

2.2.2 Deljenje

Metoda je nasprotna združevanju. Na začetku predpostavimo en grozd, ki povezuje vse objekte. Z manjšanjem dolžinskega kriterija se pojavljajo podskupine, ki so bolj oddaljene od začetnega grozda; tako celota razpada na dele in povsem na koncu mora ostati vsak objekt zase.

2.3 Razdeljevalno grozdenje

2.3.1 Metoda k-sredin

Glavna slabost metode *k-sredin* je, da moramo za vhodni podatek povedati, koliko skupin želimo dobiti. Število grozdov *k* mora biti torej vnaprej znano. Obstajajo sicer metode, s katerimi lahko predvidimo optimalno število skupin, lahko pa delamo tudi s poskušanjem in se nazadnje odločimo, pri katerem številu je rešitev najboljša.

Pri postopku z metodo *k-sredin* lahko, tako kot prej, uporabimo različne mere za razdalje in različna pravila za ugotavljanje razdalj med grozdi. Za opis delovanja postopka bomo uporabili primer z evklidsko razdaljo in neuteženim centroidom, ker se zdi najlažje predstavljiva (Statsoft, 2007).

Potek postopka je naslednji:

- Izberemo število skupin *k*.
- V hiperprostoru podatkov, v meje, ki jih določa razpon podatkov, naključno postavimo *k* točk, ki predstavljajo središča skupin.
- Vsak objekt pripišemo tistemu središču, ki mu je najbližje.
- Vsako središče premaknemo v težišče točk (objektov), ki jih je povezalo.
- Nato spet izračunamo razdalje, poiščemo najbližje objekte in iz njih izračunamo novo središče.
- Postopek ponavljamo toliko časa, da se središča grozdov nehajo premikati oziroma dokler premiki niso manjši od zahtevane meje.

S statističnega vidika postopek lahko opišemo takole: algoritem poskuša ustvariti takšne skupine, da bo varianca objektov znotraj njih kar najmanjša, varianca med njimi pa kar največja; to bo doseženo z vključevanjem in izključevanjem objektov v skupine in iz njih.

Prednosti postopka *k-sredin* sta preprostost in hitrost, kar omogoča obdelave velikih zbirk podatkov.
Slabost tega pa je, da postopek za iste podatke ne vrne nujno dvakrat istega rezultata, kar je posledica naključnega izbora začetnih središč. Metoda sicer minimizira varianco znotraj grozdov, vendar pa ne zagotavlja globalnega minimuma varianc. Poleg tega zahteva podatke, ki jim je mogoče definirati sredino, kar pa se ne zgodi vedno.

2.3.2 Mehko grozdenje c-sredin

Mehko (fuzzy) grozdenje je metoda grozdenja, ki dopušča, da isti objekt pripada dvema ali več skupinam (Bezdek, 1981). Pri grozdenju *k-sredin* vsak objekt pripada samo eni skupini. Metoda *c-sredin* temelji na minimizaciji funkcije

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m} \left\| x_i - c_j \right\|^2$$

kjer je *m* realno število večje ali enako ena, u_{ji} je stopnja pripadnosti objekta x_i grozdu *j*, x_i je *i*-ti izmed *N* *d*-dimenzionalnih podatkov, c_j je središče *j*-tega izmed *C* grozdov (*d*-dimenzionalna točka) in $\|*\|$ je znak za eno izmed razdalj, ki izraža podobnost med podatkom in središčem grozda.

Mehko razvrščanje se izvaja z iterativnim postopkom optimizacije funkcije. To dosežemo s posodabljanjem vrednosti pripadnosti objektov u_{ji} in centrov grozdov c_j.

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\left\| x_i - c_j \right\|}{\left\| x_i - c_k \right\|}\right)^{\frac{2}{m-1}}} \qquad c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

Iteracije se končajo, ko so spremembe pripadnosti objektov dovolj majhne

$$\max_{ij}\left\{\left| u_{ij}^{(k+1)} - u_{ij}^{(k)} \right|\right\} < \varepsilon$$

kjer je ε predpostavljeni kriterij večji od nič in manjši od ena. Tak proces konvergira proti lokalnemu minimumu funkcije J_m.

Algoritem teče po naslednjih korakih:

- Inicializacija matrike *U*, ki vsebuje pripadnosti vseh objektov vsem skupinam. Začetna matrika je $U^{(0)}$.
- Za *k*-ti korak (k = 1,2 …) izračunamo središča grozdov $C^{(k)} = \left[c_j\right]$ z uporabo matrike $U^{(k)}$ in enačbe za c_j, ki je navedena zgoraj.
- Matriko $U^{(k)}$ posodobimo z uporabo vrednosti iz matrike $C^{(k)}$ in enačbe za u_{ji}. Dobimo novo matriko $U^{(k+1)}$.
- Če je $\left\|U^{(k+1)} - U^{(k)}\right\| < \varepsilon$, postopek zaključimo, v nasprotnem primeru pa algoritem ponovimo od drugega koraka naprej.

Metoda mehkega grozdenja ima podobne prednosti in slabosti kot metoda *k-sredin*. Grozdenje z maksimizacijo verjetnosti je bolj statistično usmerjena, a prav tako uporablja delno pripadnost skupinam. Metoda EM konvergira hitreje in je bolj priljubljena kot mehko grozdenje.

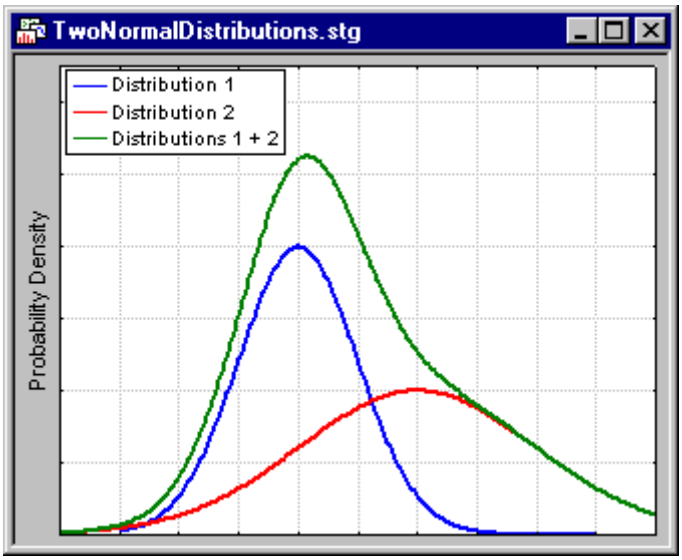
2.3.3 Grozdenje z maksimizacijo verjetnosti - *EM (Expectation maximization)*

Metoda je zelo podobna metodi *k-sredin*. Glavni namen te tehnike je zaznati skupine v opazovanjih (spremenljivke) in potem opazovanja pripisati tem skupinam.

EM grozdenje pomeni nadgraditev klasičnih metod razvrščanja v skupine zaradi dveh stvari. Prva je ta, da namesto pripisovanja opazovanj skupinam (z namenom povečevanja razlik v sredinah) *EM* grozdenje računa verjetnosti pripadnosti skupinam, ki temeljijo na eni ali več verjetnostnih porazdelitvah. Cilj postopka je maksimizirati celotno verjetnost oziroma podobnost podatkov, ki so v končni skupini. Druga stvar pa je ta, da za razliko od *k-sredin* razvrščanja v skupine lahko *EM* algoritem obdeluje tako tekoče kot kategorične spremenljivke.

Denimo, da merimo enojne tekoče spremenljivke v velikem nizu opazovanj in da komplet vsebuje dva grozda z različnima sredinama in različnima standardnima odklonoma; v obeh primerih je spremenljivka normalno porazdeljena.

Porazdelitev vrednosti naj bi izgledala takole:



Verjetnostne porazdelitve spremenljivk (Statsoft, 2007)

Na sliki vidimo porazdelitve obeh skupin in vsoto njunih porazdelitev. V resnici poznamo samo vsoto in iz nje želimo sklepati na srednjo vrednost in standardni odklon skupin. Ta postopek je naloga *EM*-grozdenja.

2.3.4 Algoritem kvalitativnega praga - *QT (Quality Threshold)*

Metodo *QT* so odkrili leta 1999 z namenom razvrščanja genov v skupine. Računski procesi so sicer zahtevnejši kot pri *k-sredin*, vendar metoda *QT* ne zahteva vnaprej znanega števila skupin, poleg tega pa ob ponovitvah vedno vrne enak rezultat.

Postopek je naslednji:

- Uporabnik določi največji dovoljeni radij grozda.
- Okrog vsakega objekta ustvarimo grozd s predpisanim radijem, tako da se vanj uvrstijo vsi tisti objekti, ki so bližji od tega predpisanega radija.

- Skupino, ki je zajela največ objektov, si zapomnimo kot prvi pravi grozd, nato pa vse njene člane odstranimo iz nadaljnjega postopka.
- Nadaljujemo z zmanjšanim nizom objektov.

Kot pravilo za povezavo (razdalja grozd — objekt) naj se uporablja popolna povezava, to je razdalja med objektom in njemu najbolj oddaljenim objektom iz skupine.

2.4 Spektralno grozdenje

Dan je niz objektov *A*, podobnostna matrika *S* pa je definirana tako, da vsaka njena vrednost S_{ij} predstavlja mero podobnosti med elementoma *i* in *j* množice *A*.

Tehnika spektralnega grozdenja uporabi spekter podobnostne matrike za redukcijo dimenzionalnosti, tako da se razvrščanje v skupine izvaja v prostoru z manj dimenzijami.

Ena od takih tehnik je algoritem *Shi-Malik*, ki je pogosto uporabljan za segmentacijo podob. Ta algoritem razdeljuje objekte v dva niza (S_1, S_2) na podlagi lastnega vektorja ν, ki ustreza drugi najmanjši lastni vrednosti Laplaceove matrike

$$L = I - D^{-1/2} S D^{-1/2}$$

in matrike *S*, kjer je *D* diagonalna matrika

$$D_{ii} = \sum_j S_{ij}$$

Razdeljevanje lahko poteka na različne načine. Prva možnost je, da vzamemo mediano *m* komponent vektorja ν in nato vse objekte, katerih komponenta ν je večja od *m,* postavimo v množico S1, ostale pa v S2. S takim načinom razdeljevanja je ta algoritem primeren tudi za hierarhično grozdenje.

2.5 Sklep

Poleg opisanih načinov razvrščanja v skupine je vredno omeniti še konceptualno grozdenje; deluje na principu drevesne strukture, kjer vsaka veja predstavlja svoj koncept. Veje se lahko delijo na »podveje«. Vsaka od vej vsebuje svoje člane (objekte, ki jih razvrščamo) oziroma člane svojih podvej. Postopek za reševanje našega problema ni zanimiv, saj obdeluje kvalitativne podatke (na primer: spol, krilatost, raso …), medtem ko so podatki o vektorjih geodinamičnih premikov kvantitavni ali številski.

Ta naloga govori o nevronskih mrežah, ker je predpostavljeno, da nobena od klasičnih metod razvrščanja objektov v skupine ne more zadovoljivo rešiti zastavljenega problema. Poglejmo, zakaj.

Hierarhično grozdenje vedno vrne dendrogram. V našem primeru to pomeni, da bi se morali odločiti, na katerem nivoju (pri kako velikem kriteriju razdalje) je končni rezultat. Ta ocena bi bila subjektivna, česar pa nočemo. Želimo si, da bi postopek sam povedal, kdaj se dva vektorja dovolj razlikujeta, da padeta v različni skupini.

Pri razdeljevalnem grozdenju imamo metodi *k* in *c-sredin* ter algoritma *EM* in *QT*. Za prvi dve je treba navesti apriori število skupin, pri algoritmu *QT* pa bi morali (podobno kot pri hierarhičnem) določiti najvišjo mejo, v kateri bi vektorji spadali skupaj. Še najbolj zanimiva je metoda *EM*. Pri njej moramo poznati statistične porazdelitve spremenljivk niza podatkov. Porazdelitve v nizih podatkov, ki jih bomo uporabljali, sicer niso dane, vendar bi se jih morda dalo izračunati. Metodo *EM* bi bilo morda vredno preizkusiti, prav tako pa tudi spektralno grozdenje. Ker pa smo se odločili, da v tej nalogi raziskujemo nevronske mreže, bomo ti dve metodi zaenkrat pustili ob strani.

3 NEVRONSKE MREŽE

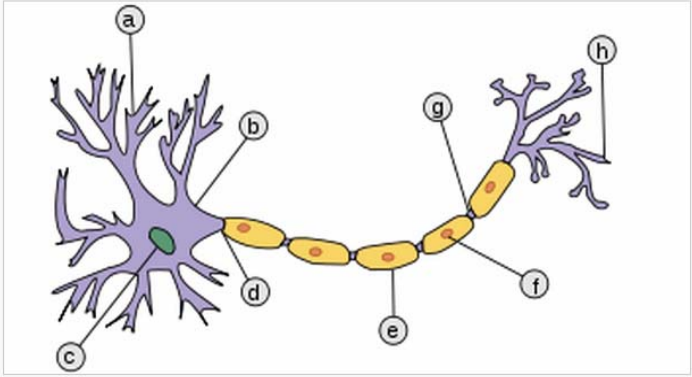
3.1 Uvod

V tem poglavju si bomo najprej na kratko pogledali, kako delujejo človeški možgani, nato pa bomo videli, kako na osnovi istih načel sestavljamo nevronske mreže. Preleteli bomo vrste nevronskih mrež in izvedeli, zakaj se katera od njih uporablja. Nazadnje si bomo natančneje ogledali postopek tekmovalnega učenja, ki predstavlja najboljšo rešitev našega problema.

3.1.1 Človeški možgani

Tema se zdi zanimiva zaradi razumevanja, v čem so si umetne in organske mreže med seboj podobne. Ker pa ta tema nima ključnega pomena za to nalogo, sem izbral nekoliko bolj poljuden vir informacij, in sicer Wikipedijo (Wikipedija - Nevron).

»Možgani so pretežno sestavljeni iz dveh vrst celic: nevronov in glialnih celic. Slednje pretežno služijo podpori in varovanju nevronov. Nevroni prenašajo informacije v obliki električnih sunkov, znanih kot akcijski potenciali. Z drugimi nevroni v možganih in drugod po telesu komunicirajo tako, da prek rež, imenovanih sinapse, pošiljajo molekule kemijskih snovi, imenovane nevro-transmiterji. Možgani manjših nevretenčarjev, kot denimo žuželk, so sestavljeni iz približno milijona nevronov, možgane večjih vretenčarjev pa lahko sestavlja več kot sto milijard nevronov. Možgani človeka so glede na telesno velikost nadpovprečno veliki in kompleksni.«



Shema unipolarnega nevrona
a: dendrit
b: telo (soma)
c: jedro
d: nevrit
e: mielinska ovojnica
f: Schwannova celica
g: Ranvierov zažetek
h: končni del nevrita

Shema nevronske celice - Nevrona (http://sl.wikipedija.org/wiki/Nevron)

»Številni nevroni so visoko specializirani in se po izgledu močno razlikujejo. V splošnem so sestavljeni iz naslednjih komponent:

- perikarion ali soma,
- dendriti,
- nevrit.

Soma je telo nevrona z jedrom in citoplazmo. V somi poteka večina beljakovinske sinteze.

Dendriti so izrastki iz some, navadno drevesasto razvejani. Skupno strukturo vseh dendritov enega nevrona imenujemo dendritsko drevo in je glavni del nevrona za sprejemanje informacij od sosednjih nevronov.

Najdaljši izrastek iz some je običajno nevrit, ki je lahko dolg tudi en meter. Po nevritu se prenaša impulz od some proti naslednjemu nevronu ali efektorni celici. Impulz včasih potuje tudi v smeri proti somi. Večina nevronov ima le en nevrit, ki pa se zelo razveja in tvori povezave z različnimi celicami. Na mestu, kjer iz some izstopa nevrit, je največje število napetostno odvisnih natrijevih kanalčkov; na

tem mestu se namreč seštejejo vsi dražljaji, ki pridejo od dendritov preko some do nevrita. Če je vsota dražljajev (spremembe membranskega potenciala) večja od pražne vrednosti, se na tem mestu sproži akcijski potencial.

Konec nevrita imenujemo živčni končič. Živčni končič je specializirana struktura, ki vsebuje mešičke z živčnim prenašalcem. Ko akcijski potencial dospe do membrane živčnega končiča, se na membrani odprejo kalcijevi kanalčki. Zaradi vdora kalcija v notranjost celice se mešički z živčnim prenašalcem zlijejo s plazmalemo in se sprostijo v sinaptično špranjo. Nevroni komunicirajo med seboj s pomočjo živčnih prenašalcev.

Nevriti in dendriti v osrednjem živčevju so debeli okrog en mikrometer, perikarioni pa so znatno debelejši (njihov premer je okoli 10–25 mikrometrov). Nevriti motoričnih nevronov so lahko dolgi tudi en meter.« (Wikipedija - Nevron)

3.1.2 Razvoj umetnih nevronskih mrež

Človeštvo že od nekdaj poskuša razumeti, kako delujejo človeški možgani. Téma je postala toliko bolj zanimiva v dobi računalnikov. Stroje znamo naučiti, da opravljajo fizična dela namesto človeka, želimo pa si, da bi znali stroji tudi »razmišljati« na podoben način kot človek.

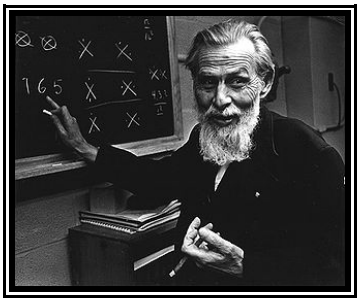
Klasični računalniški algoritmi so sposobni, da zelo hitro procesirajo podatke, vendar pod pogojem, da računalnik pozna zaporedje korakov. Človeški možgani pa so se zmožni učiti postopke, tudi če poznajo samo začetno situacijo in končni rezultat brez vmesnega postopka.

»Bistvo nevronskih mrež je v tem, da med učenjem same ugotovijo pravilo, ki izhodne podatke povezuje z vhodnimi. To pomeni, da se lahko naučijo tudi več in bolje kot učitelj oziroma človek. Ko je nevronska mreža naučena (kar lahko traja dlje časa), deluje tudi v situacijah, s katerimi v procesu učenja ni imela opravka. To pomeni, da lahko rešuje tudi naloge, kjer ne obstaja rešitev v obliki zaporedja

korakov (kot npr. pri računalniških algoritmih), čeprav pri tem obstaja večja nevarnost nepredvidljivega delovanja.« (Wikipedija – Nevronska mreža)

Prvo idejo o delovanju umetne nevronske mreže sta leta 1943 podala nevrofiziolog Warren McCulloch in mladi matematik Walter Pitts. Napisala sta članek in tudi izdelala preprost elektronski model umetne nevronske mreže (Haykin, 1994) .

Walter Pitts



Warren McCulloch

Leta 1949 je Donald Hebb napisal knjigo Organizacija védenja, v kateri je izpostavil dejstvo, da se poti signala skozi mrežo okrepijo vsakič, ko jih uporabimo. Kot bomo videli, je to dejstvo zelo pomembno za učenje mreže.

V petdesetih letih 20. stoletja je razvoj računalnikov nekako zasenčil raziskave umetnih nevronskih mrež. Vendar je bilo mogoče s pomočjo računalnikov simulirati delovanje, ne da bi bilo treba sestaviti fizični model mreže.

John von Neumann si je zamislil imitacijo preproste nevronske mreže z uporabo telegrafskih relejev ali vakumskih cevi. To je vodilo do izuma von Neumannovega stroja.

Frank Rosenblatt je leta 1958 izumil perceptron, ki je bil prva »praktična« umetna nevronska mreža. Ideja perceptrona temelji na raziskavah muhinega očesa. Oko namreč muhi sporoči, da naj beži, ko je nevarnost dovolj blizu. Žal sta leta 1969 Marvin Minsky in Seymour Papert v knjigi z naslovom Perceptroni dokazala omejene zmožnosti te vrste mrež.

Leta 1959 sta Bernard Wildrow in Marcian Hoff razvila sistem »ADALINE«, kar pomeni ADAptivni LINearni Elementi. To je bila prva umetna nevronska mreža, ki se je lahko soočala s praktičnimi problemi; odpravljala je namreč odmeve v telefonskih zvezah. Minsky in Papert sta spet pokazala omejene zmožnosti mreže.

V sedemdesetih letih so nevronskim mrežam posvečali veliko pozornosti mediji, ki so v glavnem napihovali resnična dejstva. Ljudje so začenjali verjeti, da bodo stroji znali opravljati vse človeško delo, čemur je seveda sledilo razočaranje. Ti dejavniki so dejansko pripeljali do velikih kritik umetne inteligence in so celo povzročili zmanjšanje financiranja raziskav.

John Hopfield

Leta 1982 je John Hopfield znanstveni javnosti predstavil članek, v katerem je predlagal, naj se umetna inteligenca ne ukvarja samo z oponašanjem človeških možganov, ampak naj se uporabi za reševanje dinamičnih problemov. Predstavil je tudi načrte za take stroje. Njegov članek, prijeten značaj in ogromno znanje matematične analize so prepričali člane narodne akademije znanosti k obnovi raziskav umetne inteligence. Iz njegovih idej pa se je s časom razvila tudi svoja vrsta nevronskih mrež, ki jo imenujemo Hopfieldov model (Haykin, 1994).

Istočasno so japonski znanstveniki objavili, da se bodo posvetili raziskovanju zmožnosti nevronskih mrež, kar je spodbudilo Združene države Amerike, da so takoj začele financirati nadaljnje raziskave.

Prve konference Nevronske mreže za računanje se je leta 1986 udeležilo več kot 1800 delegatov. Istega leta so Rumelhart, Hinton in Williams poročali o razvoju

algoritma »back-propagaton«, ki je postal najbolj priljubljen način za učenje večslojnih mrež.

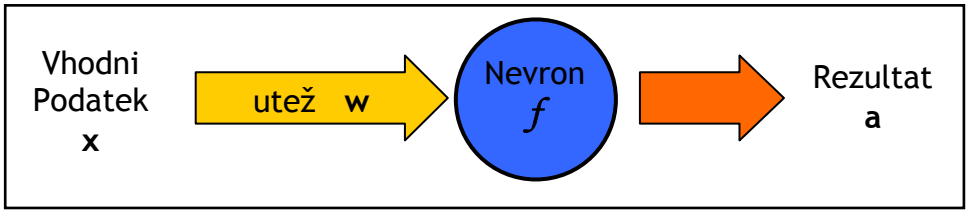
Z začetkom devetdesetih let so po vsem svetu opazni napredki v raziskovanju umetnih nevronskih mrež. Tudi sama narava je dokaz, da nevronske mreže v resnici delujejo. Danes je glavni izziv elektronska implementacija tehnologije nevronskih mrež. Trudijo se z izdelavo treh vrst nevro-čipov: digitalnih, analognih in optičnih. Če se bo z njimi res dalo izdelati nevronske mreže, se tehnologiji obeta svetla prihodnost.

3.2 Osnovni princip nevronskih mrež

Nevronska mreža je računalniška simulacija delovanja možganov. Celotna ideja nevronskih mrež temelji na sestavljanju enostavnih objektov »nevronov« v velike sisteme. Nevron je osnovna celica nevronske mreže. Predstavljamo si ga kot element, ki dobi nekaj vhodnih podatkov ter vrne en izhodni podatek. Izhodni podatek je hkrati lahko vhodni za nek drugi nevron. S takim principom iz nevronov sestavimo nevronsko mrežo, ki naj bi služila določenemu namenu. V tem poglavju večino resnic povzemam po priročniku za uporabo *Matlabove* orodjarne *Neural network toolbox* (Demuth et al., 2009).

3.2.1 Model nevrona

Začnimo s sliko



Slika 3.1: Osnovni model nevrona

Nevron na sliki ima samo en vhod in en izhod. Nevroni imajo lahko več vhodov, vedno pa le en izhod. Vsak vhod v nevron ima svojo utež, vsak nevron pa ima tudi določen predpis (funkcija prehoda).

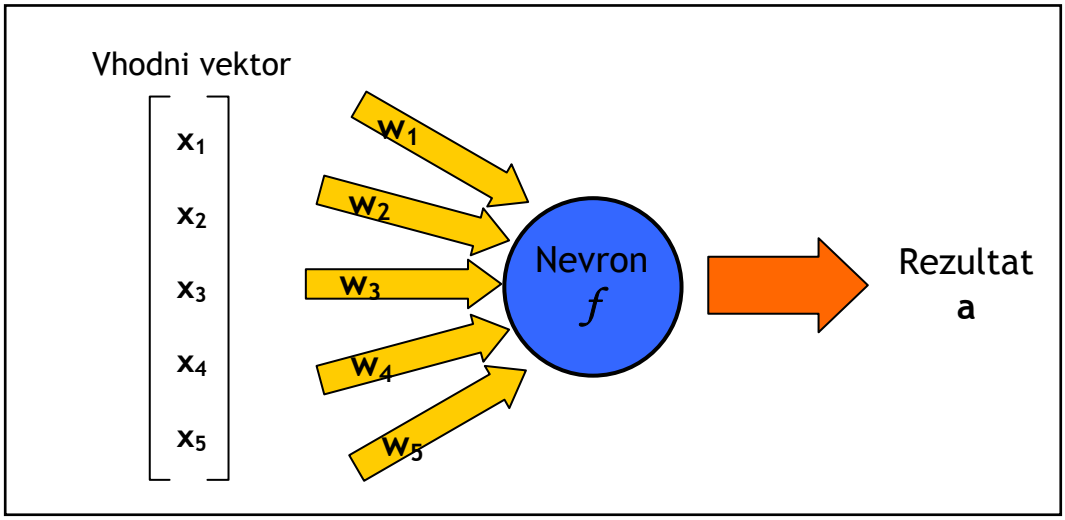
Osnovna ideja je, da se vhodni podatek pomnoži z utežjo svojega vhoda. Funkcija f sprejme produkt in skozi izhod vrne rezultat a.

Z enačbo to zapišemo:

$$a = f(x \cdot w) \tag{1}$$

Pri nevronu z enim vhodom velja, da so *a, x* in *w* realna števila.

Ponavadi pa imajo nevroni več kot en vhod. Takrat stvar izgleda tako, kot vidimo naslednji sliki.



Slika 3.2: Model nevrona z več vhodi

Vhodne podatke lahko organiziramo v vektor. Vsak od podatkov se pomnoži z utežjo svojega vhoda. Vsi produkti se seštejejo in vsota vstopi v funkcijo prehoda; ta nato vrne rezultat

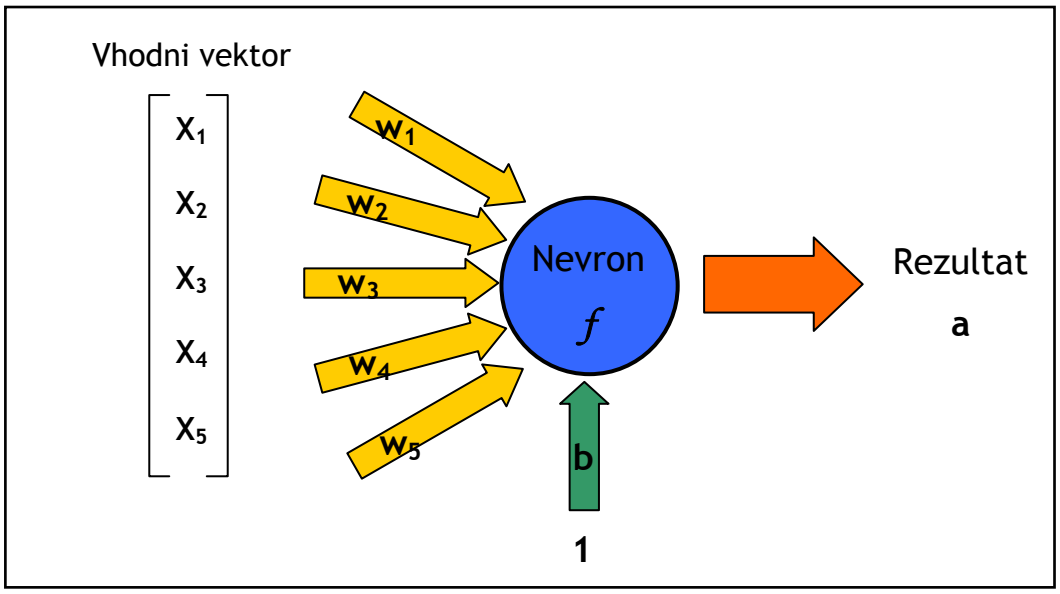
$$a = f\left(\sum_{i=1}^{n} x_i \cdot w_i\right) \tag{2}$$

kjer so *a*, *x* in *w*, realna števila. V matrični obliki je enačba podobna, saj je skalarni produkt definiran kot seštevanje produktov komponet.

$$a = f\left(x \cdot w\right) \tag{3}$$

Tokrat sta *x* in *w* vektorja dimenzije $1 \times n$, ki je število komponent vhodnega vektorja; *a* je realno število. Ker je tudi skalarni produkt realno število, funkcija f slika iz realnih v realna števila.

Ponavadi imajo nevroni še stranski vhod, skozi katerega vedno vstopi vrednost ena. Ta vhod oziroma utež tega vhoda imenujemo prag nevrona. Prag predstavlja vrednost, ki bo v vsakem primeru prišla v nevron, tudi če skozi druge vhode ne bo prišlo nič. Uporabnost praga si bomo ogledali kasneje.



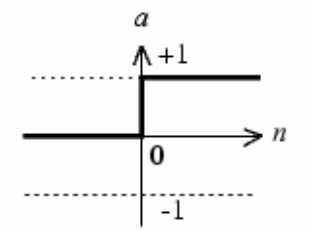
Slika 3.3: Model nevrona z utežjo

3.2.2 Funkcije prehoda

Funkcije prehoda pomenijo odločitev, kakšno vrednost naj vrne nevron v odvisnosti od vhodnih podatkov. Produkt vhodnega vektorja z vektorjem uteži na vhodih bomo imenovali vhodna vrednost.

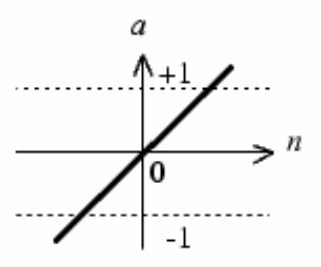
Za funkcijo prehoda se lahko izberejo različne funkcije, odvisno od tega, za kaj potrebujemo mrežo. Najpogostejše možnosti so naslednje:

- **Funkcija prehoda s trdo mejo** pomeni, da nevron vrne vrednost ena, če je vhodna vrednost večja od praga, sicer vrne vrednost nič.



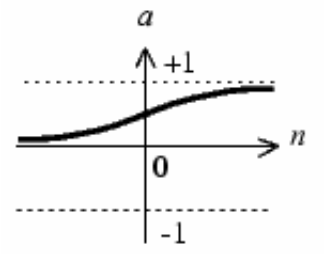
Graf funkcije prehoda s trdo mejo (Demuth et al., 2009)

- **Linearna funkcija prehoda** zagotavlja, da je velikost izhoda iz nevrona linearno sorazmerna velikosti vhoda.



Graf linearne funkcije prehoda (Demuth et al., 2009)

- **Sigmoidna funkcija prehoda** predstavlja krivuljo, ki je nekako srednja pot med trdo mejo in linearno funkcijo.



Graf sigmoidne funkcije prehoda (Demuth et al., 2009)

Za majhne vhodne vrednosti bo izhod blizu nič, za velike pa blizu ena, kar je podobno funkciji trde meje, vendar pa prehod ni diskreten ampak zvezen. Najbolj pogosto se uporabljata funkciji:

$$\sigma(x) = \frac{1}{1+e^{x}} \qquad \text{in} \qquad \sigma(x) = \tanh(x)$$

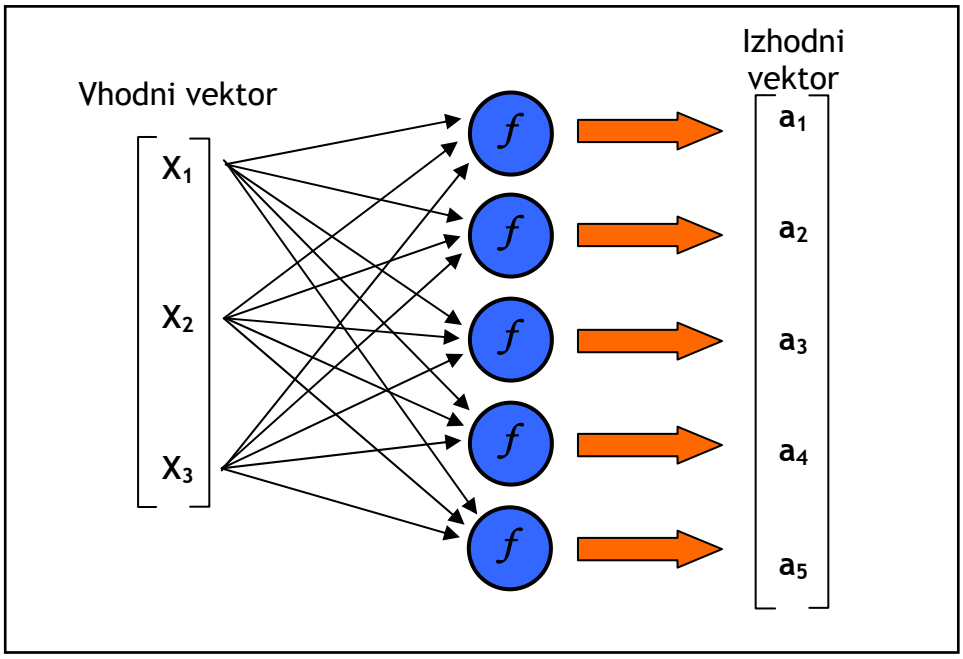
Funkciji se da prilagoditi tako, da ju premikamo levo in desno ali pa jima spreminjamo naklon (Dobnikar, 1990).

Obstaja še nekaj drugih prehodnih funkcij, vendar ne bodo opisane. V našem primeru gre namreč za tekmovalno učenje, kjer vrednost izhoda ni odvisna od funkcije prehoda, ampak od tega, kateri nevron je izračunal največji vhod.
Pri razlagi delovanja nevrona je uporabljen izraz vsota produktov podatkov z utežmi. Pri dejanskem računanju gre za skalarni produkt dveh vektorjev, pri

čemer je treba samo paziti, da so vrednosti vhodnih podatkov in uteži smiselno urejene.

3.2.3 Arhitektura mreže

Dva ali več predstavljenih nevronov lahko povežemo skupaj in tako formiramo nevronsko mrežo.



Slika 3.4: Model enoslojne umetne nevronske mreže

Na zgornji sliki je predstavljen en sloj nevronske mreže. Seveda je lahko velikost vhodnih vektorjev poljubna, prav tako je poljubno tudi število nevronov v enem sloju; izhodni vektor je po velikosti enak številu nevronov.

V sloju nevronov ima še vedno vsak nevron svoje uteži, tako da v primeru na sliki obstaja petnajst uteži (za vsako puščico ena).

V primeru sloja nevronov uteži organiziramo v matriko uteži:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \tag{4}$$

Tu vsaka vrstica predstavlja svoj nevron, vsak stolpec pa svoj element v vhodnem vektorju. Matrika je velikosti $m \times n$, kjer je m število nevronov v sloju, n pa število podatkov v vektorju.

Tako pri vhodnem vektorju p in vektorju pragov b:

$$p = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{5, 6}$$

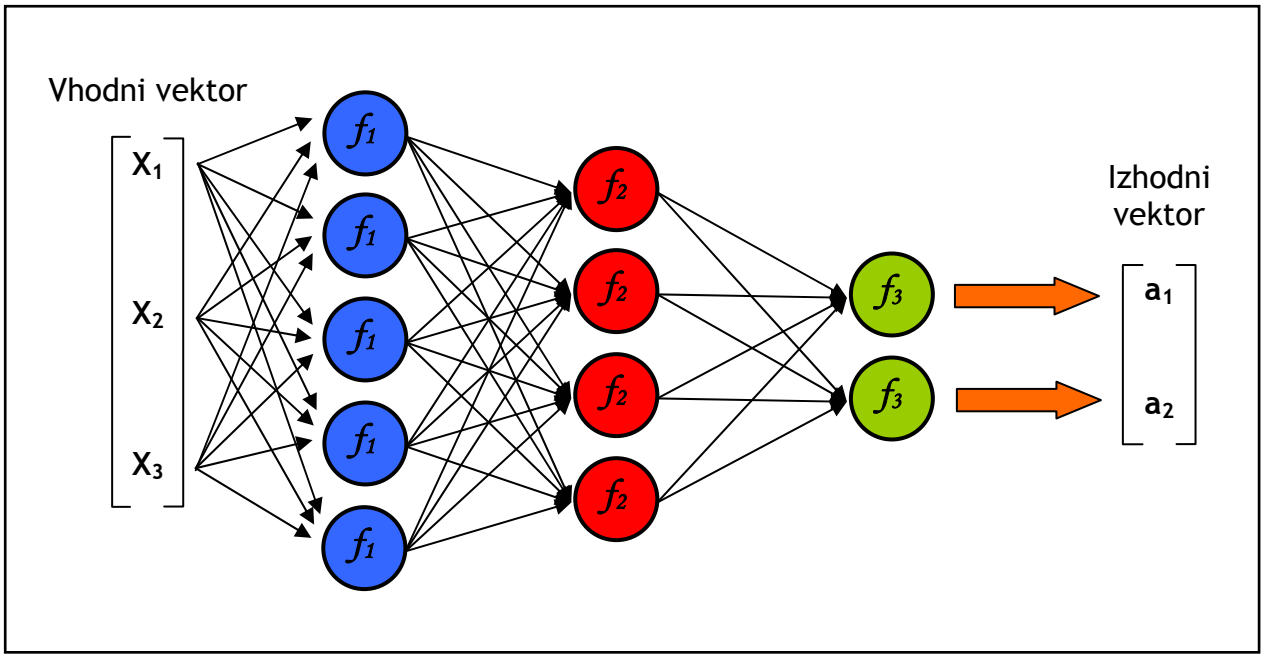
izračun s slojem nevronov lahko zapišemo z enačbo

$$r = f\left(W \cdot p + b\right) \tag{7}$$

kjer $f : R^m \to R^m$. Rezultat je vektor *r*.

$$r = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{8}$$

Seveda je mogoče zaporedno sestavljati tudi sloje nevronov. V takem primeru se izhodni vektor *r* uporabi kot vhodni v naslednjem sloju.



Slika 3.5: Model večslojne umetne nevronske mreže

Pri delu z več sloji moramo začeti uporabljati še indekse slojev. Za vsak sloj morajo biti znani vhodni vektor, vektor pragov, utežna matrika ter funkcija prehoda. Izhodni podatki nekega sloja so hkrati vhodni podatki naslednjega. Navadno se prvi sloj nevronov imenuje vhodni nivo, zadnji sloj pa izhodni nivo. Vse vmesne sloje s skupno besedo imenujemo skriti sloj (Demuth et al., 2009).

3.2.4 Učenje

Nevronska mreža sama po sebi ni uporabno orodje, dokler z njo ne moremo reševati želenih problemov. V zgoraj opisanih modelih smo videli, da moramo za delovanje mreže poznati uteži in funkcijo prehoda za vsak nevron.

Funkcije prehoda se izberejo glede na vrsto rabe mreže. V naslednjem poglavju si bomo ogledali vrste nevronskih mrež in videli, da ima vsaka vrsta določeno, katere funkcije prehoda se uporabljajo na katerem od nivojev.

Določitev uteži nevronom pa je bolj kompleksen postopek, ki ga imenujemo učenje nevronske mreže. Učenje je postopek, pri katerem, ponavadi v več korakih, izučimo nevronsko mrežo, tako da je sposobna rešiti konkretni problem.

Obstajata dva glavna principa za učenje nevronske mreže, to sta nadzorovano in nenadzorovano učenje (angl. supervised, unsupervised training).

Pri nadzorovanem učenju mreži podamo niz vhodnih podatkov (vektorjev), poleg tega pa še želeni rezultat za vsak vektor. Mreža tako procesira vhodni niz z naključnimi utežmi ter dobljeni rezultat primerja z želenim; glede na razliko med rezultatoma posodobi uteži. Postopek se ponavlja z vsemi podatki toliko časa, da so razlike pod dovoljeno mejo. Tako naučena mreža zna potem izračunati rezultat tudi za podatke, ki jih med učenjem nismo uporabili. Kadar nevronsko mrežo želimo uporabiti za razvrščanje objektov v skupine, bomo morali praviloma uporabiti mrežo z nenadzorovanim učenjem (Kangas in Kohonen, 1996).

Pri nenadzorovanem učenju mreži podamo samo vhodne podatke brez želenih rezultatov. Tako mora mreža sama poiskati značilnosti podatkov, po katerih lahko loči podatke med seboj.

Učilno pravilo

Učilno pravilo je funkcija, ki določa način učenja mreže. Rezultat funkcije učenja so nove uteži, s katerimi zamenjamo stare. Nekatera učilna pravila določijo spremembe za uteži vseh nevronov v mreži, druga samo za določene skupine nevronov, nekatera pa samo za en nevron v vsakem koraku učenja.

Vhodni podatki funkcije učenja so v splošnem:

- stare uteži nevronov,
- učilni parameter,
- dejanski in želeni rezultat mreže.

Funkcija je navadno zapisana v obliki

$$\omega_{nove} = \omega_{stare} + \lambda \cdot f(x, r) \tag{9}$$

kjer je ω znak za uteži, λ za učilni parameter, x in r pa vektorja vhodnih podatkov in rezultatov.

Učilni parameter je parameter, ki določa, koliko naj se med potekom učenja spreminjajo uteži. Učilni parameter se mora načeloma manjšati od ena proti nič, kar pomeni, da se uteži na začetku učenja zelo spreminjajo, potem pa vedno manj.

Kaj več o učilnem pravilu na splošno je težko povedati, ker so ta pravila ena od bistvenih specifik različnih vrst mrež. V naslednjem poglavju bomo za vsako vrsto mrež videli njene bistvene značilnosti in v čem se razlikuje od drugih.

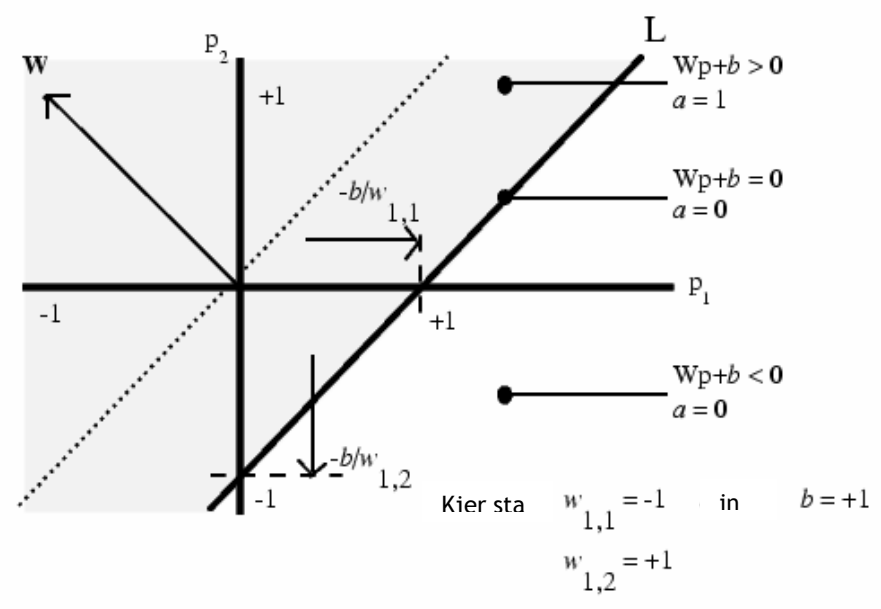
3.3 Vrste nevronskih mrež

Nevronske mreže se med seboj razlikujejo po obliki, velikosti, številu slojev, funkciji prehoda nevronov, učilnem pravilu in drugih specifičnih lastnostih. Konec koncev si različne vrste mrež sploh niso podobne, razen po osnovnem modelu nevrona in pravilih za povezovanje nevronov med seboj.

Rad bi predstavil glavne vrste nevronskih mrež in njihovo uporabnost, vendar je to nekoliko težje, kot sem pričakoval. Različni viri povsem različno klasificirajo vrste mrež. Glavna vira sta bila Dobnikarjev učbenik o nevronskih mrežah ter navodila za uporabo *Matlabove* orodjarne *Neural network toolbox*. V učbeniku poglavja niso organizirana po različnih vrstah mrež, ampak po smiselnem zaporedju tem, ki so pomembne za študij nevronskih mrež; v drugem viru pa so opisana samo orodja, ki jih pozna Matlab, in njihovo ozadje. Sledi izbor najbolj bistvenih vrst mrež.

Perceptron

Perceptron je najbolj osnovna verzija nevronske mreže. Mreža peceptrona je sestavljena iz enega sloja nevronov, ki imajo funkcijo prehoda s trdo mejo. Če je torej vhod v nevron večji od vrednosti praga, vrne nevron vrednost ena, sicer pa vrednost nič. Sloj več nevronov nam torej vrne rezultat v binarni obliki (npr. [0, 0, 1, 0, 1]).
Za primer dveh vhodov lahko ponazorimo nevron perceptrona kot operator, ki vhodne vrednosti dodeli v eno od dveh polravnin prostora vhodnih podatkov.



Polravnini kot rezultat Perceptrona (Demuth et al., 2009)

Na sliki vektor *W* predstavlja vektor uteži na vhodih nevrona, *b* je prag nevrona, premica *L* pa delilka ravnine. Za večdimenzionalne podatke nevron razdeli hiperprostor v dva polprostora. Perceptron se uporablja za enostavne *klasifikacije* (ne grozdenje) vzorcev.

Učilno pravilo perceptrona je precej preprosto. Obstajajo trije možni primeri:

- ✓ *Nevron vrne vrednost, ki je enaka želeni.*
  Uteži nevrona naj se v tem primeru ne spremenijo.

- ✓ *Nevron vrne vrednost 0, moral pa bi vrniti vrednost 1.*
  Utežem nevrona prištejemo vhodni vektor; tako bo naslednjič vhod v nevron za tak vektor večji in s tem se približamo možnosti, da bo rezultat tak, kot je želeno, to je ena.

- ✓ *Nevron vrne vrednost 1, moral pa bi vrniti vrednost 0.*
  Utežem tokrat odštejemo vhodni vektor; tako se vhod v nevron zmanjša in verjetnost za rezultat nič povečamo.

Z enačbo to zapišemo:

$$\Delta\omega = (t - a) \cdot p$$

kjer je $\Delta\omega$ popravek uteži, t želeni rezultat, a trenutni (dobljeni) rezultat nevrona in p vhodni vektor (Demuth et al., 2009).

Kadar sta želeni in trenutni rezultat enaka, je popravek uteži enak nič, saj nevron že pravilno deluje. Če je želena vrednost ena, dobimo pa nič je popravek uteži kar enak vhodnemu vektorju, kar povzroči, da bo naslednjič pri podobnem vektorju vhod v funkcijo prehoda večji. Tako se bo povečala možnost za rezultat ena. Če pa je želena vrednost nič, dobimo pa ena, so popravki uteži enaki negativnemu vhodnemu vektorju. Tako bo naslednjič s podobnim vektorjem vhod v funkcijo prehoda manjši, verjetnost, da funkcija vrne nič, pa večja.

Med učenjem torej premikamo premico (v primeru prostora ravnino), ki je na zgornji sliki označena z *L*. S spreminjanjem uteži premico rotiramo, za translacijo pa moramo popraviti prag.

Linearni filtri

Linearne mreže so zelo podobne perceptronu, le da so funkcije prehodov linearne. Izhodne vrednosti mreže imajo lahko kakršnokoli vrednost in nič več samo nič in ena. Podobno kot perceptron so linearni filtri sposobni rešiti linearno razčlenljive probleme.

Postopek učenja je lahko nekoliko drugačen kot pri perceptronu. Definirajmo razliko med izračunanim in želenim izhodnim vektorjem kot napako. S spreminjanjem uteži in pragov želimo doseči stanje, ko bo vsota kvadratov napak kar najmanjša. Ta problem je obvladljiv, ker imajo linearni sistemi samo eno minimalno vrednost napake. V večini primerov je mogoče izračunati rešitve direktno, včasih pa je zaradi numeričnih problemov taka rešitev nemogoča. Na

srečo lahko vedno uporabimo učilni algoritem najmanjših kvadratov oziroma Widrow-Hoffov algoritem (Demuth et al., 2009).

Večnivojske mreže (Backpropagation)

Večnivojske mreže so mreže, sestavljene iz več slojev nevronov. Izum večnivojskih mrež je rešil enega od osnovnih problemov v začetkih tehnologije nevronskih mrež. Vsak računski sistem bi moral biti sposoben izvesti osnovne logične operacije NOT, AND, OR in EX-OR. Prve tri niso bile problematične, zadnjo pa je sposobna izvesti samo večnivojska mreža. Gre za logično operacijo *ekskluzivni ali*, ki odgovori z ena, če sta vhodni trditvi različni in z nič če sta enaki.

Ena od ključnih besed, ki se uporablja v povezavi z večnivojskimi mrežami, je *backpropagation*. *Backpropagation* je v bistvu princip učilnega pravila, ki pravi, da se morajo večnivojske mreže vedno učiti od končnega proti začetnemu sloju. Najprej adaptiramo uteži in pragove nevronom iz zadnjega sloja in šele na podlagi te posodobitve lahko določimo popravke za en sloj prej.

Uporabnost večnivojske mreže je, da lahko za razliko od perceptrona, ki definira dve polravnini, takšna mreža zagotavlja poljubna območja. Dva nivoja delita ravnino na pol, vendar ne nujno s premico, medtem ko trije nivoji že lahko oblikujejo otoke znotraj območja. (Dobnikar, 1990)

Sočasno razširljiva mreža (Counterpropagation)

»Sočasno razširljiva mreža je po uporabnosti zelo podobna Backpropagation učnemu pravilu, le da ni tako splošna, ima pa bistveno krajši učilni čas.« (Dobnikar, 1990; 53)

SRM mreža je sestavljena iz dveh plasti. Prvi sloj je samoorganizirajoča Kohonenova mreža, drugi pa zvezdasta mreža Grossberga. Princip dveh slojev spominja na strukturo možganov, kjer različne plasti opravljajo različne funkcije.

Prvi sloj mreže razvršča podatke v skupine tako, da na podobne vzorce vedno odgovarja isti nevron, medtem ko drugi sloj klasificira podatke, da glede na rezultat prvega sloja producira želeni rezultat. Učenje drugega sloja zahteva poleg vhodnih podatkov tudi želene izhodne podatke. (Dobnikar, 1990)

Hopfieldove nevronske mreže

Hopfieldove nevronske mreže je skupno ime za rekurentne mreže. To pomeni, da imajo mreže iz tega sklopa privzeto povratno povezavo, tako da izhod iz mreže vedno ponovno vstopa na vhod. Toliko, kolikor ima mreža nevronov, mora imeti vsak nevron vhodov, saj izhod iz vsakega nevrona vstopa v vse nevrone.

Računanje izhodnih vrednosti se torej ponavlja, rezultat mreže pa je odvisen od njene stabilnosti. Stabilna mreža bo privedla do tega, da se bodo vrednosti na izhodih vedno manj spreminjale in se bo nazadnje vzpostavilo ravnotežje. Nestabilne mreže na drugi strani pa predstavljajo niz računanja, ki se nikoli ne konča. Takšne mreže imajo lastnosti kaotičnih sistemov.

Nestabilne mreže so na začetku veljale za neuporabne, danes pa smo priča tudi raziskavam kaotičnega obnašanja mrež.

Dvosmerne asociativne mreže

Dvosmerne asociativne mreže so podobne Hopfieldovim. Zaradi lastnosti hetero-asociativnosti te mreže vedno vračajo izhodne vrednosti na drugem nivoju nevronov kot sprejemajo vhodne. Mreža je sposobna posploševanja in zna iz niza podatkov s šumom razbrati čisti signal. (Dobnikar, 1990)

Samoorganizirajoče se mreže

Na področju samoorganizirajočih se mrež je bil vodilni Kohonen, ki ni samo teoretično izumil principa, ampak ga je tudi praktično izvedel.

Samoorganizirajoče se mreže imajo lastnost nenadzorovanega učenja, kar pomeni, da za učenje ne potrebujejo niza želenih rešitev. Mreža uporablja tekmovalna učenja po principu »winner takes all« oziroma »zmagovalec dobi vse«. (Kohonen, 1998)

Mreža vhodne podatke razvršča v eno- ali dvodimenzionalno mrežo nevronov po načelu, da so podobni podatki razvrščeni v nevrone, ki so si blizu med seboj.

Več govora o samoorganizirajočih se mrežah bo v poglavju o tekmovalnem učenju.

Adaptivna resonančna teorija

Večina mrež, ki smo si jih ogledali, ima v primerjavi z možgani veliko pomanjkljivost; ne morejo si namreč zapomniti naučenih vzorcev ter se hkrati učiti novih. Navadno učenje novih vzorcev pomeni modifikacijo ali kar brisanje starih. Ta problem rešujejo mreže ART.

Mreža ART je namenjena klasifikaciji vhodnih podatkov, sestavljena pa je, kot pravi Dobnikar, iz dveh nivojev, razpoznavnega in primerjalnega. Za vsak vhodni vzorec se najprej poišče shranjeni vzorec. Če sta si vzorca dovolj podobna, se vhodni vzorec pripiše tej skupini, hkrati pa se shranjeni vzorec adaptira kot pri tekmovalnem učenju (glej naslednje poglavje). Če sta si vzorca manj podobna od določne tolerance, se formira nova skupina. (Carpenter in Grossberg, 2002)

Cognitron

Cognitron je nevronska mreža, ki poskuša ponazoriti človekov sistem za dojemanje vizualnih informacij. Mreža je sestavljena iz dveh slojev nevronov, ki so med seboj povezani s sinapsami, zato ju imenujemo *predsinaptični* in *posinaptični* sloj.

Mreža je samoučeča in uporablja princip, ki je zelo podoben tekmovalnemu učenju, le da pri *cognitronu* obstajajo določena dodatna pravila.

V posamezni posinaptični nevron ne prihajajo signali iz vseh predsinaptičnih nevronov, ampak le iz tistih iz bližnje okolice (podobno so sestavljeni možgani). Tako ima vsak posinaptični nevron svoje vplivno okolje, ki je sestavljeno iz njemu bližnjih predsinaptičnih nevronov. Vsak posinaptični nevron pa ima tudi svoje tekmovalno okolje (sosedje v njegovem sloju). Nevroni ne tekmujejo z vsemi drugimi, ampak le s svojimi sosedi.

V predsinaptičnem sloju so pospeševalni in zaviralni nevroni. Pospeševalni nevroni težijo k vžigu posinaptičnega nevrona, zaviralni pa ravno obratno. Tisti nevron, v posinaptičnem sloju, ki vžge, je tudi nagrajen. Poleg nagrade za zmagovalca poznamo tudi kazen za poražence.

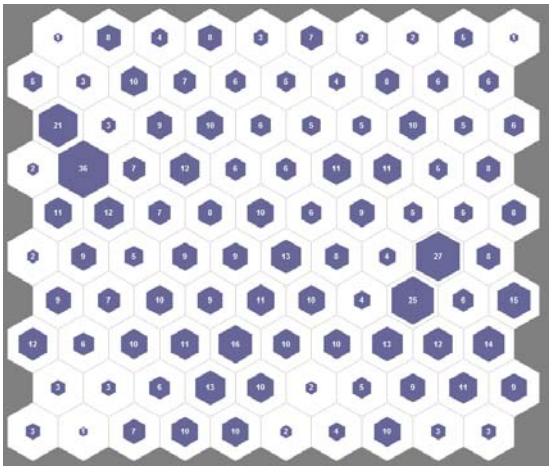
Cognitron je sposoben prepoznavanja vzorcev z neverjetno natančnostjo. (Dobnikar, 1991)

3.4 Tekmovalno učenje

Tekmovalno učenje je princip učenja, ki se sicer uporablja v Kohonenovih samoorganizirajočih se mrežah *SOM* (angl. Self organizing map). Mreža *SOM* je sloj nevronov, ki se učijo s tekmovalnim učenjem, poleg tega pa so sestavljeni tako, da njihove medsebojne lege predstavljajo topologijo prostora vhodnih podatkov. Uteži posameznih nevronov se zvezno spreminjajo, če zvezno potujemo po mreži. Tako se bližnji nevroni javljajo pri podobnih podatkih.

Rezultat mreže *SOM* je dvodimenzionalna mreža nevronov ter podatek, v katerega od nevronov spada posamezni objekt. Objekti, ki spadajo v isti nevron, so si zelo podobni. Čimbolj sta si dva nevrona med seboj oddaljena, tem manj so si podobni objekti, ki jima pripadajo. Navadno se v mreži formirajo zgoščine oziroma območja nevronov, ki pritegnejo več objektov, ter območja, ki jih pritegnejo manj. Pri analizi rezultata bi bila za naš problem spet potrebna odločitev, kje potegniti meje med zgoščinami.

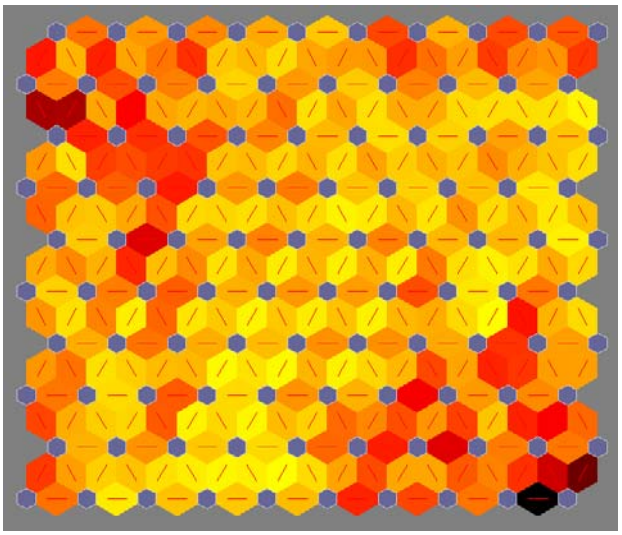
Primer rezultata, ki bi ga vrnila mreža *SOM*, prikazuje slika 3.6.



Slika 3.6: Število objektov, ki jih sprejme nevron v mreži SOM

Nekateri nevroni sprejmejo več vektorjev kot drugi. S slike bi bilo zelo težko razbrati, v koliko skupin so se organizirali vektorji in kje potegniti meje med njimi.

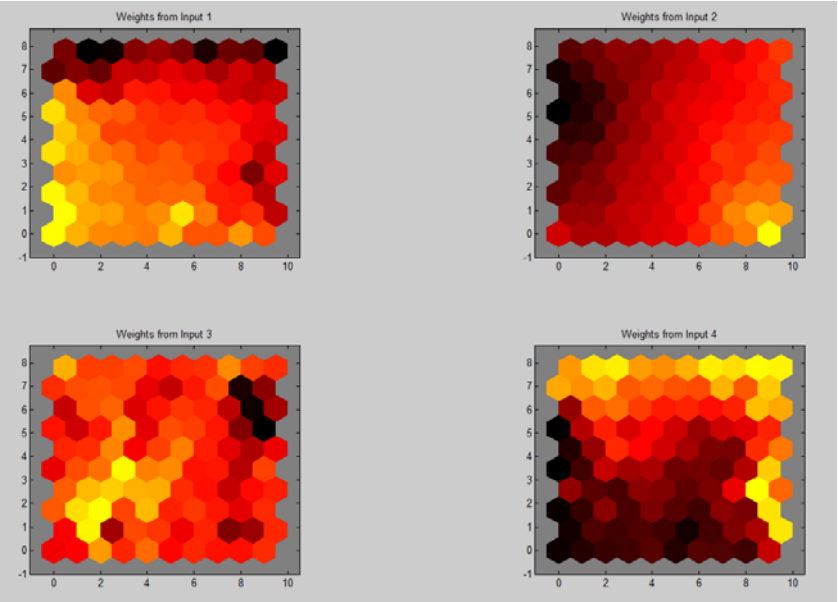
Naslednja slika prikazuje »bližino« nevronov, ki so v mreži sosedje, temnejše povezave pa pomenijo večjo bližino.



Slika 3.7: Bližina nevronov v mreži SOM

Tretja predstavitev, ki jo ponuja mreža *SOM*, je porazdelitev uteži posamezne komponente po mreži. Vsak od grafov prikazuje porazdeljevanje ene od štirih komponent vhodnih podatkov (x, y, dx in dy). Prvi dve komponenti pomenita

položaj točke, drugi dve pa sta pravokotni komponenti njene hitrosti, toda več o tem pri predstavitvi podatkov.



Slika 3.8: Porazdelitev uteži za posamezno komponento

Nevron v zgornjem desnem vogalu mreže bi na primer ustrezal vektorjem z zelo velikim x, srednje velikima y in dx, ter z zelo majhnim dy.

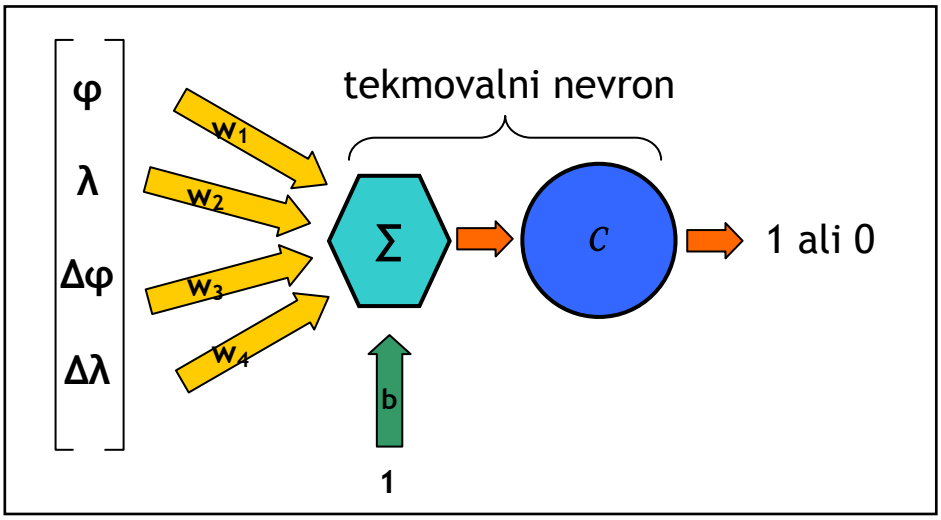
Mreža *SOM* ima gotovo zelo veliko moč pri obdelavi »naravoslovnih« podatkov, pri katerih se lastnosti objektov zvezno spreminjajo. Tak primer so nedvomno tudi premiki litosferskih objektov, saj plošče niso toga ampak elastična telesa. Kakorkoli, za rešitev našega problema, to je določitev mej, mora biti odločitev popolnoma enopomenska oziroma nedvoumna in diskretna. Vsak vektor lahko obenem pa tudi mora pripadati samo enemu območju.

Ker potrebujemo enopomenski oziroma nedvoumen odgovor, v katero skupino spada posamezni vektor, sem se odločil uporabiti načelo tekmovalnega učenja brez razporeditve nevronov v topološko urejeno mrežo. Nevroni v tem načinu delujejo vsak zase in njihov položaj ne vpliva na podobnosti njihovih uteži. Tako lahko vsaj trdimo, da so si objekti, ki padejo v isti nevron, podobni, tisti, ki padejo v drugega, pa so drugačni od njih.

3.4.1 Osnovno načelo tekmovalnega učenja

Nevroni v tekmovalnem sloju se med učenjem prilagodijo tako, da vsak od njih ustreza določenemu vzorcu vhodnih podatkov.

Na sliki vidimo primer poljubnega nevrona tekmovalnega sloja, na njegovem vhodu pa podatke enega objekta (vektorja premika). Lastnosti nevrona so vrednosti uteži na vhodih, vrednost praga in tekmovalna funkcija prehoda, ki je označena s *c* (competitive).



Slika 3.11: Model tekmovalnega nevrona

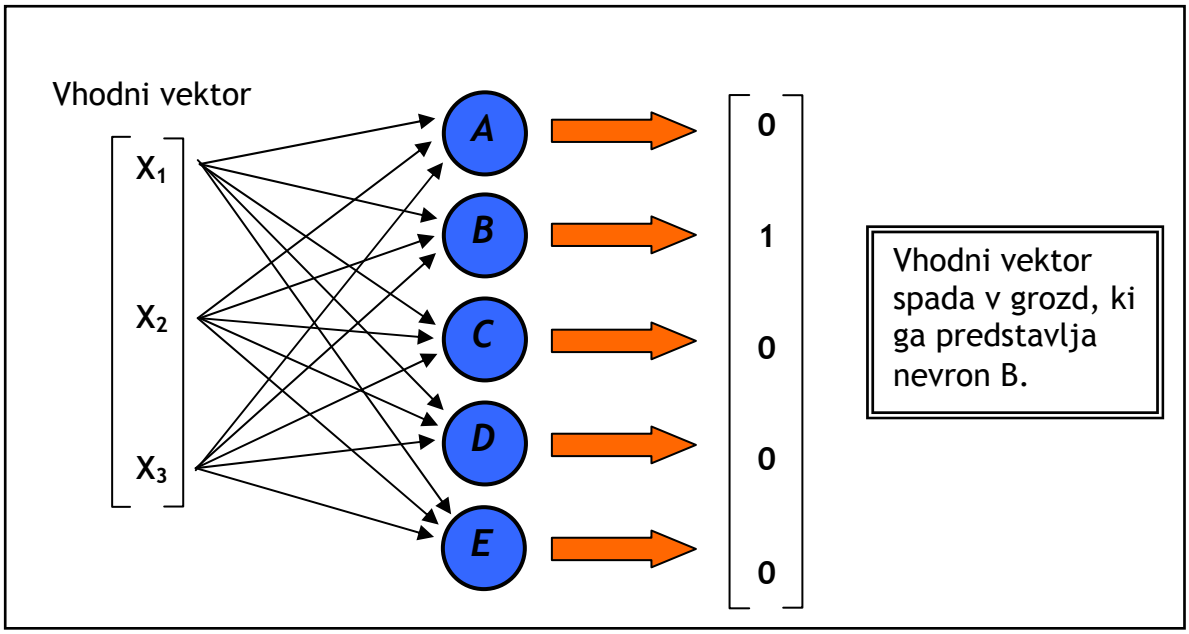
Vhodni podatki so štiri vrednosti vektorja premika. Prvi podatek (φ) se pomnoži s prvo utežjo (w_1), drugi z drugo in tako naprej, nato pa se vsi produkti seštejejo (na sliki znak Σ). Prišteje se jim tudi vrednost praga b, toda več o tem kasneje. Velikost vsote produktov se primerja z vsotami v drugih nevronih sloja. Tisti nevron, ki ima največjo vsoto, vrne vrednost ena, vsi drugi pa vrednost nič.

Če so uteži nevrona podobne vhodnemu vektorju, potem je skalarni produkt (vsota produktov členov) večji, drugače pa je manjši. Tako bo torej nevron vedno zmagoval, kadar se bodo na vhodu pojavili vektorji, ki so podobni njegovim utežem.

Nujen pogoj pri tem postopku je, da so vhodni podatki normirani. To pomeni, da je srednja vrednost vsake od koordinat nič in standardni odklon enak ena. Več o tem bomo izvedeli pri pripravi podatkov. Obstaja tudi možnost, da namesto vektorskih produktov računamo negativne razdalje med vektorjem podatkov in vektorjem uteži v hiperprostoru. Na voljo imamo vse vrste razdalj, ki so opisane v drugem poglavju.

Rezultat skalarnega produkta je tem večji, čim bolj podobna sta si faktorja (vhodni vektor in vektor uteži nevrona). To velja samo v primeru, kadar so vektorji, ki jih uporabljamo, normirani. Sicer ima vpliv tudi sama absolutna vrednost vektorja.

Postopek z negativnimi dolžinami pomeni, da se izračunajo razdalje med točkami, ki jih predstavljajo vektorji v hiperprostoru. Vektorji, ki so si podobni, realizirajo točke, ki so blizu skupaj. Ker med seboj zelo oddaljeni vektorji vrnejo največji rezultat, moramo uporabiti negativno vrednost dolžin.
Ko je mreža enkrat naučena (postopek učenja bomo še pogledali), je postopek izračuna preprost. Zaporedoma pripeljemo na vhod mreže vsakega od vektorjev podatkov ter pri vsakem opazujemo, kateri od nevronov je edini vrnil vrednost ena. Nevron nam sedaj predstavlja grozd oziroma skupino podobnih vektorjev. Lahko rečemo, da vsak vektor pripada enemu nevronu, tako kot pripada eni skupini.



Slika 3.12: Model tekmovalnega sloja

Pri matematičnem opisu postopka bo poenostavljeno, da so vhodni podatki dimenzije 4, kakor so v našem primeru, drugače pa so lahko poljubne velikosti.

Naj bo *W* matrika uteži nevronov

$$W = \begin{bmatrix} \omega_{1,1} & \omega_{2,2} & \omega_{3,3} & \omega_{4,4} \\ \omega_{2,1} & \omega_{2,2} & \omega_{2,3} & \omega_{2,4} \\ \vdots & \vdots & \vdots & \vdots \\ \omega_{m,1} & \omega_{m,2} & \omega_{m,3} & \omega_{m,4} \end{bmatrix} \tag{10}$$

Vsaka vrstica predstavlja svoj nevron, ki ima štiri uteži. Nevronov je *m*, uteži pa so štiri, za vsakega od podatkov (φ, λ, Δφ, Δλ) eden.

Vektorji podatkov so organizirani v matriko *x*. Vsak stolpec predstavlja svoj vektor, vseh vektorjev pa je *n*.

$$x = \begin{bmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_n \\ \lambda_1 & \lambda_2 & \dots & \lambda_n \\ \Delta\varphi_1 & \Delta\varphi_2 & \dots & \Delta\varphi_n \\ \Delta\lambda_1 & \Delta\lambda_2 & \dots & \Delta\lambda_n \end{bmatrix} \tag{11}$$

Množenje matrik je poenostavljen zapis skalarnega množenja.

$$M = W \cdot x \tag{12}$$

V matriki *M* se tako nahajajo skalarni produkti vsakega od vektorjev podatkov z utežmi vsakega od nevronov.

Matrika *M* je velikosti $m \times n$ (*število nevronov* $\times$ *število vektorjev*); *i*-ti stolpec v matriki predstavlja vrednosti, ki so jih nevroni vrnili za *i*-ti vektor.

Nato za vsak stolpec pogledamo, katera od vrednosti je največja ter na njeno mesto zapišemo enico, na vsa druga polja v stolpcu pa ničle.

Indeks vrstice, v kateri se nahaja maksimalna vrednost *i*-tega stolpca, je hkrati indeks nevrona, kateremu pripada *i*-ti vektor.

Edina predpostavka, ki jo zahteva postopek tekmovalnega učenja, je število nevronov v sloju. Dosti klasičnih postopkov grozdenja smo zavrnili kot neuporabne za naš problem, ker so zahtevali, da vnaprej poznamo število skupin. Pri tekmovalnem sloju pa se kljub izboru števila nevronov zgodi, da nekateri nevroni ostanejo prazni, to pomeni, da nikoli niso zmagali. V enem od poskusov se je zgodilo, da so samo trije nevroni pobrali vse vektorje, drugih devet pa je ostalo praznih. Zato sklepamo, da ima tekmovalni sloj nevronov »lastno voljo« in je rezultat objektiven kljub temu, da smo izbrali število nevronov v sloju.

3.4.2 Kohonenov učilni algoritem

V sloju so razen števila nevronov vsi drugi podatki neznani. Ne poznamo še vrednosti uteži nevronov, prav tako ne vrednosti pragov. Te vrednosti se določijo med učenjem. Določiti moramo samo še število epoh – korakov učenja. Določitev števila je ponavadi stvar poskušanja. Če je korakov premalo, se uteži ne bodo dokončno prilagodile, če pa jih je preveč, tratimo čas, saj se uteži ne

spreminjajo več. Glede na to, kakšno je število epoh, tolikokrat bodo vsi podatki prešli mrežo in za vsak prehod se bodo uteži posodobile.

Na začetku se nevronom pripišejo uteži, ki so naključne, vendar so izbrane iz obsega vhodnih podatkov. Lahko bi začeli tudi z vsemi utežmi, enakimi nič, vendar bi to tehnično nekoliko zapletlo algoritem.

Postopek učenja je povzet po *Matlabovem* priročniku (Demuth et al., 2007). Najprej pošljemo skozi mrežo prvi vektor. Vsi nevroni izračunajo skalarne produkte in vrnejo vrednosti, samo eden od njih (tisti z največjim produktom) pa zmaga. Zmagovalčeve uteži so bile očitno najbolj podobne prvemu vhodnemu vektorju. Sedaj *popravimo* njegove uteži tako, da bodo še bolj podobne vektorju, s pomočjo katerega so zmagali.

Pomembno je, da popravimo uteži *samo* zmagovalcu

$$w_i = w_{i-1} + \alpha_i \cdot (x_i - w_{i-1}) \tag{13}$$

kjer *w* pomeni vektor uteži, ***x*** vhodni vektor, *i* je indeks koraka, *a* pa učilni parameter. Če bi starim utežem prišteli razliko med vhodnim vektorjem in njimi samimi, bi bile nove uteži kar enake vhodnemu vektorju; to je dobro v prvem koraku postopka, kasneje pa ne več. Zato je uveden učilni parameter *a*, ki se med postopkom manjša. V prvem koraku je načeloma lahko enak ena, ponavadi pa je malo manjši. S funkcijo učilnega parametra je določeno, kako njegova vrednost pada z naraščanjem števila korakov. Najbolj splošna oblika je

$$\alpha(i) = \frac{n}{i} \tag{14}$$

kjer je *i* število koraka, *n* pa začetna vrednost.

Obstaja veliko možnosti za določitev te funkcije, vendar pri tekmovalnem učenju ta problem ni pomemben.

Skozi mrežo pošljemo drugi vektor. Če bo podoben prvemu, se bo spet oglasil isti nevron in spet se mu bodo posodobile uteži, a tokrat nekoliko manj. Če pa se bo oglasil drugi vektor, se bodo posodobile uteži njemu.

3.4.3 Pragovni učilni algoritem

V praksi se rado zgodi, da bo vedno zmagovalo samo nekaj nevronov. Situacija je precej odvisna od začetnih naključnih vrednosti uteži, kar pa ni dobro. Nekateri nevroni tako že na začetku dobijo uteži, ki so zelo različne od kateregakoli podatka. Te nevrone imenujemo tudi *mrtvi nevroni* in od njih nikoli nimamo nobene koristi. Želimo jih stimulirati tako, da bodo tudi oni enkrat zmagali in tako ustanovili svojo, novo skupino.

To lahko dosežemo s pragovnim učilnim pravilom. Kohonenov in pragovni učni algoritem sta združljiva. S prvim se popravljajo uteži, z drugim pa pragi nevrona. Tako ne ovirata ali izključujeta drug drugega.

Za vsakega od nevronov beležimo, kolikokrat je zmagal. Nevronom, ki zmagujejo, redko povečujemo prag. Prag je dodatni vhod v nevron, ki se prišteje vrednosti skalarnega produkta in tako poveča možnosti za zmago. Ko bo nevron začel zmagovati in se bo število njegovih zmag povečalo, se bo hkrati začel zmanjševati tudi prag. Tako se »pravičnost« v mreži še dodatno avtomatsko regulira. (Demuth et al., 2009)

Pragovni učilni algoritem ima teoretično dva pozitivna vpliva na mrežo, vendar nismo prepričani, da sta oba pozitivna tudi za naš problem. Zaradi povečevanja uteži bo namreč nevron nekoč zmagal in vektor, s katerim bo zmagal, ga bo »potegnil« proti skupini vektorjev. Tako bo lahko »zavladal« tej skupini in znebili se bomo mrtvega nevrona. Poleg tega pragovni algoritem spodbuja vektorje k temu, da vsak pobere približno enak delež vektorjev. Pri tej točki pa se pojavi problem. Naš cilj je namreč objektivno oblikovati skupine vektorjev, ne pa ustvarjati skupine s podobnim številom članov.

4 IZVEDBA GROZDENJA Z UPORABO UMETNE NEVRONSKE MREŽE

4.1 Postopek dela

4.1.1 Sklopi podatkov

Postopek grozdenja s pomočjo tekmovalnega sloja nevronske mreže sem preizkusil v štirih različnih nalogah.

- 1. naloga
  Najprej sem si izmislil niz desetih vektorjev, pri katerih je na prvi pogled očitno, v kakšne skupine se morajo povezati.

- 2. naloga
  V naslednji nalogi sem v skupine razvrščal večje število vektorjev, najprej petdeset, nato pa še sto. Območje sem obakrat razdelil na štiri dele. Podatke sem namenoma formiral tako, da so si na vsakem od delov območja vektorji med seboj podobni. Tako vnaprej vem, kakšni bi morali biti rezultati.

- 3. naloga
  Tretja naloga je praktično zelo podobna drugi. Spet si zamislim »podobmočja«, na katerih ležijo posamezne skupine, in izdelam takšne vektorje, ki so si znotraj skupine podobni. Razlika je le v tem, da so območja tokrat bolj kompleksna (nepravilnih oblik) in da so si vektorji na splošno bolj podobni med seboj (težje razločevanje različnih).

- 4. naloga
  Zadnja naloga je praktičen preizkus metode. Uporabim 804 vektorjev hitrosti geodinamičnih premikov iz območja Kalifornije ter poskušam dobiti optimalni rezultat.

Namen naloge je raziskati oziroma preizkusiti, kako nam nevronska mreža lahko pomaga pri razvrščanju v skupine. Gre samo za postopek obdelave opazovanj premikov. Podatke o premikih sem prevzel iz drugih virov ali pa sem si jih izdelal sam.

4.1.2 Programska oprema

Delo sem izvajal v programu *Matlab 7.1*, rezultate pa sem izrisoval v programu *AutoCAD 2005*.

Matlab ima vgrajeno orodjarno *Neural network toolbox*, v kateri so funkcije in orodja, s katerimi lahko simuliramo delovanje nevronske mreže. Princip delovanja tekmovalnega učenja, ki sem ga opisal, je skladen s postopki, ki jih uporablja *Matlab;* sam sem opisoval postopek s skalarnimi produkti, *Matlab* pa računa negativne dolžine.

Pri postopku z negativnimi dolžinami normiranje podatkov ni potrebno, zato je zanimivo videti, kako se rezultata nenormiranih in normiranih podatkov razlikujeta.

4.1.3 Izvedba

V *Matlabu* sta kreiranje in uporaba nevronske mreže zelo enostavna. Negativna plat te enostavnosti pa je, da nimaš popolnega vpogleda v postopek, ki se odvija.

Najprej z ukazom *newc* ustvarimo nov tekmovalni sloj. Vhodna podatka sta število nevronov in meje območja, to pomeni razpon posameznih koordinat vhodnih vektorjev.

```
net = newc(minmax(P),15);
```

Lahko si ogledamo začetne vrednosti uteži in pragov ter vse lastnosti ustvarjene mreže. Začetne vrednosti uteži so srednje vrednosti razpona posameznih koordinat podatkov.

Sledi določitev števila korakov učenja. *Matlab* žal nima funkcije, ki bi na primer izrisala povprečno spreminjanje uteži med potekom učenja. Z njeno pomočjo bi lahko videli, pri koliko epohah se uteži ustalijo in se ne spreminjajo več. Poskusil sem s primerjavo uteži, ki jih dobim s 100, 200, 300 epohami …

Tabela 1: Spremembe uteži nevronov med učenjem

Epoh	w1	w2	w3	w4
0	504.9650	495.3450	2.3075	6.1500
100	98.4221	931.4006	0.8184	5.9837
200	93.3251	936.8565	0.7677	6.0011
300	93.3096	936.7906	0.7670	6.0012
400	93.2422	936.7932	0.7674	6.0011
500	93.3048	936.9029	0.7673	6.0011
1000	93.2593	936.8505	0.7671	6.0012

Vidimo, da se vrednosti uteži ustalijo že pri dvesto epohah. Ugotavljam, da uteži ne konvergirajo v pravem pomenu besede, ampak da nihajo.
Za nadaljnje delo sem izbral vrednost 500 epoh.

```
net.trainParam.epochs = 500
```

Ukaz za začetek učenja mreže je

```
net = train(net,P)
```

Vhodna podatka sta ime kreirane mreže ter vektor vhodnih podatkov, rezultat pa je naučena mreža, ki se pravzaprav prepiše čez staro.

Ko je mreža izučena, je treba podatke samo še enkrat spustiti skoznjo ter beležiti, kateri nevron se odzove na katerega od vhodnih vektorjev. To storimo s funkcijo *sim*.

```
a = sim(net,P);
ac = vec2ind(a);
```

Rezultat funkcije *sim* je seznam polj, v katerih se je pojavila vrednost ena, zato ga s funkcijo *vec2ind* iz vektorja pretvorim v indekse. Dobim vektor z *n* elementi, katerih vrednosti so od 1 do *m*. *n* je število vhodnih vektorjev, *m* pa število nevronov. Vsakemu od vhodnih vektorjev se pripiše indeks, v katero skupino (kateremu nevronu) pripada.

Rezultat v obliki indeksov praktično ni preveč uporaben, zato sem hotel vektorje izrisati tako, da se vidi, kakšna sta njihov položaj in oblika ter da so vektorji iste skupine obarvani z isto barvo. Nalogo sem rešil s kreiranjem »script« datotek (*.scr). Za vsak grozd ustvarim svojo datoteko in jo nato uvozim v *AutoCAD* kot nov sloj. Vsak sloj mora imeti svojo barvo, barve pa se morajo med seboj čimbolj razlikovati. Pri desetih slojih to še ni bil problem, pri dvajsetih pa je bil že kar velik.

V prilogi 1 so vsi trije primeri vhodnih podatkov, v prilogi 2 primer izpisa lastnosti mreže, uteži in pragov pred učenjem in po njem, v prilogi 3 pa bo prikazan primer vektorja rezultatov ter algoritem za zapis »script« datotek. Rezultate klasifikacij v obliki risbe bo predstavljen v jedru naloge.

4.1.4 Možnosti preizkušanja mreže

Moje raziskovalno delo je preizkusiti zmožnosti nevronske mreže za grozdenje. Najprej sem se odločil le za to, kakšno mrežo bom uporabil in kako bom predstavil rezultate. Za samo preizkušanje mreže pa mi bodo ostale še naslednje možnosti:

- določanje števila epoh učenja,
- določanje števila nevronov v mreži,
- iskanje oblike podatkov, pri kateri bo mreža najlažje našla pravi rezultat.

Najpomembnejša od teh se zdi tretja točka, zato ji namenímo še nekaj vrstic.

Podatki so podani v obliki štirih komponent. Za vsak vektor poznamo dve koordinati položaja točke, na katero se nanaša, in hitrost te točke v dveh smereh. To je edina informacija o predmetih, ki jih bomo razvrščali v skupine. Postopek grozdenja lahko izvajamo na podlagi različnih kriterijev, na primer: lahko bi se odločili in razvrščal predmete samo glede na koordinato y. Tako bi se skupine formirale izključno na podlagi podatka o geografski širini točk. Med seboj bi se povezale točke s podobno geografsko širino.

V resnici bomo grozdenje izvedli tako, da se bodo upoštevale podobnosti vseh štirih komponent. Tako se bodo v iste skupine združile točke, ki imajo hkrati podobne položaje, smeri in velikosti vektorjev.

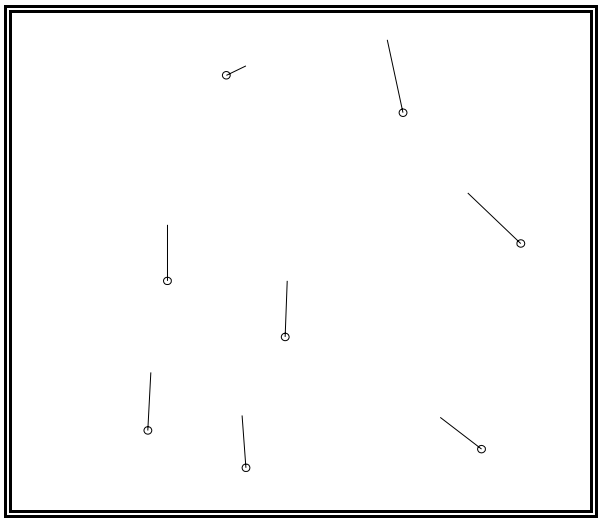
Načeloma so vrednosti komponent, ki predstavljajo položaj, veliko večje od komponent, ki predstavljajo hitrost premika; zato bo imel morda položaj premoč nad obliko vektorjev in se bodo v isto skupino povezale točke, ki ležijo na nekem območju, čeprav se premikajo v povsem različnih smereh.

Vse našteto je razlog, da sem med delom začel ustvarjati različne oblike podatkov, ki jih imenujem »*forme*«. Gre za to, da vektorje, ki jih razvrščam v skupine, zapišem na različne načine in zato mreža vrne različne rezultate, na primer: poleg osnovnih štirih komponent vsakemu vektorju dodam še njegovo smer in velikost. Dejansko podatkom ne dodajam nobene informacije, le isto stvar prikažem na drugačen način in opazujem, kako se na to odziva nevronska mreža.

Isto nalogo (niz vektorjev) lahko grozdim osemkrat, le da vsakič spremenim formo podatkov. Vsakega od teh poskusov poimenujem s »forma *n*«. Včasih v kateri od form spremenim tudi število nevronov, nikoli pa ne spremenim števila epoh. Pri vsaki od form je načeloma navedeno, kako so oblikovani vhodni podatki, prikazan je rezultat grozdenja in dodan kratek komentar.

4.2 Naloga 1 - grozdenje desetih vektorjev

Na sliki 4.1 so prikazani vhodni podatki

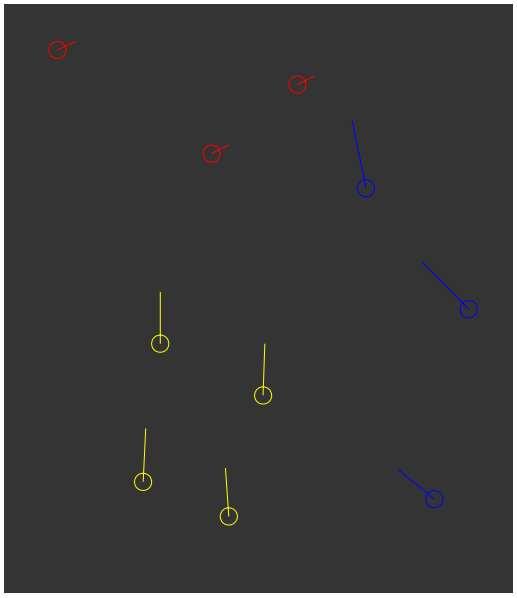


Slika 4.1: Podatki prve naloge

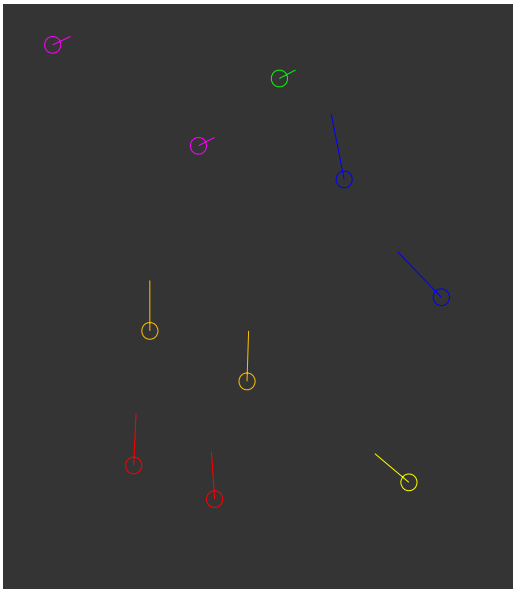
Tabela 2: Podatki prve naloge

X	Y	dX	dY
10	5	0.15	3.1
11	13	0.0	3.0
15	3	-0.2	2.8
17	10	0.1	3.0
27	4	-2.1	1.7
29	15	-2.7	2.7
23	22	-0.8	3.9
14	24	1.0	0.5
19	28	1.0	0.5
5	30	1.1	0.5

Klasifikacijo izvedem s tremi in šestimi nevroni v mreži.



Slika 4.2: Rezultat grozdenja s tremi nevroni



Slika 4.3: Rezultat grozdenja s šestimi nevroni

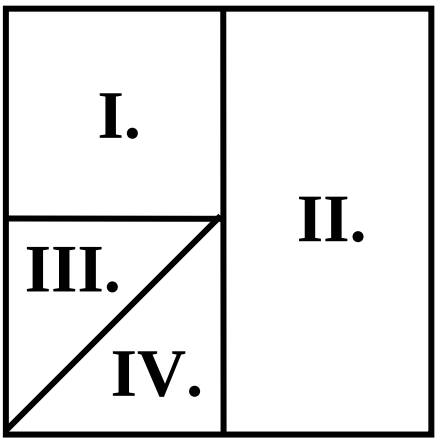
S tremi nevroni dobim želeni rezultat. Zaradi majhne gostote točk oziroma relativno velike razdalje dobim pri večjem številu nevronov preveč skupin. V primeru desetih nevronov bi vsak vektor ustvaril svojo skupino, v kateri bi bil edini član.

Tako lahko zaključim, da je pri tako majhnem vzorcu nujno, da poznamo pravilno število skupin, še lažje pa je »na pamet« ugotoviti skupine.

4.3 Naloga 2 – kvadrat - grozdenje petdesetih in stotih vektorjev

4.3.1 Opis naloge

V tej nalogi bom poskusil iz dveh nizov podatkov prepoznati oblike skupin podatkov. Prvi niz bo sestavljalo petdeset vektorjev, drugega pa sto. Vektorji v obeh nizih bodo razpostavljeni po kvadratnem območju velikosti 100 × 100. Kvadrat bom razdelil na štiri dele kot kaže slika.



Slika 4.4: Razdelitev območja podatkov

Vsak od delov bo predstavljal eno skupino oziroma grozd. Podatke bom izdelal sam in sicer tako, da si bodo vektorji, ležeči v istem delu območja, med seboj podobni.

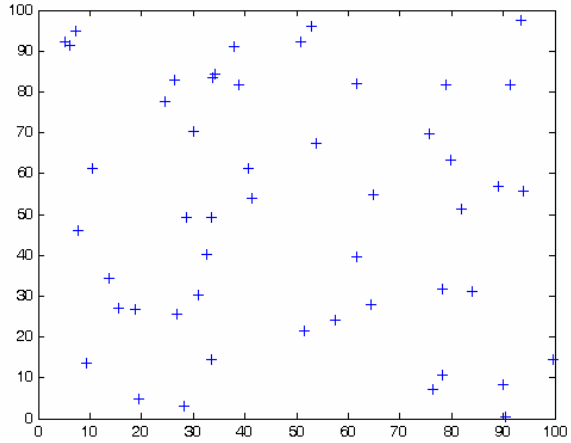
4.3.2 Izdelava podatkov

Podatke za ta poskus sem izdelal tako, da sem začel s položaji vektorjev. V *Matlabu* z ukazom

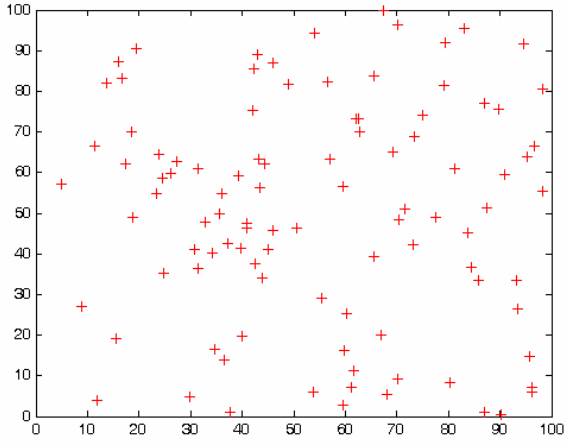
```
P=rand(2,100).*1000;
plot(P(1,:),P(2,:),'+r');
```

kreiram matriko v velikosti 2 × 50, v kateri so naključne vrednosti od nič do ena, in jo pomnožim s sto, tako da se območje točk razširi na območje od nič do tisoč v

obeh smereh. Popolnoma enako storim za vzorec stotih točk. Dobim naslednja primera, prikazana na slikah:



Slika 4.5: Položaji točk (50)

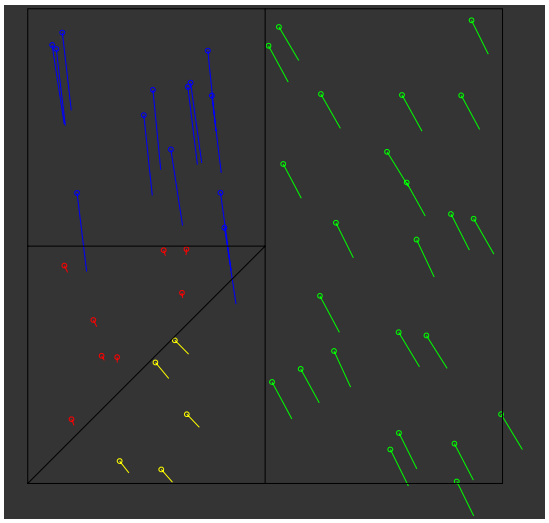


Slika 4.6: Položaji točk (100)

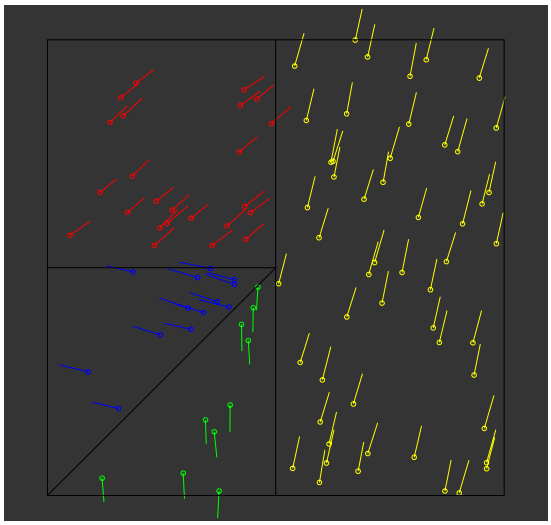
Koordinate točk sem uvozil v program *MS Excel*, kjer sem vsaki od točk pripisal še dva podatka, komponenti vektorja hitrosti. Vsem točkam na desni strani območja (področje II, kjer je koordinata x > 50) sem pripisal podobne vektorje, prav tako pa tudi vsem vektorjem s področij I, III in IV. Vsako področje tako vsebuje med seboj podobne vektorje, vendar drugačne od tistih iz drugih področij.

Za vsako od skupin sem določil globalni vektor. Vsaka skupina je dobila dve naključni vrednosti z intervala nič do ena, ki sem ju pomnožil z neko celo vrednostjo. Tako sem za vsako skupino dobil par komponent »glavnega« vektorja. Ker pa ne bi rad, da imajo vse točke iz istega področja povsem enake vektorje, sem te, »glavne« vrednosti, pokvaril. Vsaki točki sem pripisal vrednost glavnega vektorja, pokvarjeno za naključno vrednost od –0,5 do 0,5.

Nazadnje sem dobil naslednjo sliko; vektorji posameznih skupin so obarvani z različnimi barvami. Na levi sliki je situacija petdesetih vektorjev, na desni pa stotih.



Slika 4.7: Podatki, druga naloga (50)



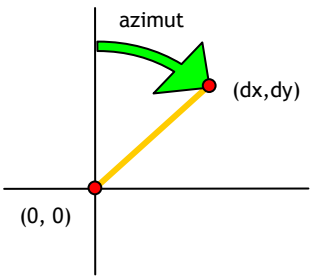
Slika 4.8: Podatki, druga naloga (100)

4.3.3 Možnosti oblik podatkov - forme

Iste podatke lahko v nevronsko mrežo pošljemo v različnih oblikah. Dejstvo je, da pri različnih oblikah dobimo različne rezultate. Pri spreminjanju oblik podatkom ne dodamo nobene informacije. Preden pa začnemo ustvarjati forme podatkov, spoznajmo metode, s katerimi sem preoblikoval osnovne podatke.

Azimut - smer vektorja

Azimut je kot med smerjo osi Y in smerjo vektorja. Če vektor kaže navzgor, je enak 0° oziroma 360°.



Azimut predstavlja smer vektorja. Velja, da imajo vektorji podobnih azimutov podobne smeri, razen v primeru azimutov, ki so blizu nič; na primer smer 359° je

zelo podobna smeri 0°, vendar pa sta številki zelo različni, zato ju algoritem ne šteje za podobni lastnosti.

Dolžina vektorja

Dolžina vektorja (dX, dY) praktično pomeni hitrost premikanja zemeljskega površja. Definirana je kot

$$d = \sqrt{dX^2 + dY^2} \tag{15}$$

Podatka o smeri in dolžini vektorja sta linearno odvisna od podatkov dX in dY, zato mreži ne prinašata nobenih novih informacij; zanimivo pa ju je dodati v niz podatkov, ker s tem poudarimo obliko vektorjev pred položajem.

Normiranje

Normiranje je postopek, s katerim dosežemo, da je posamezna koordinata vektorja standardno normalno porazdeljena. Praktično to pomeni, da je težišče vseh vektorjev v koordinatnem izhodišču ter da je standardni odklon vsake od komponent enak ena. (Turk, 2005)

Definirajmo srednjo vrednost vsem štirim komponentam:

$$m_x = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad m_y = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad m_{dx} = \frac{1}{n}\sum_{i=1}^{n} dx_i \qquad m_{dy} = \frac{1}{n}\sum_{i=1}^{n} dy_i \tag{16}$$

kjer je n število vektorjev. Nato definiramo še standardni odklon teh komponent:

$$s_x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - m_x)^2} \qquad s_y = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - m_y)^2}$$

$$s_{dx} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(dx_i - m_{dx})^2} \qquad s_{dy} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(dy_i - m_{dy})^2} \tag{17}$$

Za primer stotih točk imajo »surovi« podatki takšne srednje vrednosti in standardne odklone, kot je razvidno iz tabele 3.

Tabela 3: Srednje vrednosti in standardni odkloni podatkov

	x	y	dx	dy
Srednja vrednost	508.4893	561.5435	2.2780	6.1081
Standardni odklon	275.4152	268.8436	1.8675	1.6362

Želimo doseči, da so srednje vrednosti vsake od komponent vhodnega vektorja enake nič in njihovi standardni odkloni enaki ena.

Novo vrednost vsakega podatka izračunamo tako, da mu odštejemo srednjo vrednost ter razliko delimo s standardnim odklonom komponente, ki mu pripada.

$$x_{nov} = \frac{x_{star} - m_x}{s_x} \qquad y_{nov} = \frac{y_{star} - m_y}{s_y} \qquad dx_{nov} = \frac{dx_{star} - m_{dx}}{s_{dx}} \qquad dy_{nov} = \frac{dy_{star} - m_{dy}}{s_{dy}} \quad (18)$$

Sedaj velja, da je srednja vrednost vsake od komponent vektorjev enaka nič in standardni odklon vsake enak ena.

Vsak vektor je v osnovi dan s štirimi podatki: (x, y, dx, dy). Tem podatkom dodajamo še podatka o dolžini in smeri ali pa jih nadomestimo z normiranimi. Število nevronov je bilo v vseh varianatah 15.

V tabeli so predstavljene vse uporabljene oblike podatkov, ki sem jih uporabil za razvrščanje vektorjev v skupine. Vsako od oblik grozdim posebej in mreža vselej vrne drugačen rezultat.

Tabela 4: Komponente, ki jih vsebuje vektor v določeni formi (obliki)

Forma						
1	x	y	dx	dy		
2	x	y	dx	dy	dolžina	azimut
3			dx	dy		
4			dx	dy	dolžina	azimut
5	x	y	3x dx	3x dy		
6	x_{norm}	y_{norm}	dx_{norm}	dy_{norm}		
7	x_{norm}	y_{norm}	dx_{norm}	dy_{norm}	$dolž_{norm}$	azi_{norm}

Primera: v formi 3 je vektor podan samo s komponentama dx in dy, v formi 6 pa je vsak vektor podan s štirimi normiranimi vrednostmi svojih starih komponent.

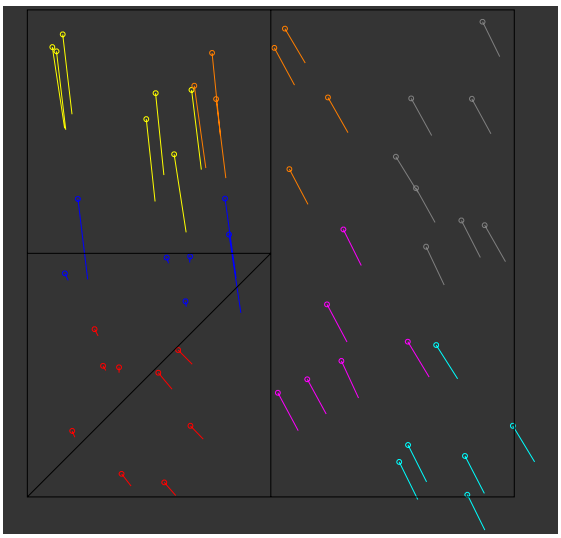
V nadaljevanju bom oba niza (petdeset in sto) grozdil z vsako od predstavljenih oblik podatkov, form. Pri grozdenju petdesetih vektorjev sem uporabil mrežo s sedmimi nevroni, pri stotih vektorjih pa mrežo z desetimi nevroni. Nazadnje bom poskusil rezultate izboljšati še s spreminjanjem števila nevronov.

V vseh poskusih sem za učenje uporabil število dvesto epoh.

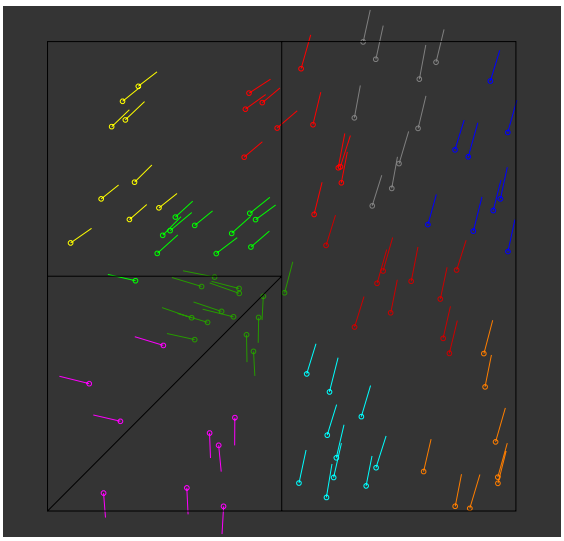
4.3.4 Razvrščanje vektorjev v skupine

Forma 1

Uporabim osnovne štiri podatke vektorjev (x, y, dx, dy) in dobim naslednji rezultat.



Slika 4.9: Kvadrat - forma 1 (50)

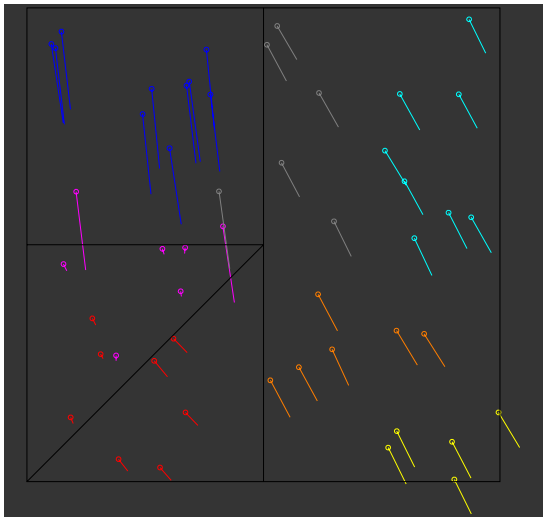


Slika 4.10: Kvadrat - forma 1 (100)

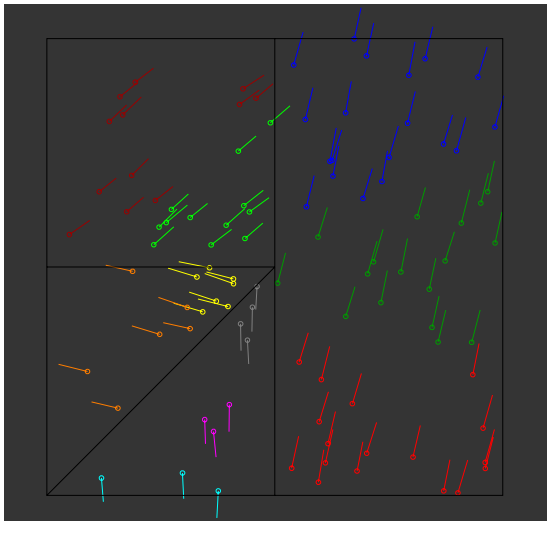
Rezultat se mi v obeh primerih zdi izjemno slab. Vsi izdelani grozdi (skupine pobarvane z isto barvo) padajo prek meja želenih območij. Poleg tega je problem tudi pragovni učilni algoritem, ki povzroča skupine s podobnim številom vektorjev.

Forma 2

Osnovnim podatkom dodam še dolžino in azimut ter dobim (x, y, dx, dy, dolžina, azimut).



Slika 4.11: Kvadrat - forma 2 (100)

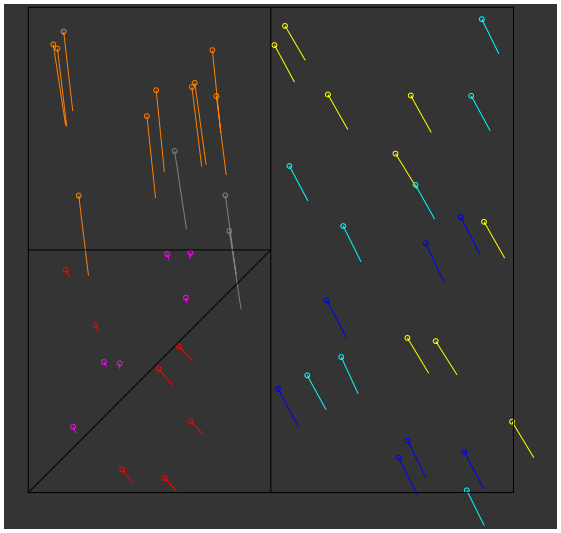


Slika 4.12: Kvadrat - forma 2 (100)

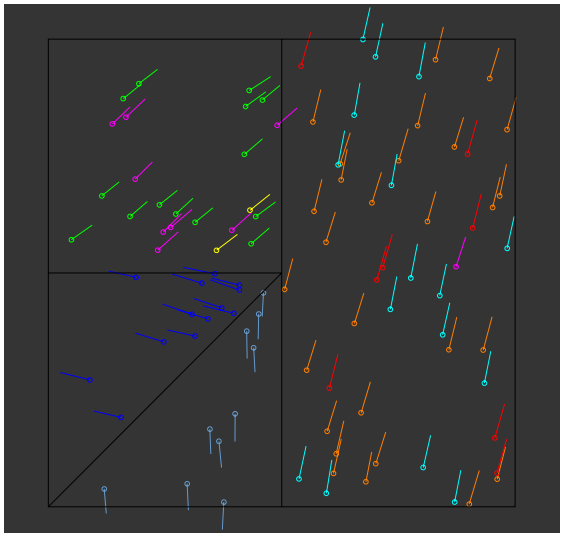
Rezultat na levi (50 vektorjev) je še vedno zelo slab, na desni pa se je izboljšal. S povezovanjem določenih grozdov bi že lahko dobili pravilen rezultat. Meje med formiranimi skupinami v nekaterih primerih sovpadajo z želenimi mejami; vprašanje pa je, kako ugotoviti, katere so prave meje z realnimi podatki, pri katerih ne poznamo pravilnega rezultata.

Forma 3

Uporabim samo komponenti vektorja (dx, dy), brez položajev. Povezali naj bi se vektorji, ki so si podobni po obliki ne glede na medsebojno lego.



Slika 4.13: Kvadrat - forma 3 (50)



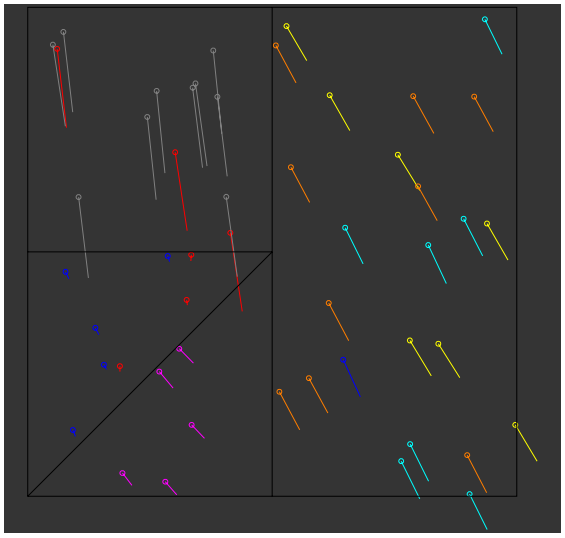
Slika 4.14: Kvadrat - forma 3 (100)

Spet nastane problem s petdesetimi vektorji. Kakorkoli, desno polovico bi lahko prepoznali z združitvijo rumenih, svetlo in temno modrih skupin, zgornjo levo četrtino pa z združitvijo oranžnega in sivega; spodnja leva četrtina ostaja nerešena.

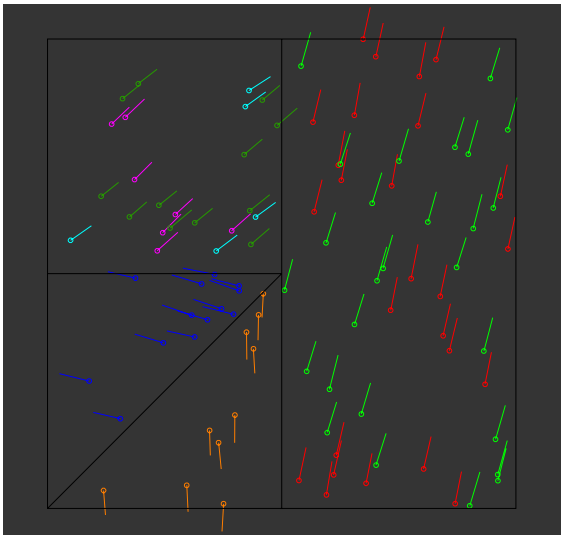
Pri stotih vektorjih se spodnja leva četrtina reši pravilno, ostali del pa bi lahko rešili z združevanjem. Očitno na rezultat vpliva količina vektorjev oziroma podobnost vektorjev iz sosednjih območij.

Forma 4

Z uporabo komponent (dx, dy, dolžina, azimut) še bolj poudarim obliko vektorjev.



Slika 4.15: Kvadrat - forma 4 (50)

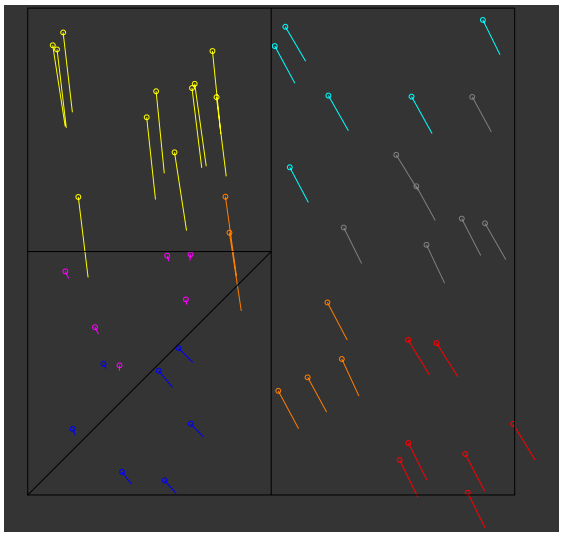


Slika 4.16: Kvadrat - forma 4 (100)

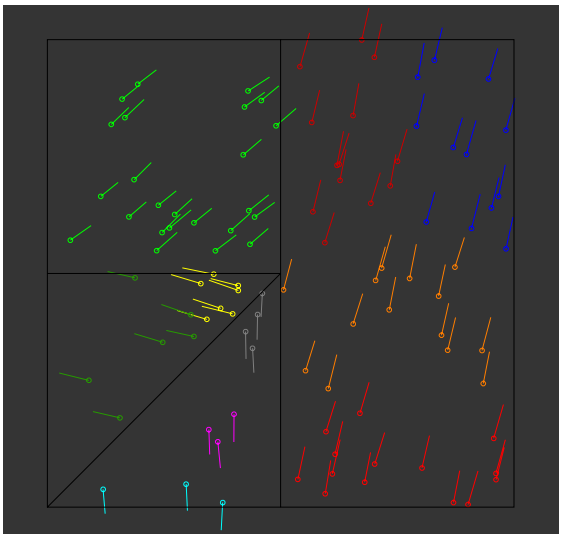
Rezultata sta zelo podobna kot pri formi 3.

Forma 5

Forma 5 je poskus poudariti obliko pred položajem s tem, da komponenti oblike vektorjev (dx, dy) uporabim trikrat, komponenti položaja pa samo enkrat; tako so vhodni vektorji oblike (x, y, dx, dy, dx, dy, dx, dy,).



Slika 4.17: Kvadrat - forma 5 (50)

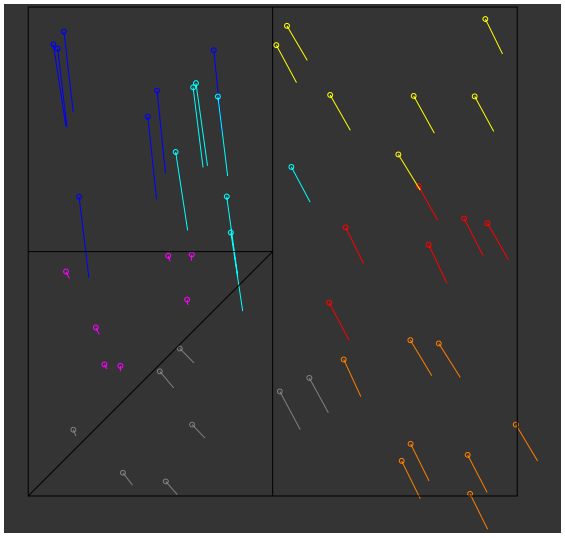


Slika 4.18: Kvadrat - forma 5 (100)

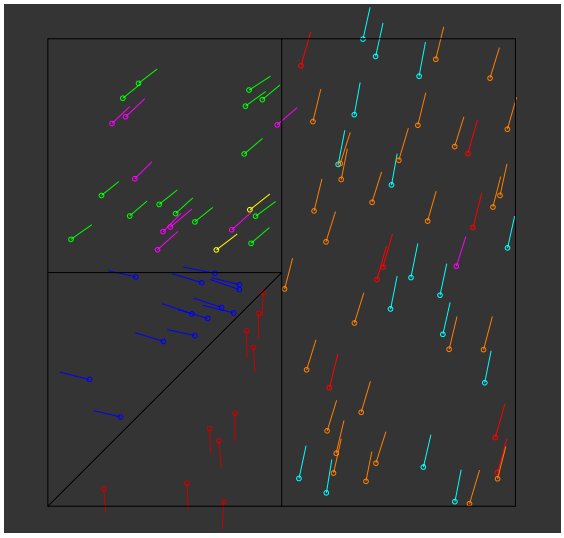
Na levi se je situacija v spodnjem levem delu morda nekoliko izboljšala. Na desni se je prvič pravilno formirala skupina območja levo zgoraj, kar je svojevrsten dosežek.

Forma 6

Uporabim normirane vrednosti osnovnega vektorja $(x, y, dx, dy)_{normirano}$. Od te forme si obetam najboljši rezultat.



Slika 4.19: Kvadrat - forma 6 (50)

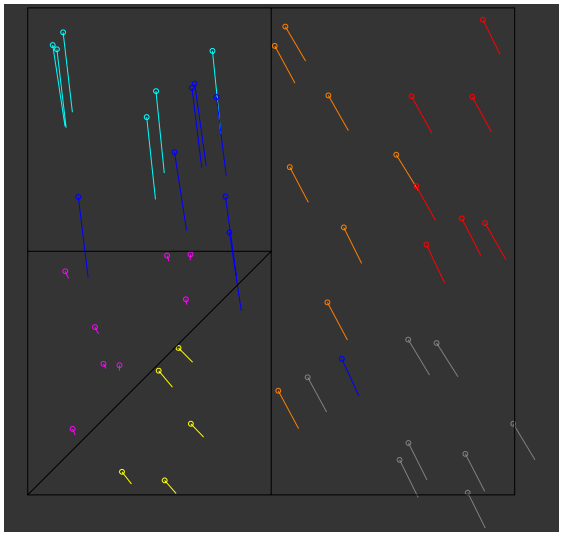


Slika 4.20: Kvadrat - forma 6 (100)

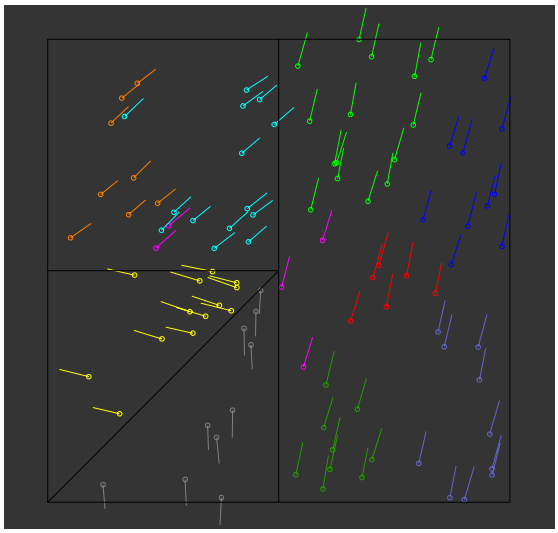
S petdesetimi vektorji dobim že skoraj zadovoljiv rezultat. Menim, da je problem v relativni redkosti točk na območju. Rezultat stotih vektorjev je zelo podoben rezultatu forme 4. Razen osamljenega vijoličnega vektorja na desni bi lahko, s povezovanjem tistih grozdov, ki so pomešani v skupen grozd, dobili pravilen rezultat.

Forma 7

Ta forma je samo normirana oblika forme dve, torej (x, y, dx, dy, dolžina, azimut)$_{normirano}$. Rezultat bo moral biti podoben rezultatu forme 6, s tem da bi bila nekoliko poudarjena oblika pred položajem.



Slika 4.21: Kvadrat - forma 7 (50)



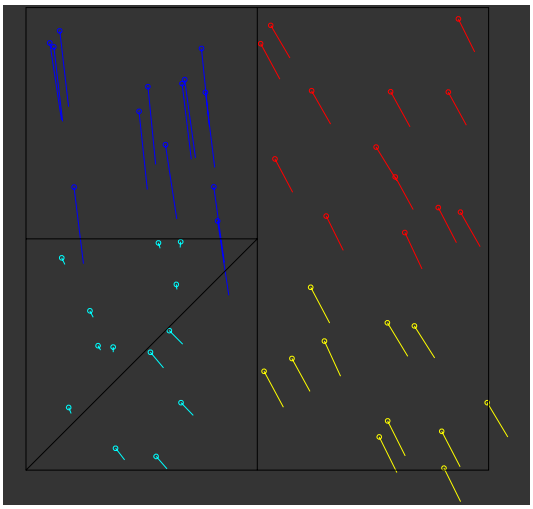
Slika 4.22: Kvadrat - forma 7 (100)

S petdesetimi vektorji mi je končno uspelo razrešiti spodnji levi predel. Pri stotih vektorjih rezultat ne prinaša ničesar, česar ne bi pokazale že prejšnje forme.

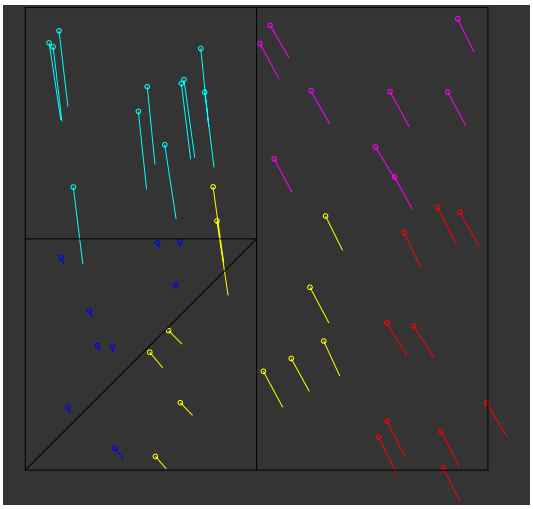
Dodatni poskusi

Kadar je bil rezultat že zelo blizu želenemu, sem ga poskusil izboljšati s tem, da sem mreži določil drugačno število nevronov; to pomeni, da sem ji dal podatek, koliko skupin želim dobiti.

Za primer petdesetih vektorjev sem poskusil izboljšati formo 7. Najprej sem zahteval štiri skupine (določil štiri nevrone). Ker rezultat ni bil pravi, sem poskusil še s petimi.



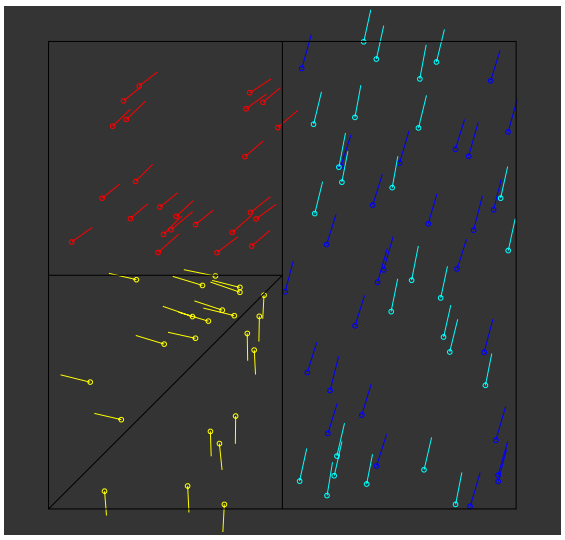
Slika 4.23: Forma 7, 4 nevroni (50)



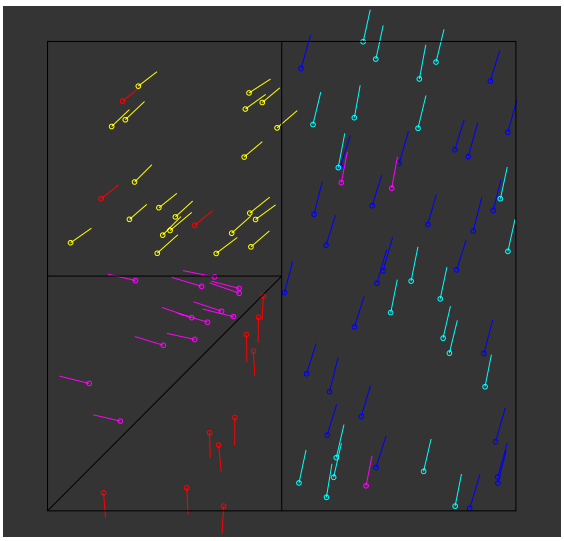
Slika 4.24: Forma 7, 5 nevronov (50)

Menim, da je levi rezultat še vedno boljši, čeprav o spodnjih levih dveh skupinah ne izvem ničesar.

Pri stotih vektorjih pa sem poskušal izboljšati rezultate, dobljene s formami 3, 4 in 5; vsako sem grozdil še s štirimi in petimi nevroni.

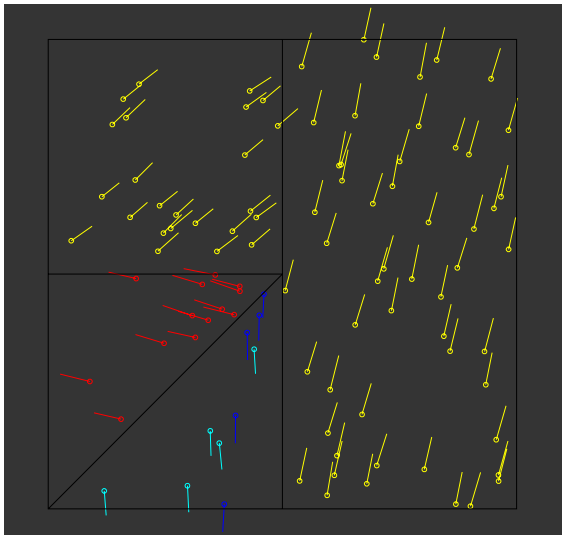


Slika 4.25: Forma 3, 4 nevroni (100)

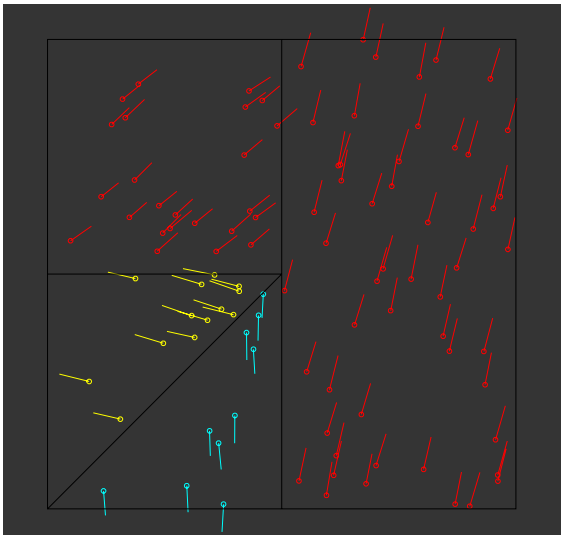


Slika 4.26: Forma 3, 5 nevronov (100)

Pričakoval sem pravilen rezultat pri štirih nevronih, vendar ima prizadevanje utežnega učilnega algoritma po podobno velikih skupinah velik vpliv; vsaka skupina je namreč pobrala približno četrtino vektorjev.



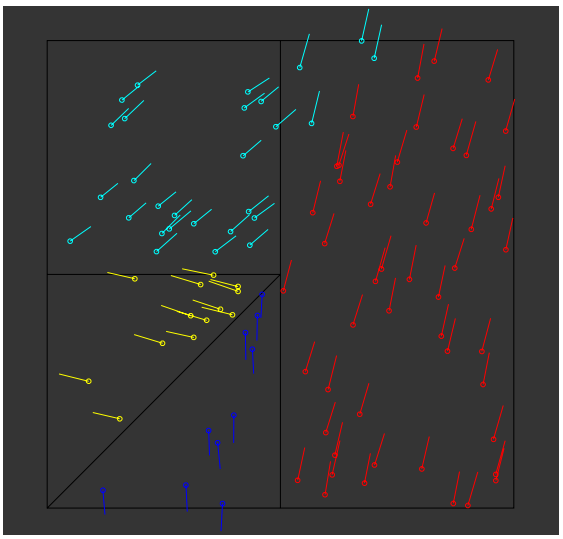
Slika 4.27: Forma 4, 4 nevroni (100)



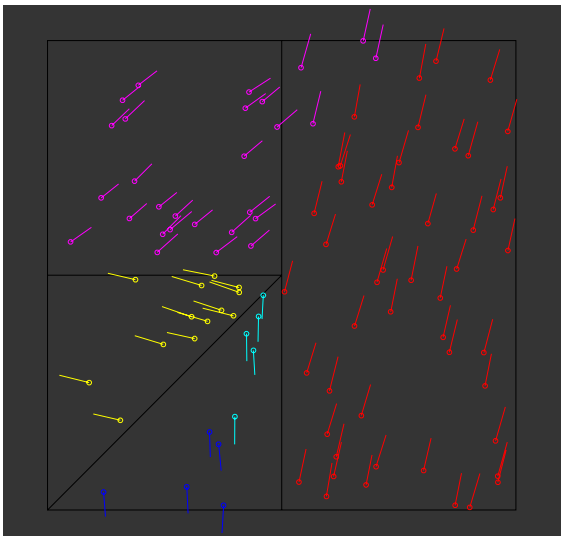
Slika 4.28: Forma 4, 5 nevronov (100)

Rezultata sta izredno zanimiva. V obeh primerih približno tri četrtine vektorjev pade v eno samo skupino, kar je v nasprotju z ugotovitvijo pri prejšnjih dveh poskusih. Navdušen sem nad dejstvom, da sta pri relativno majhnem številu nevronov kar dva ostala prazna (desni primer).

Zadnji poskus je grozdenje forme 5 s štirimi in petimi nevroni.



Slika 4.29: Forma 5, 4 nevroni (100)



Slika 4.30: Forma 5, 5 nevronov (100)

Rezultat se mi zdi odličen, še posebej tisti na levi strani.

4.3.5 Komentar druge naloge

Naloga se mi zdi zanimiva posebej zato, ker je za nevronsko mrežo težka. Iz izkušenj, ki sem jih dobil do sedaj, lahko rečem, da nevronska mreža lažje najde skupine, ki so bolj naravno oblikovane in zaokrožene, tu pa gre za pravokotnik, kvadrat in trikotnik. Druga stvar, ki tudi na neki način ovira mrežo, pa je zelo različna velikost želenih skupin. Območje II je namreč kar štirikrat večje od območij III in IV, tako po površini kot tudi po številu objektov.

Seveda je ob znanih pravilnih rezultatih lahko izvajati poskuse v smeri pridobivanja teh rezultatov iz nevronske mreže. Vprašanje pa je, kako naj to počnemo s pravimi podatki, kjer ni želenih rezultatov.

Mislim, da bi rezultate vseh grozdenj lahko izboljšali z dvema popravkoma. Prvič, ignorirati moramo osamele vektorje. Odlična primera sta rezultata forme 6 s stotimi vektorji ter forme 3 s stotimi vektorji in petimi nevroni. Če posamezen vektor leži globoko v območju druge (zaključene skupine), ga moramo ignorirati.

Drugi popravek pa bi bil združevanje več skupin v eno. To bi storil takrat, kadar je več skupin zelo pomešanih med seboj in so si njihovi vektorji po obliki zelo podobni. Dobra primera sta desni polovici območja pri formah 6 in 3 s stotimi vektorji in desetimi nevroni.

Tako lahko zaključimo, da bi bilo ob uporabi teh dveh popravkov in vseh predstavljenih rezultatov mogoče pridobiti pravilne meje med območji. Najprej moramo upoštevati oba popravka in narisati vse meje med dobljenimi grozdi. Ostaja pa velik problem, kako izmed vseh mej, dobljenih s tem postopkom, izbrati tiste štiri, ki so pravilne.

4.4 Naloga 3 - grozdenje stotih vektorjev

4.4.1 Opis naloge

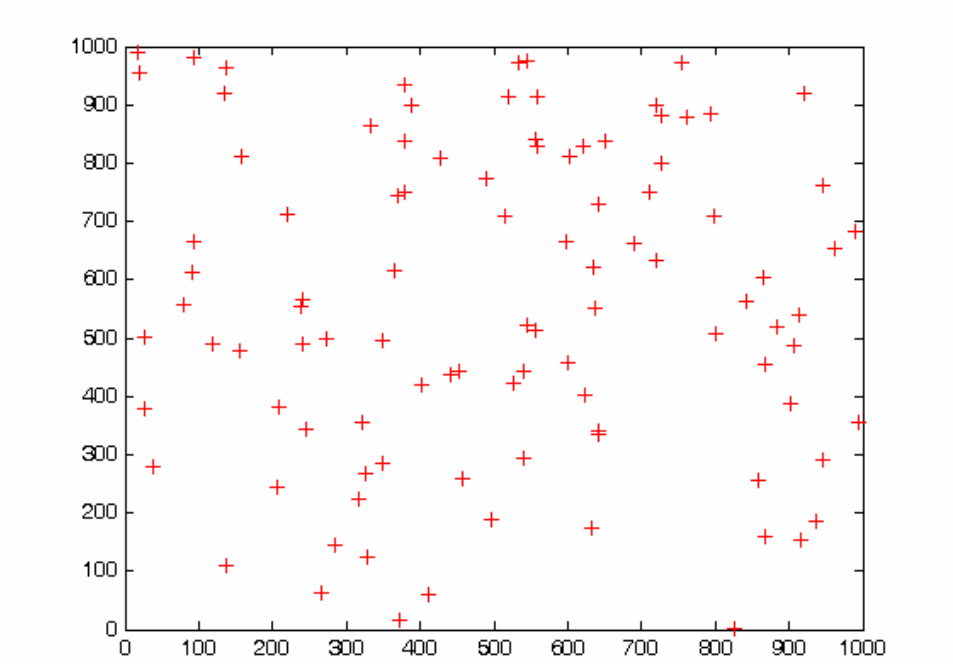
Naloga 3 bo, vsaj kar se tiče postopka, zelo podobna prejšnji nalogi; razlikovala se bo le po podatkih, ki jih bom grozdil. Tokrat bom formiral sto vektorjev ter jih razdelil v dvanajst skupin. Skupine bodo imele bolj »naravne« oblike, poleg tega pa se vektorji posameznih skupin med seboj ne bodo tako očitno razlikovali.

4.4.2 Izdelava podatkov

Tudi za ta poskus sem sam izdelal podatke. Začel sem s položaji vektorjev. V *Matlabu* z ukazom

```
P=rand(2,100).*1000;
plot(P(1,:),P(2,:),'+r');
```

kreiram matriko v velikosti 2 x 100, v kateri so naključne vrednosti od nič do ena; množim jo s tisoč, tako da se območje točk razširi na območje od nič do tisoč v obeh smereh. Rezultat izrišem.



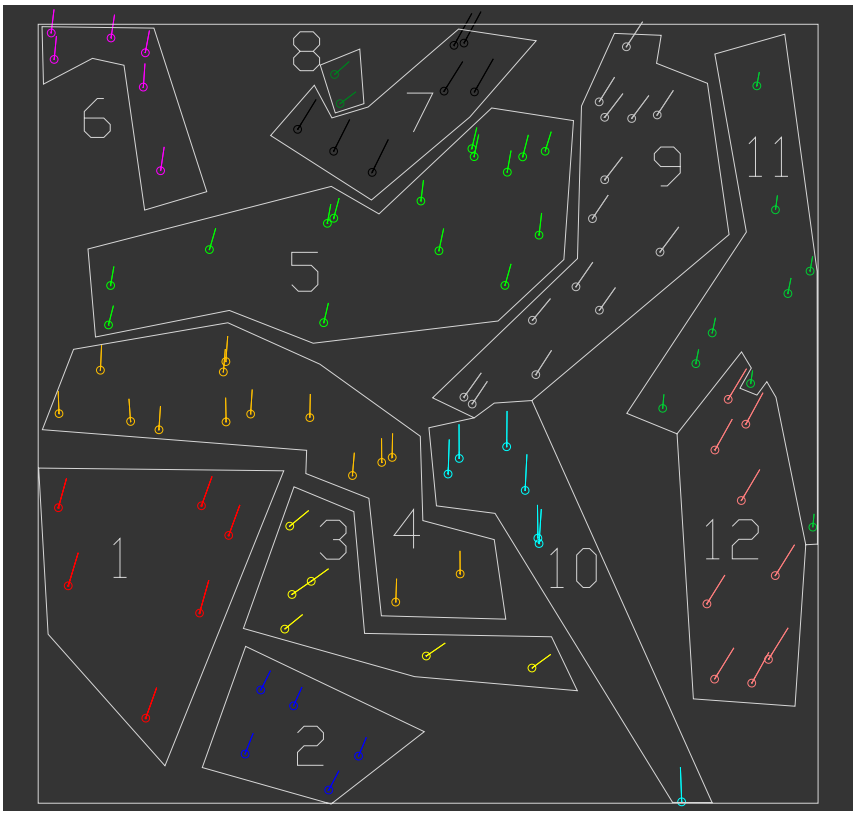
Slika 4.31: Položaji vektorjev pri tretji nalogi

Sliko sem natisnil ter s svinčnikom obkrožil točke, za katere želim, da so v isti skupini. Izdelal sem dvanajst skupin. S pomočjo *MS Excel-a* sem vsakemu paru koordinat pripisal številko skupine, v katero spada.

Za vsako od skupin sem določil globalni vektor. Vsaka skupina je dobila dve naključni vrednosti z intervala nič do ena. Prvo sem množil s 5, drugo pa z 9. Tako so globalni vektorji skupin podobno usmerjeni (azimut je med 0 in 90), njihov razpon pa je nekje med (0,0) in (5,9).

Vsaki točki sem pripisal globalni vektor njene skupine, nato pa sem ga »pokvaril« za naključno vrednost z intervala (-0,5, 0,5), sicer bi bili vsi vektorji skupine povsem enaki.

Izkaže se, da je oblika premikov podobna, kot so situacije v naravi. Vektorji si med seboj niso preveč podobni, kvečjemu premalo. Menim, da so takšni testni podatki odlični.



Slika 4.32: Izdelani podatki za tretjo nalogo

4.4.3 Možnosti oblik podatkov – forme

Uporabljal bom enake forme kot v prejšnji nalogi, le da bom tokrat kot lastnost forme upošteval tudi število nevronov. Tako sem sestavil naslednjo tabelo.

Tabela 5: Tabela form za tretjo nalogo

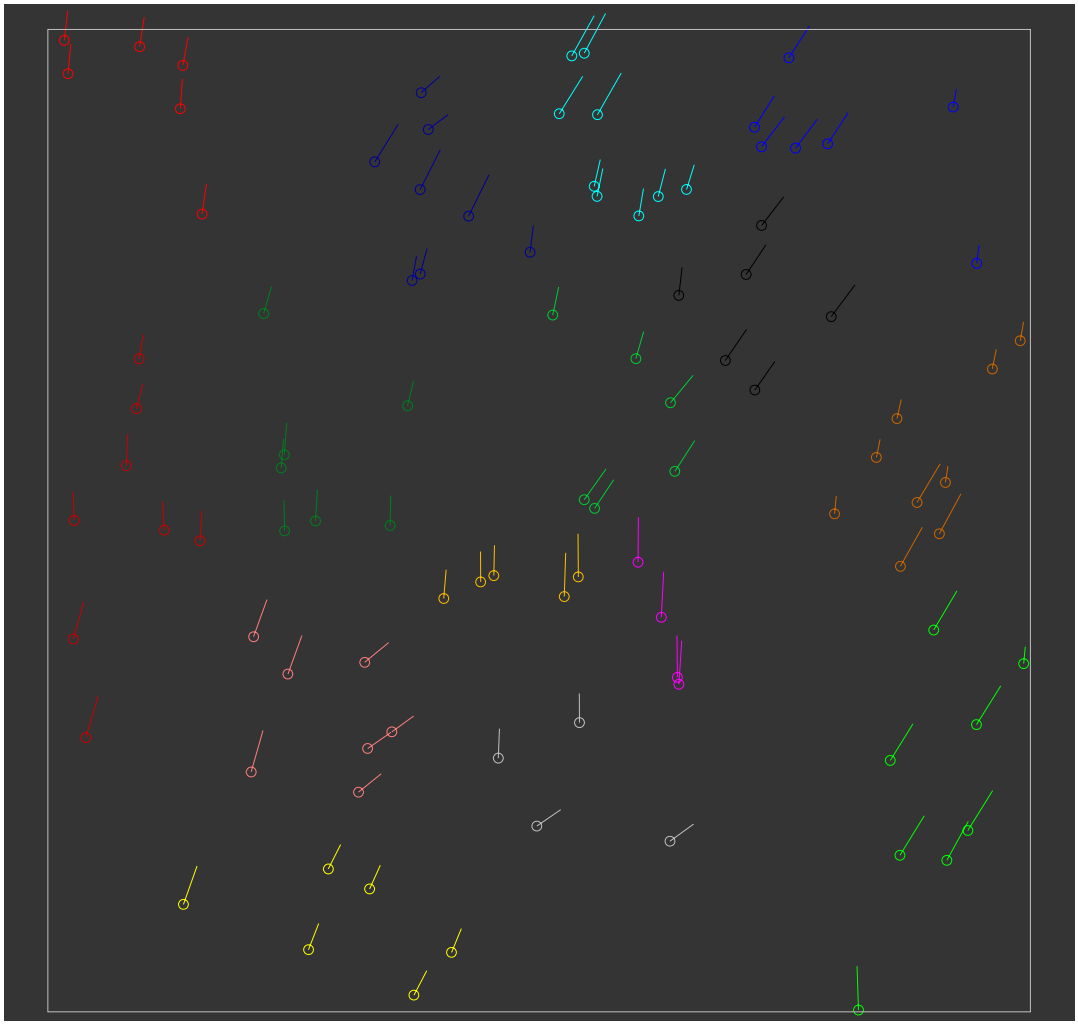
Forma							št. nevronov
1	x	y	dx	dy			15
2	x	y	dx	dy	dolžina	azimut	15
3			dx	dy			15
4			dx	dy	dolžina	azimut	15
5	x	y	3x dx	3x dy			15
6	x_{norm}	y_{norm}	dx_{norm}	dy_{norm}			15
7	x_{norm}	y_{norm}	dx_{norm}	dy_{norm}	$dolž_{norm}$	azi_{norm}	15
8	x_{norm}	y_{norm}	dx_{norm}	dy_{norm}	$dolž_{norm}$	azi_{norm}	12

4.4.4 Grozdenje variant

Za vsako obliko podatkov bom prikazal rezultat postopka z barvami vektorjev. Vektorje, ki so pobarvani z isto barvo, je mreža prepoznala kot pripadnike iste skupine. Za boljšo ponazoritev so prikazani samo rezultati nevronske mreže. Na podlagi slike si ustvarimo podobo skupin, ki jih je ustvarila mreža. Po tem s prosojnico, priloženo v prilogi C, prekrijemo sliko in vidimo, kakšne so želene skupine. To počnem zato, ker se mi zdi slika z vrisanimi želenimi mejami nejasna.

Forma 1

Grozdenje se izvede s pomočjo petnajstih nevronov in s podatki x, y, dx in dy.



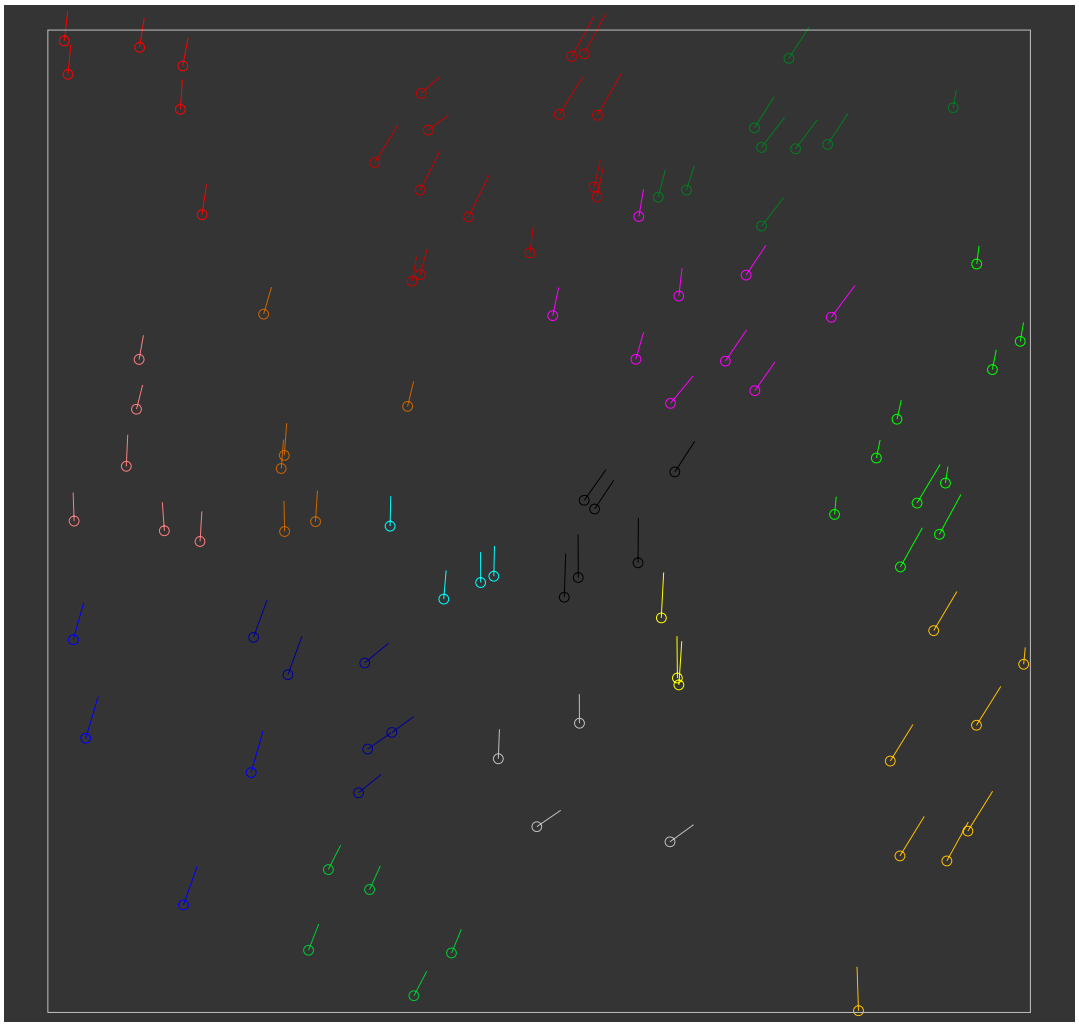
Slika 4.33: Tretja naloga - forma 1

Rezultat je slab. Vidimo, da se niti ena skupina ni oblikovala pravilno. Prevelik poudarek je na združevanju vektorjev s podobnim položajem in premajhen na obliki vektorjev. Pri naslednji formi podatkov bom poudaril obliko s tem, da bom vsakemu vektorju pripisal še dolžino in smer.

Forma 2

Podatkom x, y, dx, dy dodam še dolžino in smer.

Ker se je izkazalo, da so nekateri azimuti manjši od nič, sem vsem prištel vrednost deset. Na njihove medsebojne podobnosti ta popravek ne vpliva, razreši pa problem preskoka med 360° in 0°.

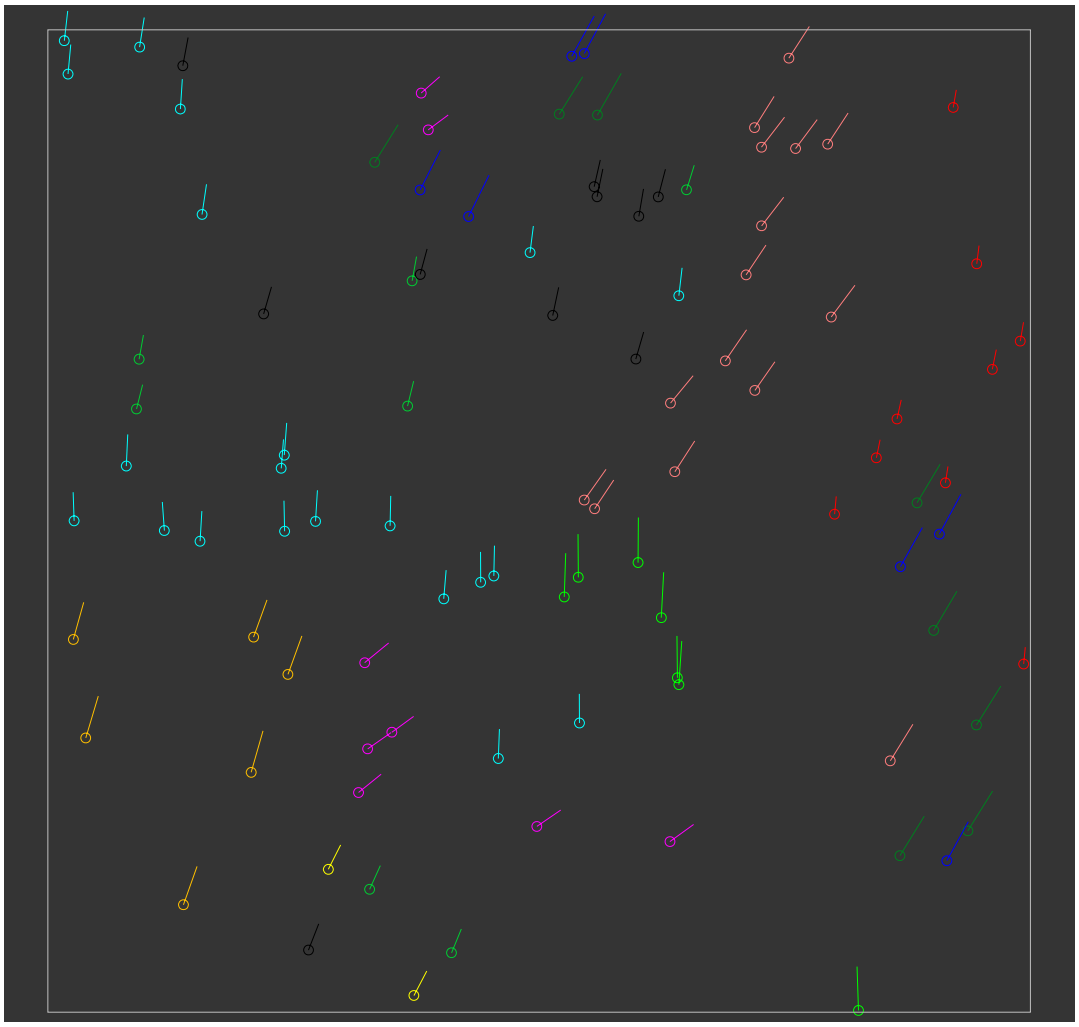


Slika 4.34: Tretja naloga – forma 2

Skupine, ki so majhne in lepo zaključene, se oblikujejo pravilno. Zelo dober primer sta skupini 2 in 6. Problem je še vedno pri skupinah, ki so bolj razpotegnjene (4),(5),(10) oziroma objemajo druga drugo (11, 12).

Forma 3

Sedaj poskusim izključiti položaje vektorjev in klasificirati samo glede na obliko dx in dy.



Slika 4.35: Tretja naloga - forma 3

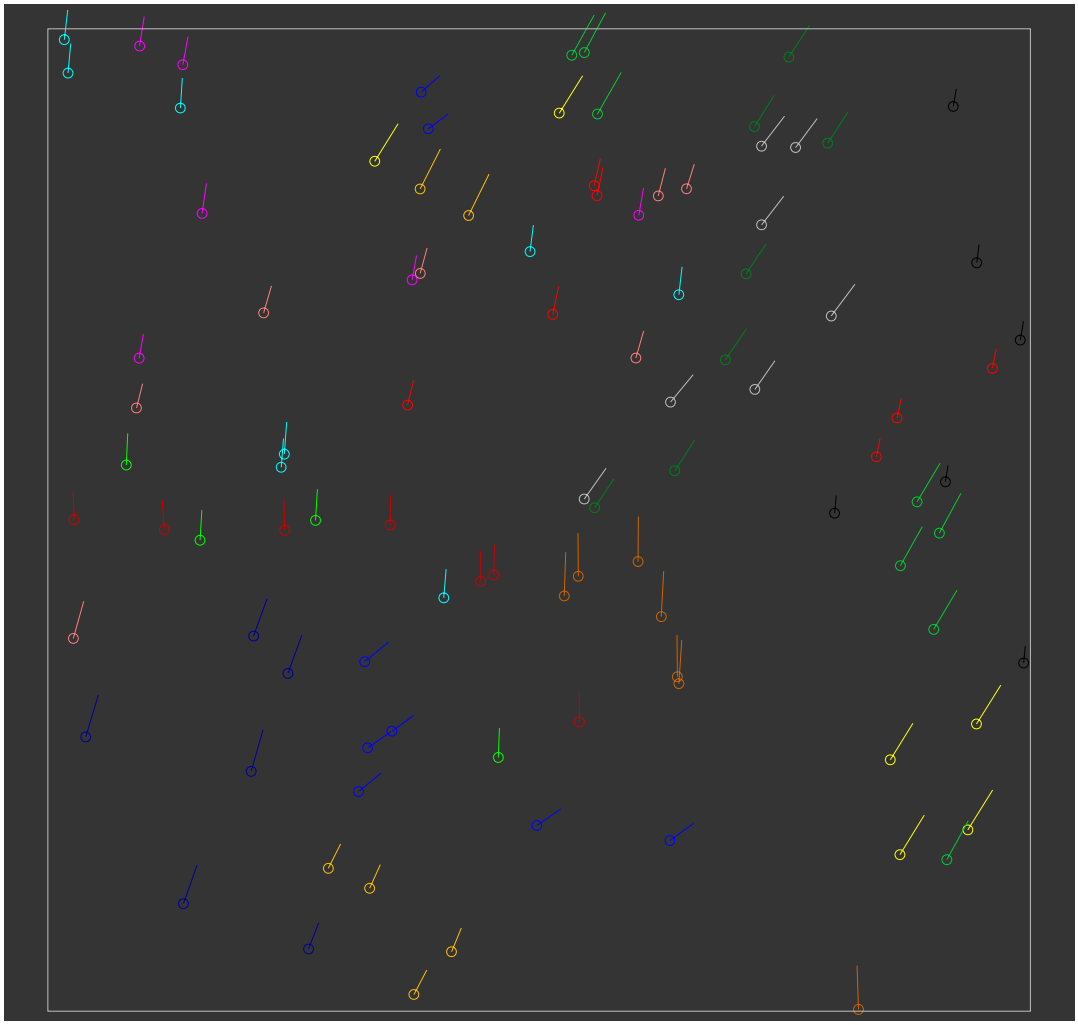
Prvič se pri delu s sto vektorji zgodi, da štirje nevroni ostanejo prazni in tako dobimo le 11 skupin. Nekatere se oblikujejo povsem pravilno. To se zgodi pri skupinah, ki imajo dovolj specifično obliko vektorjev. Skupine 2, 5 in 6 imajo precej podobne oblike, zato so vektorji nekako pomešani. Skupina 4 se je sicer

oblikovala pravilno, vendar pa so svetlo modri člani tudi v skupinah 5 in 6. Izkaže se, da je za pravilen rezultat do neke mere treba upoštevati tudi položaj vektorjev.

Problema bi lahko rešili tako, da bi komponentam vektorjev dodeljeval uteži, ki bi določale stopnjo oziroma moč vpliva posamezne komponente.

Forma 4

Poskusimo še bolj poudariti obliko, tako da uporabimo dx, dy, dolžino in azimut.

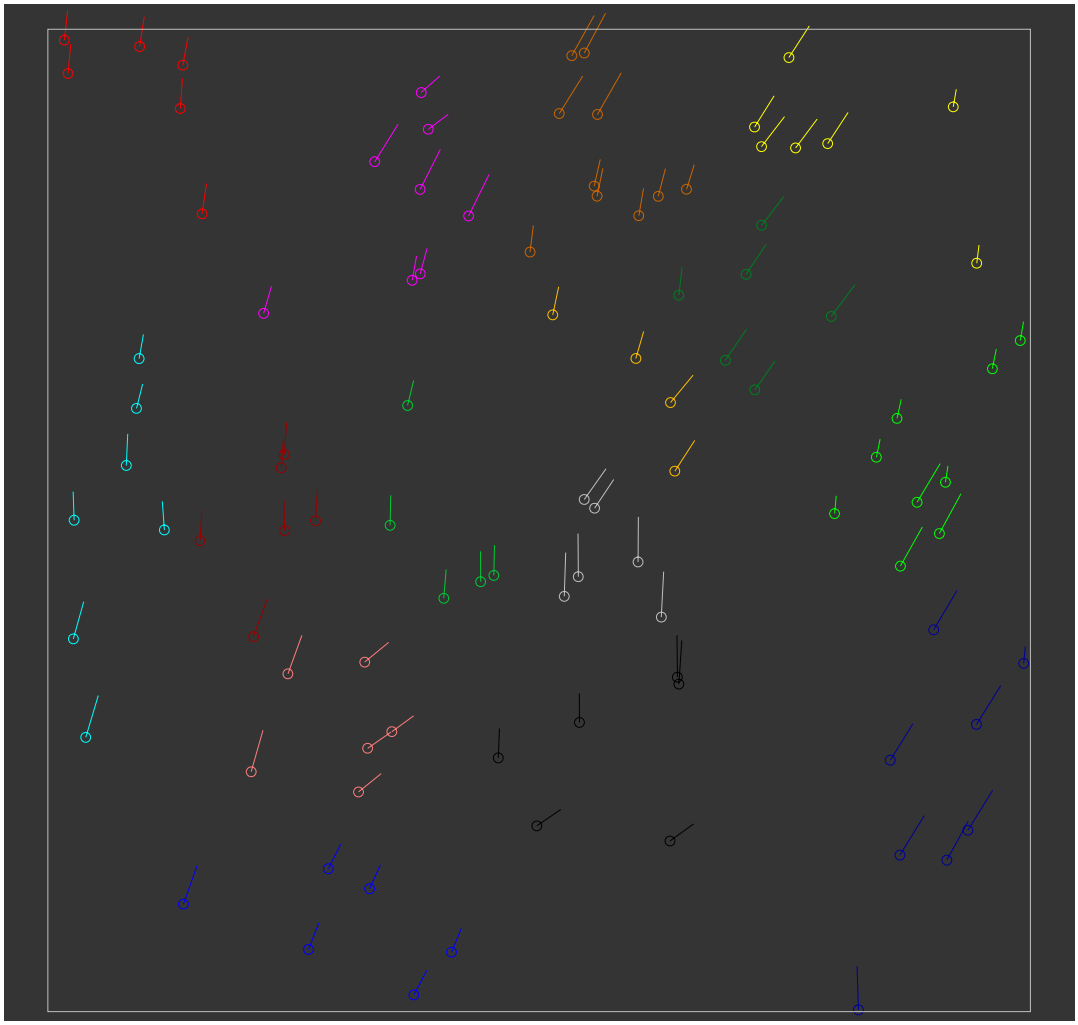


Slika 4.36: Tretja naloga – forma 4

Zaradi večje količine podatkov noben nevron ni več ostal prazen. Posledica je tudi veliko večja razdrobljenost. Pravilno sta se oblikovali le skupini 3 in 10 in to zato, ker imata zelo značilno obliko vektorjev, kar pomeni zelo različno od oblik vektorjev drugih skupin.

Forma 5

S peto obliko zapisa vektorjev poskušam povečati pomen oblike vektorjev premikov, ne da bi izločil podatke o njihovih položajih. Tako uporabim položaja x in y, spremembi dx in dy pa vsako po trikrat. Forma ima obliko (x, y, dx, dy, dx, dy, dx, dy).

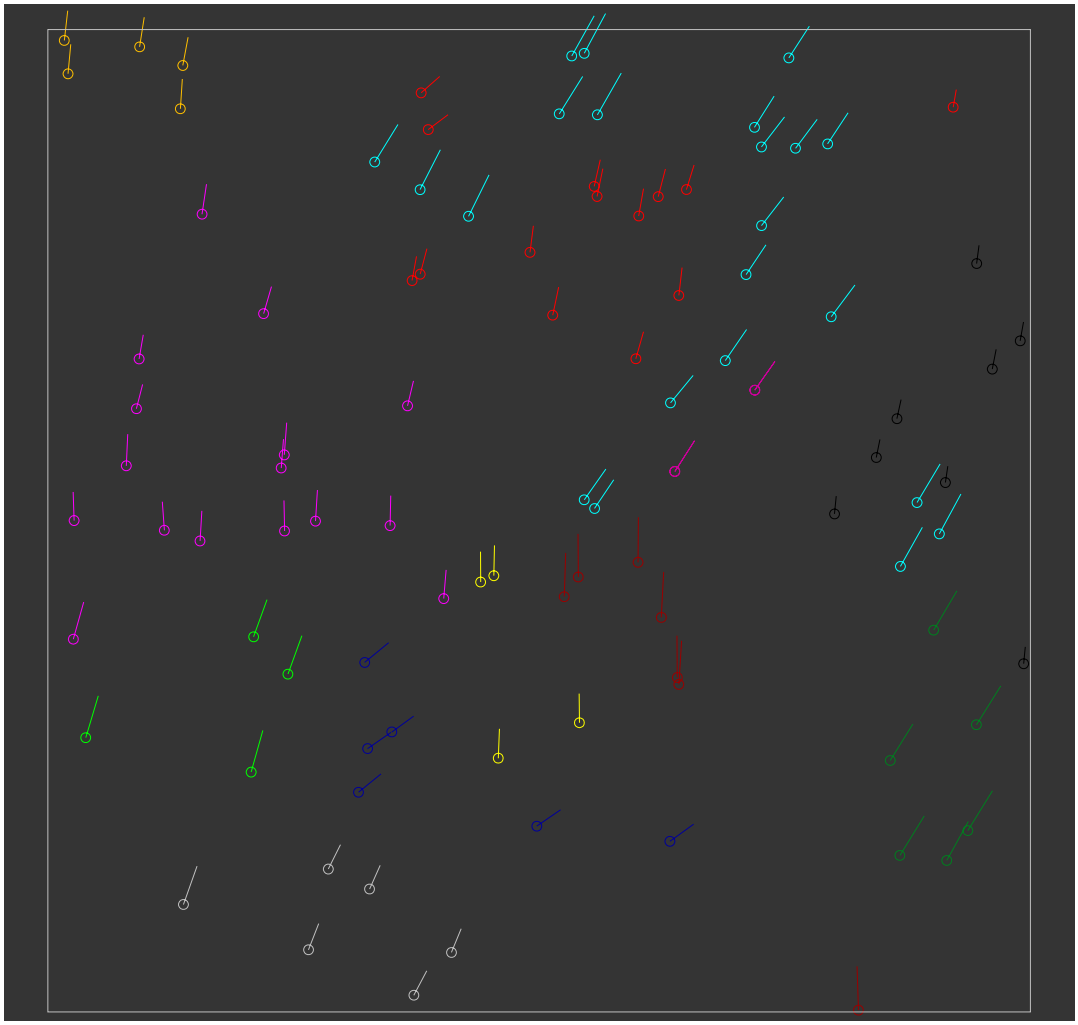


Slika 4.37: Tretja naloga – forma 5

Na prvi pogled rezultat zgleda sprejemljiv, ko pa pogledamo, kakšna bi območja morala biti, vidimo, da je zelo slab. Možen vzrok je, da ležijo vrednosti x in y na intervalu (0,1000), medtem ko vrednosti dx in dy ležijo na intervalih (0,5) in (0,9). Morda tako sama velikost vrednosti daje prevelik pomen položajem. Ta problem bomo reševali z naslednjimi formami, v katerih bomo uporabili postopek normiranja.

Forma 6

S postopkom normiranja želim zagotoviti, da so velikostni razredi vrednosti komponent primerljivi. Najprej poskusim z začetno formo (x, y, dx, dy), le da so podatki normirani.



Slika 4.38: Tretja naloga – forma 6

To je do sedaj gotovo najboljši rezultat. Predpostavka, da uporaba razdalj v hiperprostoru nadomesti normiranje, je očitno napačna.

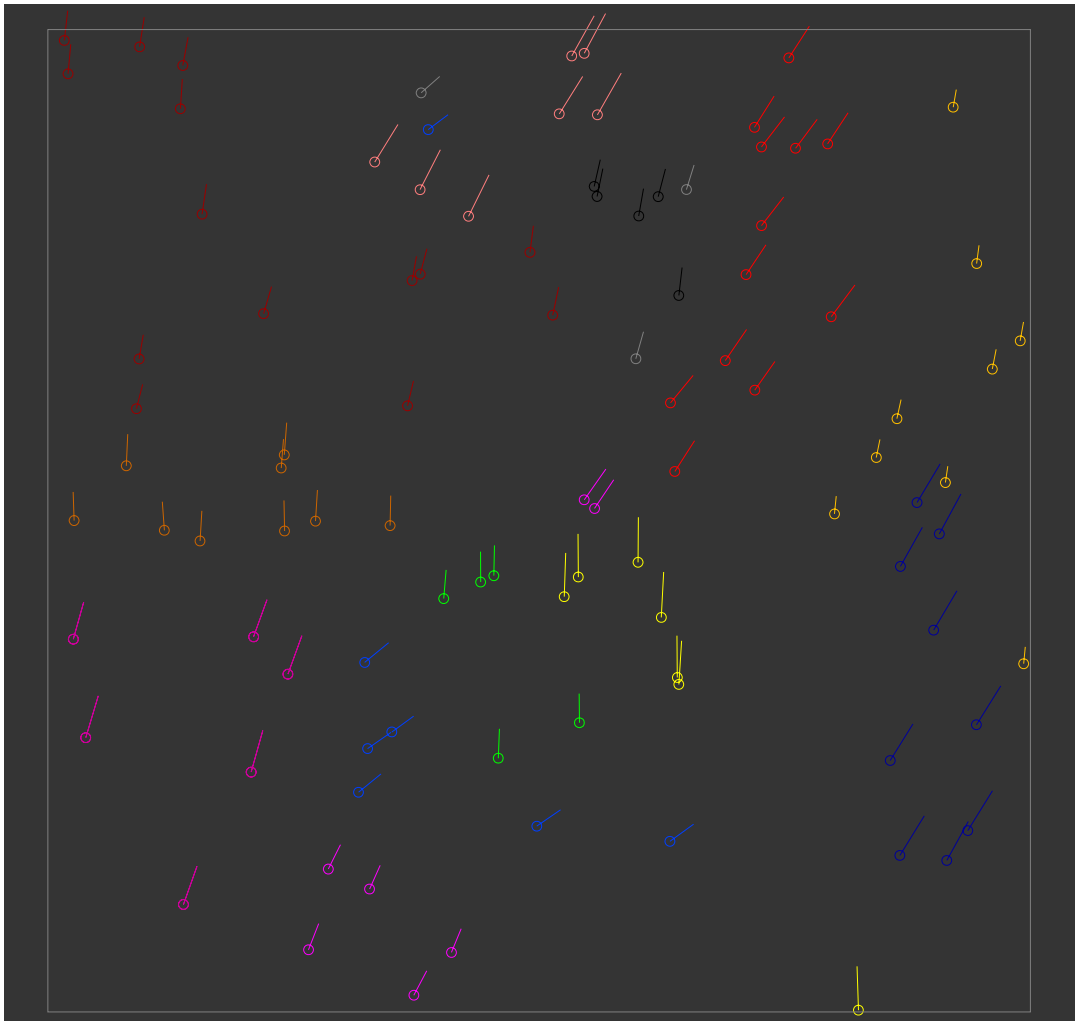
Razen skupin 4 in 5 so vse skupine oblikovane pravilno. Člani skupine 2 oziroma skupine 6 so padli v skupino 4 oziroma v skupino 5. Poleg tega je zanimivo v skupinah 7 in 9; zgleda, da so si člani obeh skupin zelo podobni. Možno je, da sta si

glavna vektorja teh dveh skupin, s katerima sem oblikoval podatke, res zelo podobna. Če bi bila situacija v resničnih podatkih takšna, se ne bi dalo trditi, da gre za dve različni območji.

Pozornost zbujata tudi vektor, ki bi moral ležati v skupini 9, pa je padel v skupino 12 in vektor, ki je ustanovil svojo skupino, ni pa pridobil dodatnih članov iz skupine 4. Takšne osamljene napake bi iz rezultata lahko izključili subjektivno, to pa bi bilo možno narediti tudi z algoritmom.

Forma 7

Čeprav je težko pričakovati boljši rezultat kot s prejšnjo formo, poskusim dodati še dolžino in azimut, oba tudi normirana. Oblika zapisa je tako določena (x, y, dx, dy, dolžina, azimut)$_{norm}$.



Slika 4.39: Tretja naloga – forma 7

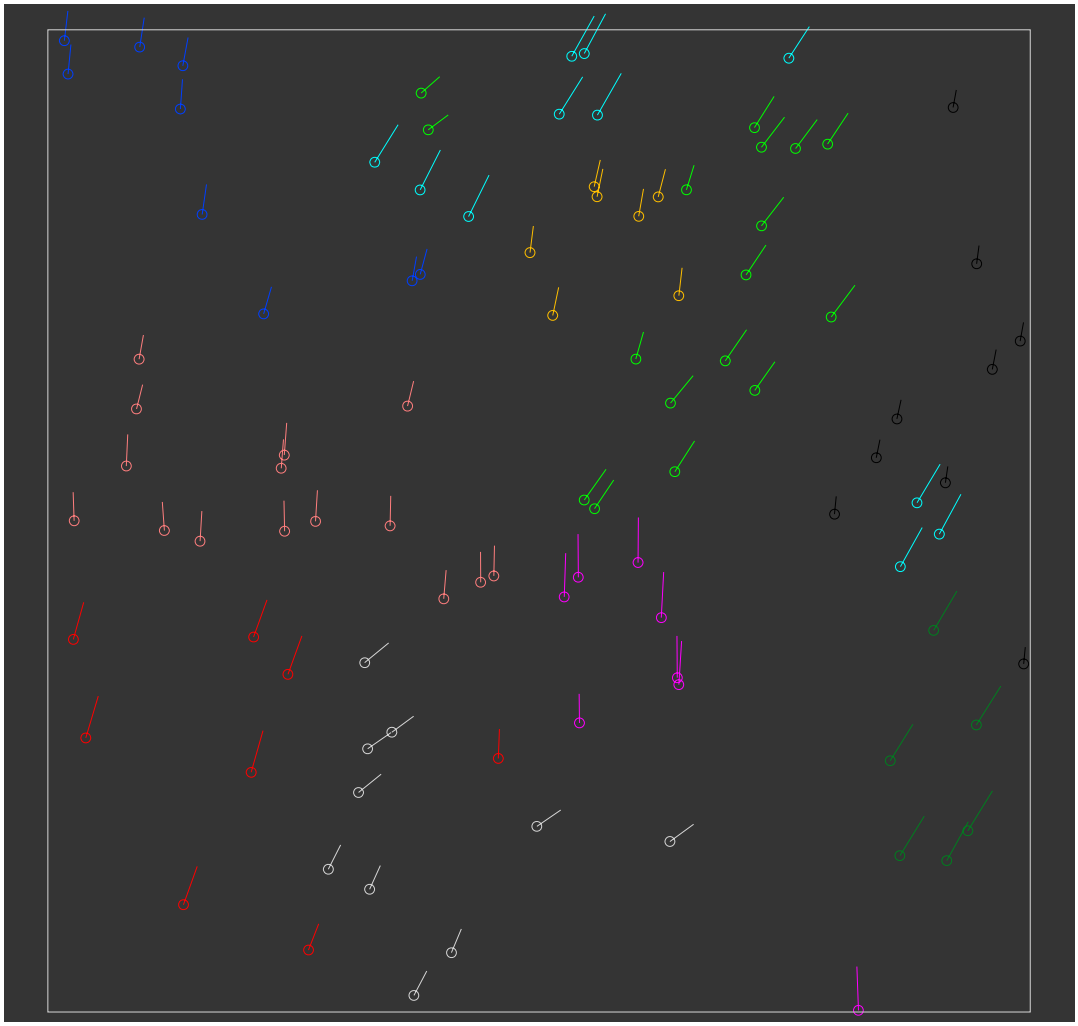
Premoč oblike nad položajem se pokaže s svetlo modrim vektorjem, ki je padel v skupino 8 namesto v skupino 3. Opažam, da se pravilno oblikujejo skupine, ki imajo samosvojo obliko vektorjev (3, 10, 11 in 12) in to kljub temu, da njihovo območje ni dobro zaključeno.

Z upoštevanjem logične topološke zaključenosti območij lahko trdimo, da je skupina sivih vektorjev povsem nesmiselna, prav tako pa je nesmiseln svetlo modri osamelec v skupini 8.

Pri prejšnjem nizu sta ostala prazna dva nevrona, pri tem pa trije. Postopek vsaj približno pravilno določi število skupin.

Forma 8

Zanima me še, kaj se zgodi, če določim pravilno število nevronov, torej namesto petnajst zdaj dvanajst. Oblika zapisa ostaja enaka kot pri formi 7.



Slika 4.40: Tretja naloga - forma 8

Rezultat je slabši kot prej. Kljub temu, da sem mrežo zmanjšal za tri nevrone, sta še dva nova ostala prazna. Pravilno se na novo ni oblikovala nobena skupina.

4.4.5 Komentar tretje naloge

Med izvajanjem naštetih poskusov sem se naučil, da se najbolj splača grozditi normirane podatke, določeno informacijo pa dobimo tudi s podatki, oblikovanimi v tretjo in četrto formo.

V praksi bi uporabne rezultate lahko pridobil le s primerjalno analizo rezultatov form 3, 4, 6 in 7. Še posebej pri formi 6 in formi 7 bi moral biti pozoren na osamele vektorje ter na topološko logičnost območij, ki jih oblikujejo vektorji, določeni isti skupini.

Razen 4. in 5. skupine se je vsaka vsaj v enem od nizov oblikovala pravilno, težko pa je ugotoviti, v katerem od nizov je neka skupina pravilno določena.

Ugotavljam, da je bila tretja naloga dober pokazatelj zmožnosti tekmovalnega sloja nevronske mreže za delo z vektorji hitrosti geodinamičnih premikov.

Postopek je precej objektiven, kar sklepam iz dejstva, da se niti ob pravilnem številu nevronov (želenih skupin) ne vrne točno želen rezultat, ampak takšen, ki najbolje ustreza »logiki« mreže. Pri iskanju rezultata, ki ga sami (subjektivno) želimo, nam seveda taka objektivnost ne pomaga najbolj.

Dvorezni meč je tudi pragovno učilno pravilo. Kot vemo, nam pomaga zapolniti čimveč nevronov, po drugi strani pa teži k skupinam z enakim številom članov. Druga lastnost se mi zdi za naš problem izrazito moteča, medtem ko se pri prvi postavlja vprašanje: Ali bi se pri večjem številu epoh vedno zapolnili vsi nevroni? Večino časa sem vztrajal pri petsto epohah, vendar pa sem nekajkrat poskusil tudi s tisoč epohami. Izkazalo se je, da tudi pri več epohah lahko ostanejo prazni nevroni. To dejstvo po mojem mnenju poudarja objektivnost mreže.

Problem, da mreža teži k oblikovanju skupin z enakim številom nevronov, se v praksi pojavi takrat, ko so skupine zelo različnih velikosti. V takem primeru se bodo večje skupine delile na več delov. Takšne napake postopka bi lahko reševali z

dodatnimi algoritmi, ki bi za vse možne pare skupin izračunal varianco oblike vektorjev ter nato povezal tiste pare z najmanjšo varianco v eno skupino.

Opisana metoda ni popolna, vsekakor pa z njeno pomočjo dobimo določeno predstavo o obliki grozdov oziroma skupin. Potrdila se je predpostavka, da se veliko lažje oblikuje skupina, katere člani so specifični glede na člane drugih skupin; take skupine, kot so 3, 10 in 11, so se namreč največkrat oblikovale pravilno.

Med grozdenjem posameznih form sem si večkrat zaželel, da bi pri pripravi podatkov izdelal med seboj bolj različne skupine; toda to bi bil v resnici le način, s katerim bi lažno predstavil nevronsko mrežo za bolj sposobno, kot v resnici je.

Nekatere skupine so si po obliki svojih članov dejansko med seboj zelo podobne. Takšne so recimo skupine 4, 5 in 6, morda pa tudi 7 in 9. V praktičnem primeru bi bilo za take skupine zelo težko določiti meje, vprašanje pa je, če bi bilo to sploh smiselno; namreč, če so si vektorji res zelo podobni, potem bi to morala biti ena skupina.

Lastnost postopka je tudi ta, da nimamo nikakršnega vpogleda v to, kako različni morajo biti vektorji, da se med njimi oblikuje meja.

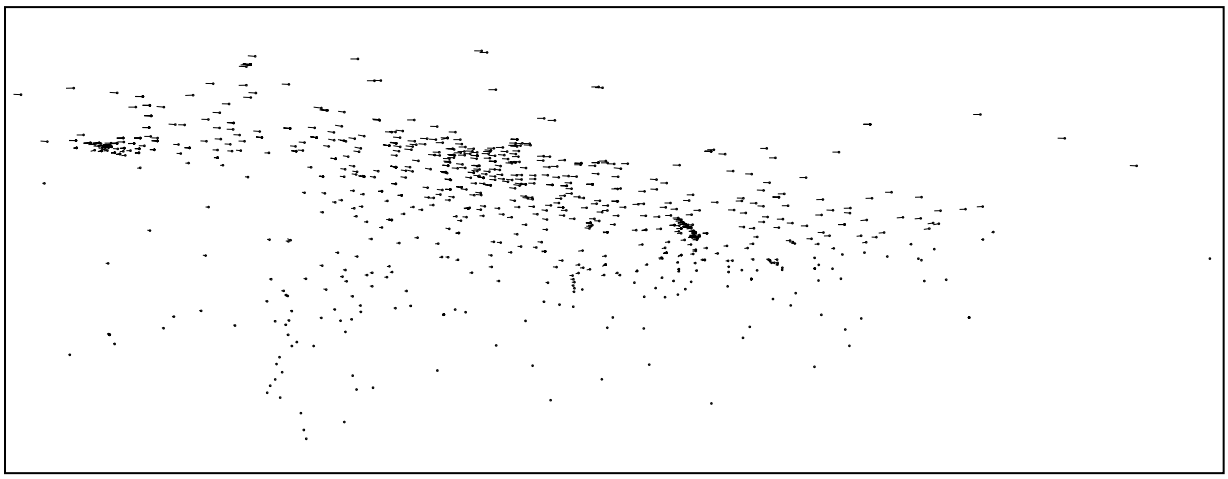
4.5 Naloga 4 – praktični primer – Kalifornija

4.5.1 Opis in priprava parametrov

Ta naloga je praktični preizkus postopka. Podatke predstavlja niz 804 vektorjev hitrosti dejanskih geodinamičnih premikov. Poskušal bom čim bolje določiti meje območij, vendar v okvirih, ki jih postavljajo rezultati grozdenja. Z dodatnim vedenjem o geologiji območja bi lahko svoje rezultate izboljšal in ovrednotil.

Podatki so podani kot geografska širina in dolžina točk ter hitrost premika v vsaki od smeri. Koordinate so krogelne, zato jih lahko uporabljamo kot ravninske brez transformacije. Geografska širina ustreza koordinati y, geografska dolžina pa koordinati x.

Izrisani podatki izgledajo takole.



Slika 4.41: Podatki četrte naloge

Glede na izkušnje iz prejšnje naloge sem se odločil, da bom podatke obdelal v štirih oblikah, ki so bili v prejšnji nalogi označeni kot forme 3, 4, 6 in 7.

Tabela 6: Forme, uporabljene v četrti nalogi

Forma						
1	x	y	dx	dy		
2	x	y	dx	dy	dolžina	azimut
3			**dx**	**dy**		
4			**dx**	**dy**	**dolžina**	**azimut**
5	x	y	3x dx	3x dy		
6	$\mathbf{x}_{norm}$	$\mathbf{y}_{norm}$	$\mathbf{dx}_{norm}$	$\mathbf{dy}_{norm}$		
7	$\mathbf{x}_{norm}$	$\mathbf{y}_{norm}$	$\mathbf{dx}_{norm}$	$\mathbf{dy}_{norm}$	$\mathbf{dolž}_{norm}$	$\mathbf{azi}_{norm}$

Trening mreže dvajsetih nevronov v petsto epohah traja dobrih pet minut. Zato se mi je zdelo smiselno najprej spet preveriti, kako se med učenjem spreminjajo uteži. Mreži sem določil tisoč epoh ter na vsakih sto zahteval izpis uteži. Uteži v mreži je 80, vsak nevron ima namreč štiri vhode za štiri komponente vektorja; stanje uteži torej v nekem trenutku predstavlja matrika v velikosti 20 $\times$ 80. Izpisala se je začetna matrika uteži ter nato še deset matrik, vsaka za nadaljnjih sto korakov učenja. Zaporedni matriki sem odštel med seboj

$$D_i = M_i - M_{i-1} \tag{20}$$

D je matrika, ki nam pove, kako so se spremenile uteži v stotih korakih. Takih matrik je deset. Indeks *i* predstavlja korak stotih epoh in teče od 1 do 10.

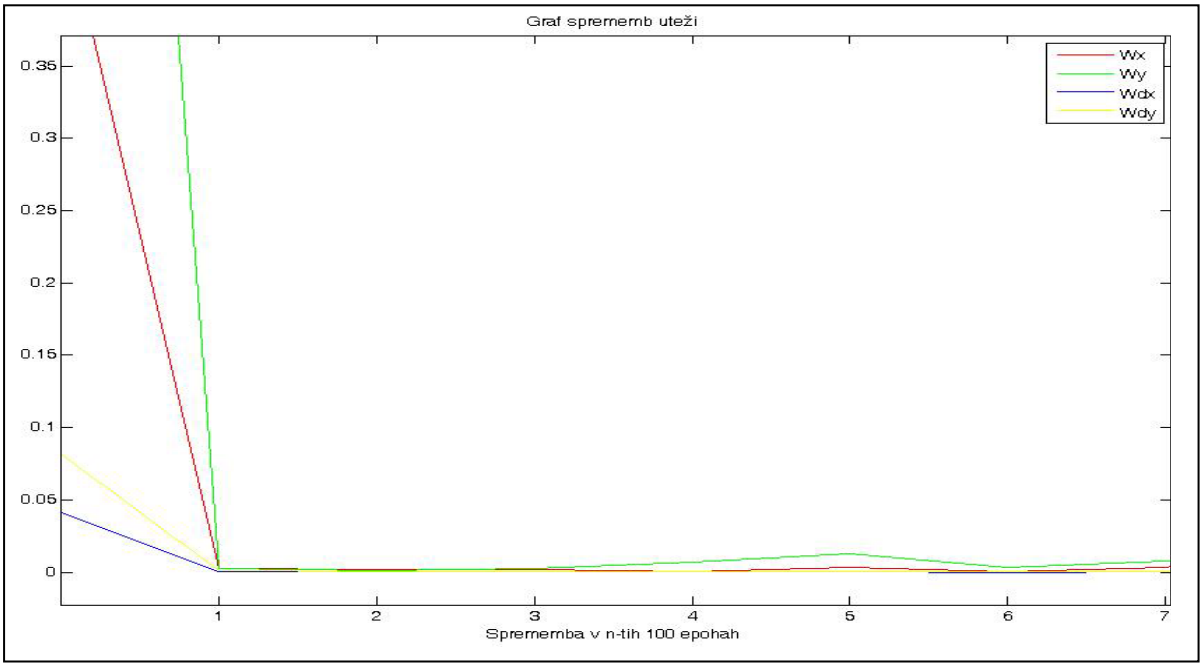
Ker je primerjava matrik dimenzij 20 $\times$ 4 nepregledna, sem za vsako matriko *D* izračunal štiri srednje vrednosti posameznih komponent. Tako je matrika D predstavljena z vektorjem dimenzije 1 $\times$ 4, v katerem vsaka vrednost pove, za koliko se je v stotih epohah povprečno spremenila vrednost uteži x, y, dx in dy. Vrednosti so absolutne.

Dobil sem naslednje rezultate.

Tabela 7: Spreminjanje uteži med učenjem za četrto nalogo

epohe	ΔWx	ΔWy	ΔWdx	ΔWdy
0-100	0.4621	1.4006	0.0414	0.0821
100-200	0.0028	0.0022	0.0001	0.0012
200-300	0.0018	0.0013	0.0001	0.0001
300-400	0.0017	0.0026	0.0000	0.0003
400-500	0.0004	0.0072	0.0001	0.0002
500-600	0.0032	0.0126	0.0002	0.0004
600-700	0.0002	0.0034	0.0000	0.0004
700-800	0.0036	0.0079	0.0001	0.0006
800-900	0.0048	0.0048	0.0001	0.0011
900-1000	0.0007	0.0067	0.0000	0.0002

Želel sem si, da so vrednosti sprememb, ko se povečuje število epoh, vedno manjše, ampak žal ni tako. Izrisal sem graf, ki nazorno pokaže, da se praktično vsa sprememba zgodi v prvih stotih korakih, v nadaljevanju pa vrednosti ne konvergirajo, ampak nihajo okrog določene vrednosti.

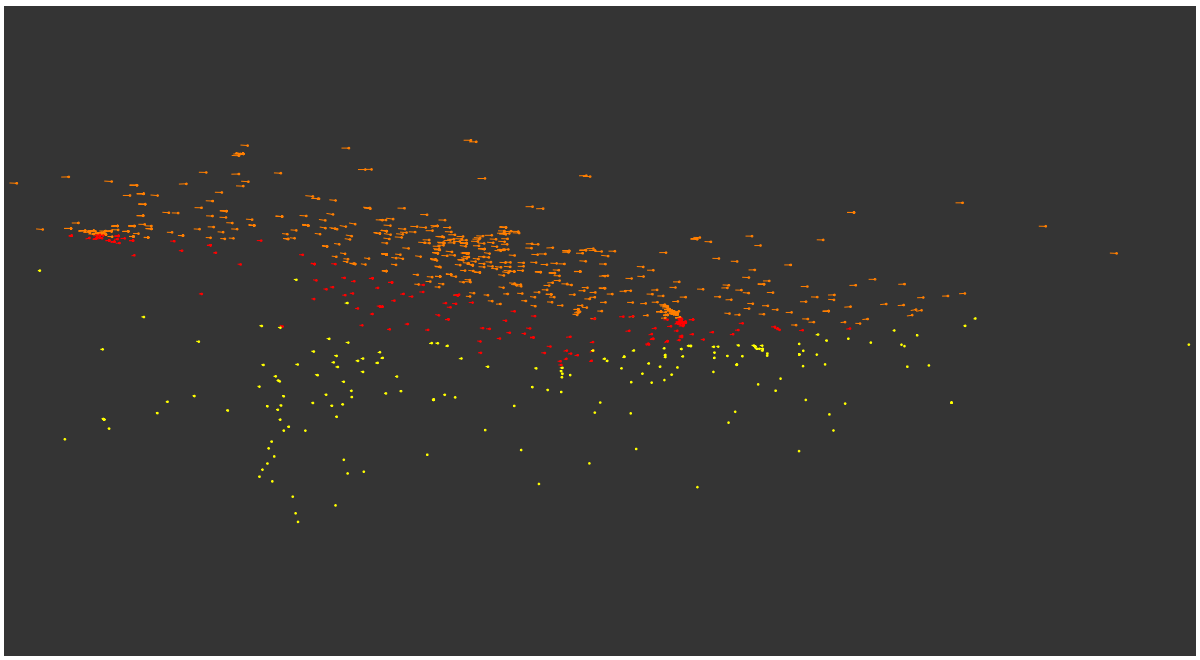


Slika 4.42: Graf spreminjanja posameznih uteži med učenjem

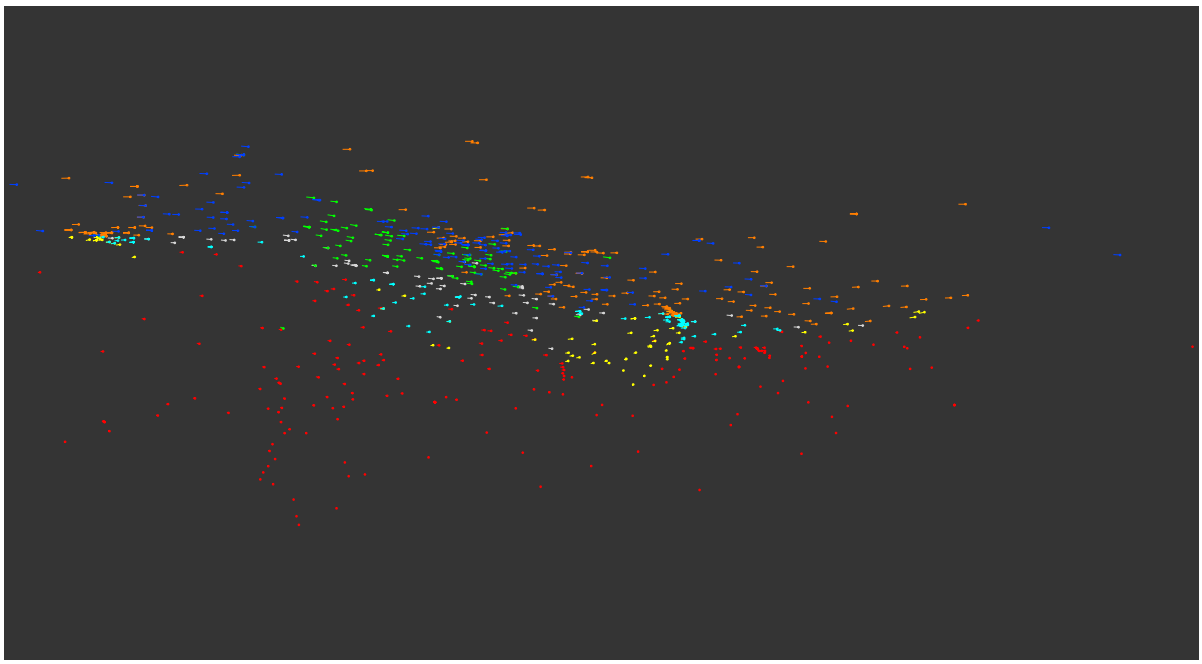
Iz tega sklepam, da je uporaba več epoh nesmiselna; vztrajal bom na petstotih epohah.

4.5.2 Forma 3

Uporabil sem samo podatka o spremembi položaja točke dx in dy, vendar sem poskus ponovil z istima dvema komponentama, le normiranima. V prvem primeru sem kljub dvajsetim nevronom dobil le tri skupine, v drugem pa je nastalo sedem skupin.



Slika 4.43: Forma 3 – četrta naloga

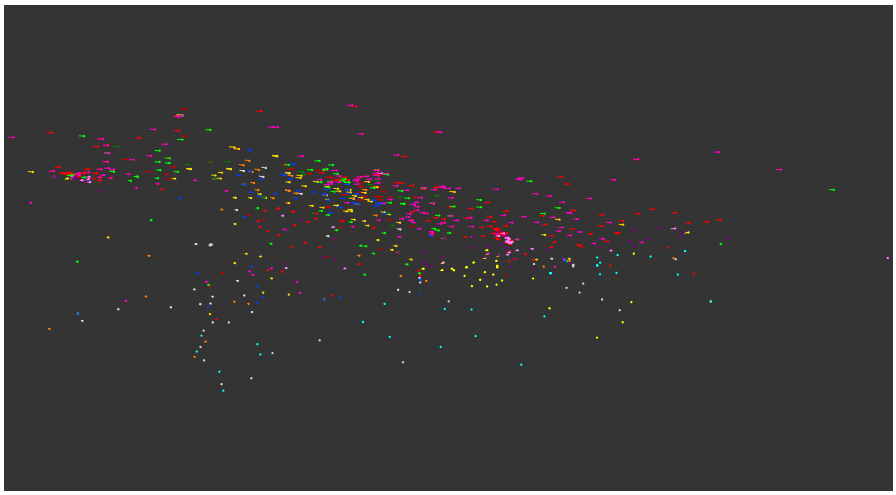


Slika 4.44: Forma 3, normirana – četrta naloga

Kot se spomnimo iz tretje naloge, forma 3 ni prinesla pravih rezultatov, vendar pa je bilo veliko mej pravilno določenih. Kljub temu, da so točke iste skupine lahko ležale v več otokih, so meje otokov v določeni meri predstavljale prave meje. Dobljenih dveh rezultatov torej ne bomo vzeli za pravilna, vendarle pa nam bosta pomagala poiskati meje območij.

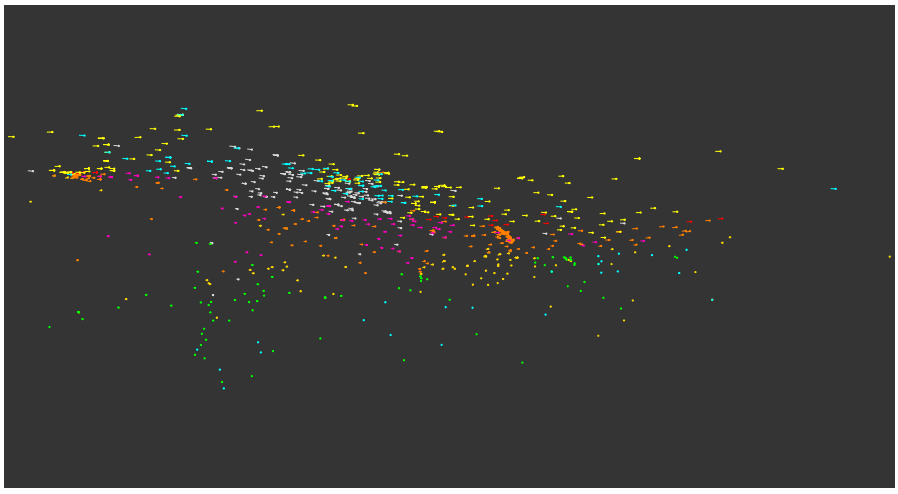
4.5.3 Forma 4

Forma 4 je od izbranih še najmanj uporabna. Pri veliki količini podatkov, ki nastopajo v danem primeru, je topološka neurejenost skupin veliko bolj moteča kot pri prejšnji nalogi. Točke, ki so pri tej nalogi padale iz glavne gmote grozda, smo hitro prepoznali in jih ignorirali, tukaj pa se skupine dobesedno pomešajo med seboj. Poglejmo.



Slika 4.45: Forma 4 - četrta naloga

Za normirane vrednosti pa dobim

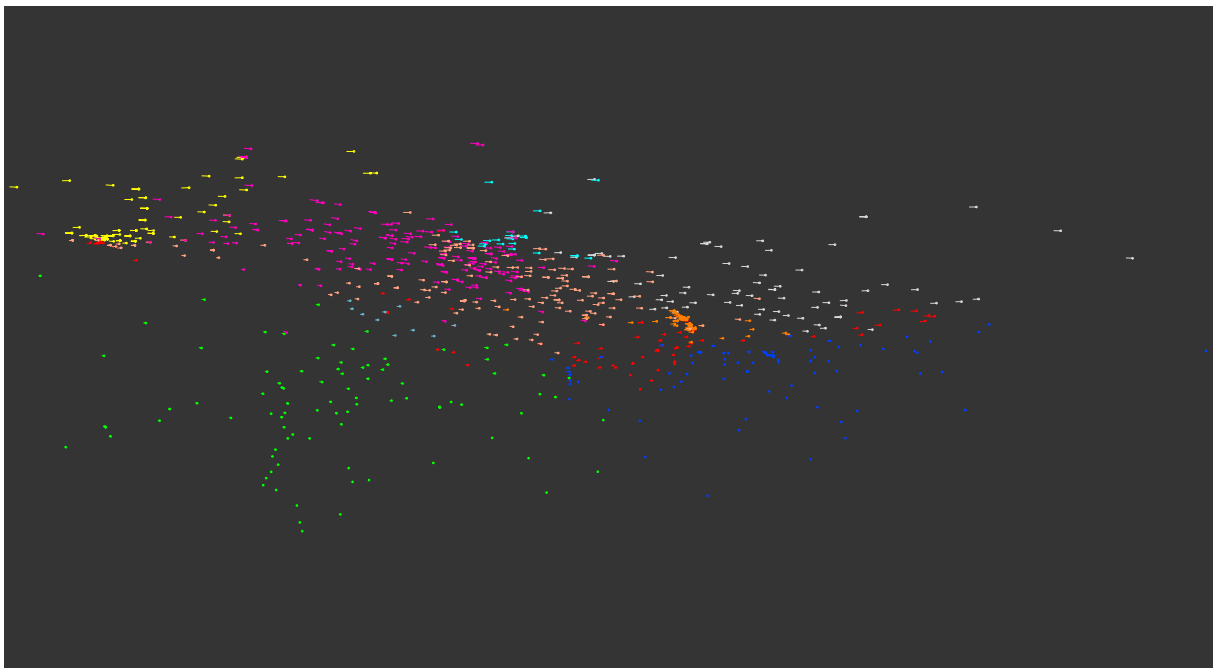


Slika 4.46: Forma 4, normirana - četrta naloga

Kaže, da mi ta niz ne bo v veliko pomoč; morda pa ga bom le lahko uporabil tam, kjer bom imel probleme z odločanjem.

4.5.4 Forma 6

V formi 6 uporabim normirane vrednosti komponent x, y, dx in dy. Ta forma se je v prejšnji nalogi izkazala za najboljšo, zato jo bom tudi v tem primeru imel za najbolj pravilno.



Slika 4.47: Forma 6 - četrta naloga

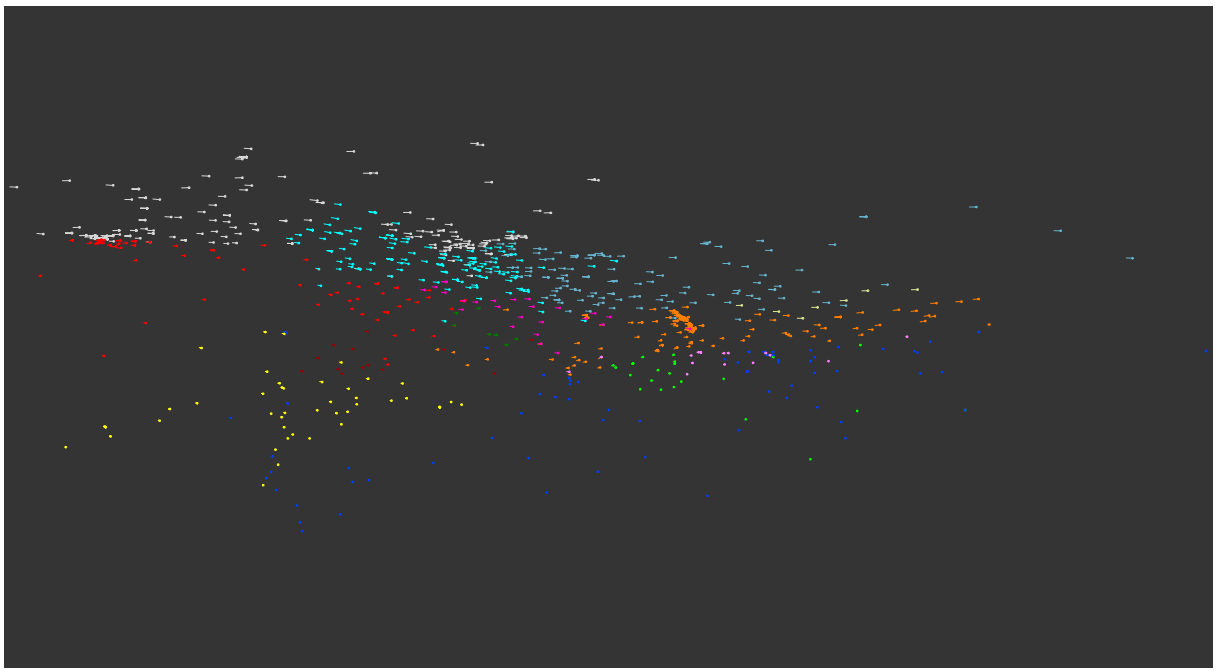
Zanimivo je, da se je kljub dvajsetim nevronom oblikovalo le deset skupin.

Zaradi mešanja skupin bo verjetno treba določene skupine povezati skupaj, še posebej tiste, ki imajo očitno podobno obliko vektorjev in so se formirale bolj zaradi različnega položaja.

Oblikovanja skupin se bom lotil v naslednjem delu poglavja.

4.5.5 Forma 7

V tej formi nastopajo iste komponente kot v formi 6, le da sta dodani še komponenti dolžine vektorjev in azimutov, seveda tudi normirani. Teoretično naj bi forma 7 v primerjavi s formo 6 bolj poudarila obliko vektorjev pred položajem, kar pa se bo morda izkazalo za negativno, če upoštevamo, da je že pri nizu 6 prišlo do precejšnega mešanja skupin.



Slika 4.48: Forma 7 - četrta naloga

Menim, da je rezultat bolj uporaben kot rezultat pri formi 6, kajti oblikovalo se je več skupin, prekrivanje pa se je pojavilo samo v zgornjem pasu. Zgleda, da zgornji pas predstavlja eno skupino, vendar se vedno razdeli na nekaj delov, in to zaradi zelo velike razpotegnjenosti v smeri y oziroma geografske dolžine.

4.5.6 Forma x

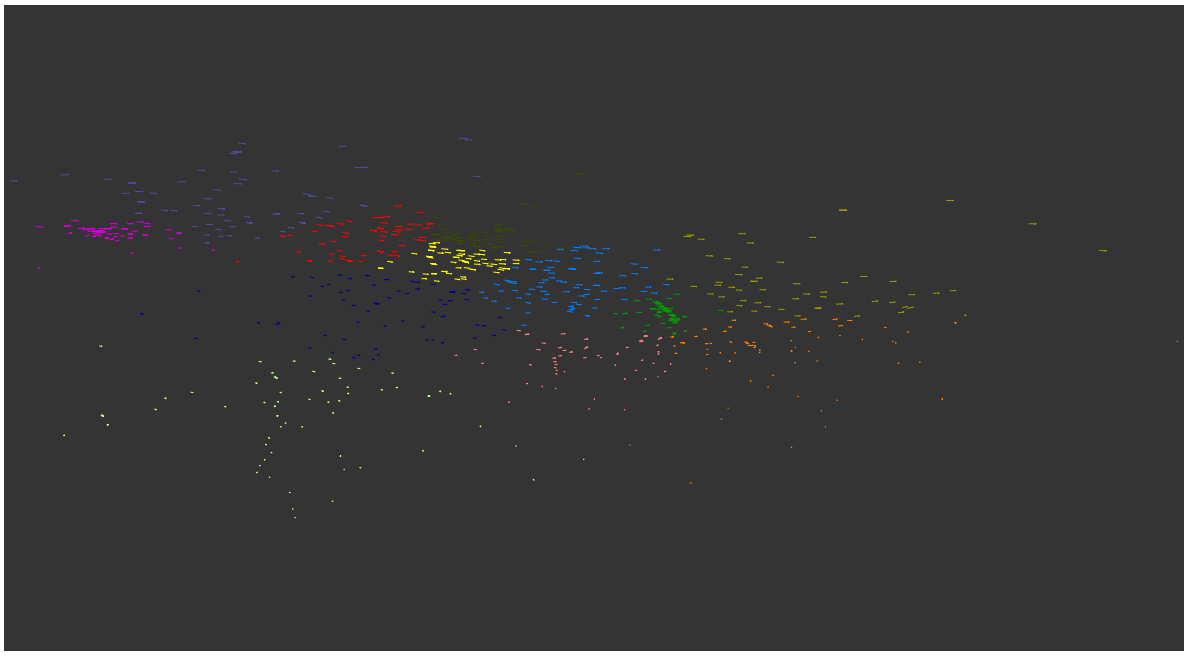
Med učenjem o delovanju in preizkušanjem različnih načinov ter možnosti nevronskih mrež sem dobil neki rezultat, ki bi bil na tem mestu morda zanimiv oziroma uporaben za končno analizo meja območij.

Rezultat sem pridobil iz forme, ki sem jo ustvaril kot nepopoln poizkus normiranja. Podatke sem centriral, tako da sem jim odštel srednjo vrednost pripadajoče komponente. Nato sem želel podatke še skrčiti v x in y smeri, tako da bi zasedali kvadrat s stranico 10 × 10. Tako posamezne komponente nisem delil s standardno deviacijo, ampak s faktorjem f, ki je

$$f = \frac{x_{\max} - x_{\min}}{a} \tag{21}$$

kjer sta x_{max} največja in x_{min} najmanjša vrednost posamezne komponente, a pa stranica kvadrata, v katerega naj se podatki skrčijo. Razpon podatkov je tako določen glede na skrajni meji posamezne komponente in ne glede na njeno razpršenost.

Mislim, da je to dosti bolj primerno normiranje, kot je tisto, ki sem ga uporabljal v vseh prejšnjih postopkih, vendar se mi zdi rezultat, ki sem ga dobil s to metodo, tudi zelo zanimiv. Meje med območji so se zelo dobro oblikovale, kar kaže na relativno velik vpliv položajev, vendar je rezultat kljub temu veliko manj dvoumen in enostavnejši za interpretacijo. Žal pa je hkrati verjetno tudi precej bolj napačen.



Slika 4.49: Forma x - četrta naloga

Pri skupinah na zgornjem robu območja (sivomodra, rdeča, temno zelena, rumena, svetlo modra in rjavozelena) ni nobene bistvene razlike v oblikah vektorjev. Edini razlog za razmejitev je položaj. Toda če zaupamo v neko »višjo logiko« nevronske mreže, lahko trdimo, da gotovo obstaja razlog, da so se skupine oblikovale tako, kot so se.

4.5.7 Analiza meja

Tu bom poskušal iz navedenih rezultatov zaključiti, kje je smiselno sklepati na meje med območji.

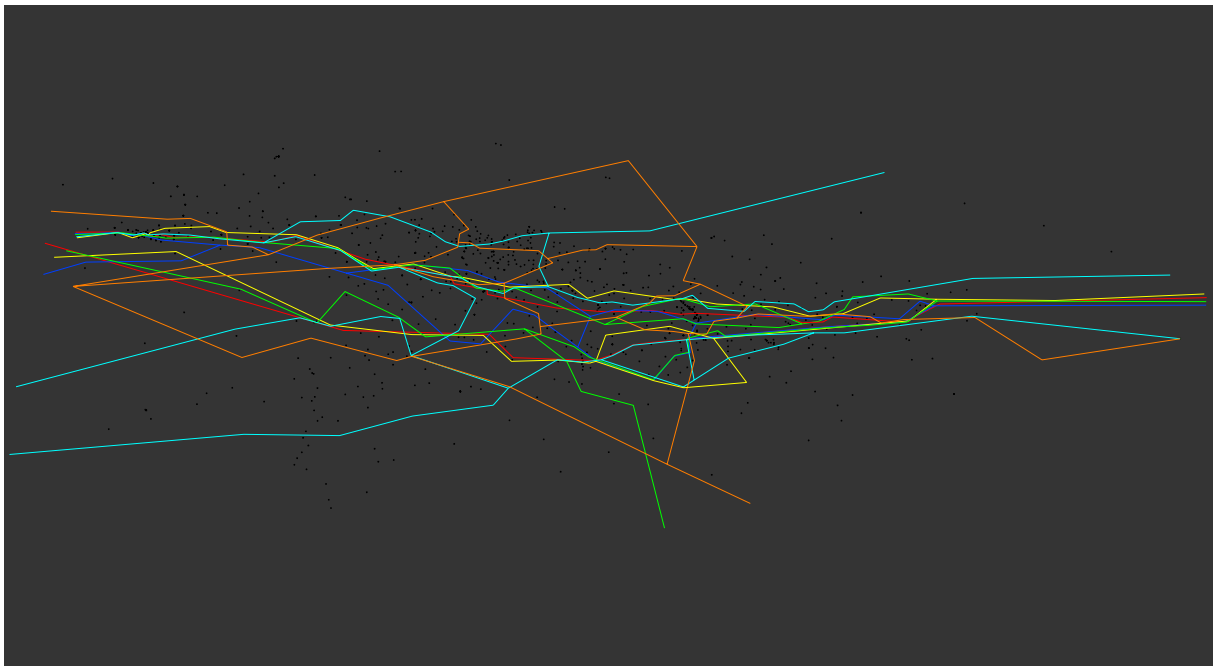
Dejansko je to, kar bom poskusil narediti »pljunek v obraz« namenu te diplomske naloge: želim pokazati, kako to delo lahko opravi nevronska mreža, v končni fazi pa bomo za dokončanje naloge spet uporabili človeško logiko.

Kakorkoli že: smiselno se mi zdi, da delo dokončam in predstavim optimalno rešitev problema, čeprav ta ne bo rezultat povsem objektivne računalniške logike.

Najprej sem na sliki vsakega rezultata narisal meje med območji, ki so se oblikovala. V nekaterih primerih je bilo to zelo težko. Na splošno sem upošteval

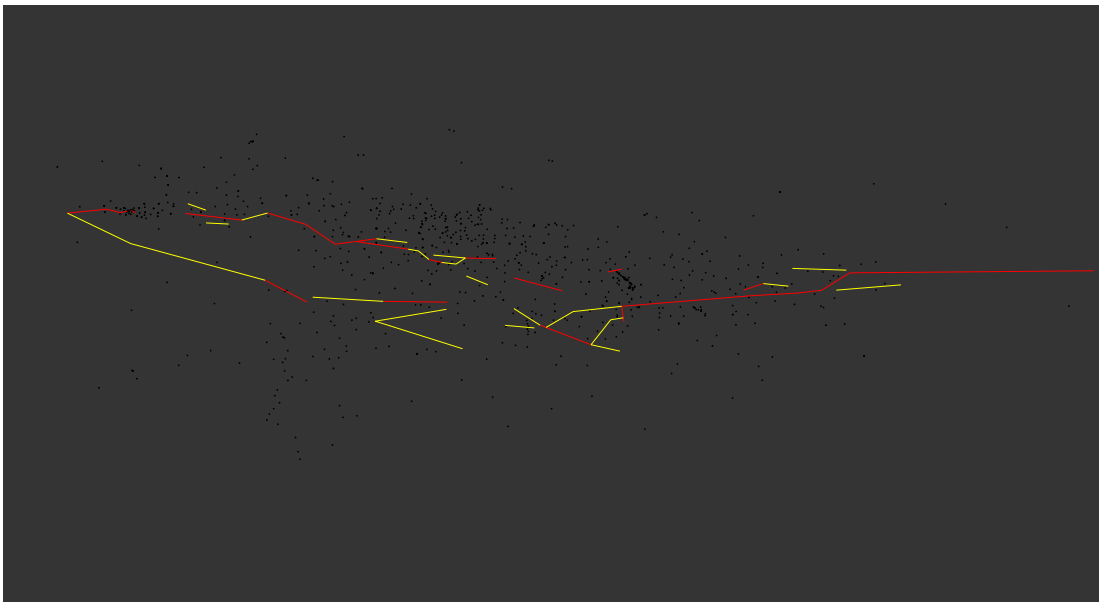
pravili, da ignoriram osamelce ter da upoštevam topološko zaključenost območij. Pri prekrivanju grozdov sem včasih predpostavil, da dve barvi pomenita eno skupino, kjer pa je bilo prekrivanje manjše, sem oblikoval mejo.

Sedaj želim na eni sliki izrisati vse dobljene meje in pogledati, če se kje prekrivajo oziroma ali lahko iz njih razberem kaj smiselnega. Za podlago bom vzel izris točk brez smeri premikov. Želim biti čimbolj objektiven, vendar vseeno potrebujem ogrodje, da vidim, ali dve meji pomenita mejo med istimi vektorji, čeprav se geometrično ne pokrivata.



Slika 4.50: Analiza meja - prvi korak

Stvar na prvi pogled zgleda precej zmedeno, zato se je moram lotiti sistematično. Meje, na katerih sovpadajo tri črte ali več, bom označil z rdečo barvo, tiste, kjer sovpadata dve, pa z rumeno.

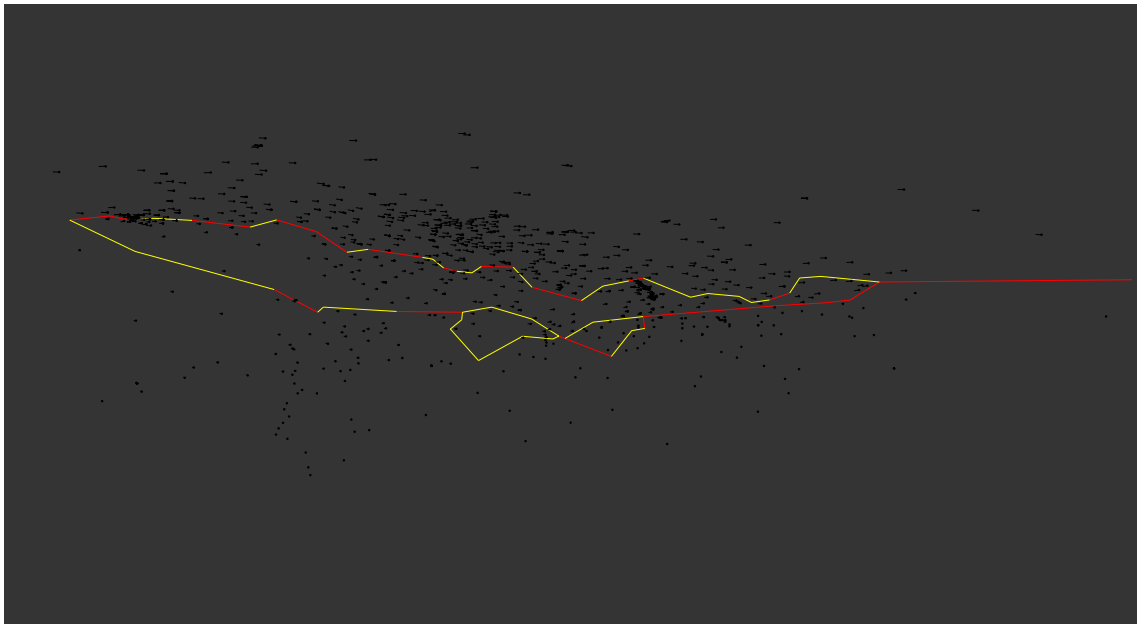


Slika 4.51: Analiza meja - drugi korak

Sedaj bom začel zaključevati območja. Kjer rumene črte od celotnega območja razmejijo samo enega ali dva vektorja, bom te črte odstranil.

Ker predpostavljam, da sta formi 6 in 7 boljši od vseh drugih, bom območja zaključeval na podlagi meja teh dveh form, če pa se bo še kje pojavil dvom, si bom pomagal še z drugimi mejami.

Končni rezultat z vključenimi oblikami vektorjev zgleda nekako takole.



Slika 4.52: Analiza meja - tretji korak

4.5.8 Komentar četrte naloge

Pri delu s konkretnimi podatki sem lahko uporabil izkušnje, ki sem jih dobil pri obravnavanju testnih podatkov. Izkazalo se je, da formi 3 in 4 sicer pomagata pri obdelavi, vendar se nanju ne moremo preveč zanesti; v veliko pomoč sta le pri odločitvah o pomembnosti meja, ki se večkrat pojavljajo. Glavna rezultata, na katera se lahko bolj zanesemo, sta rezultata, dobljena s formama 6 in 7. Dejstvo pa je, da mora biti končni rezultat nekakšna zmes vseh vmesnih form.

Pri analizi meja se je pojavljal problem, da sem enake meje med točkami zarisal na različne načine, zato je bilo določanje objektivnega poteka meja naporno. Dobil sem idejo, da bi stvar lahko poenostavil z uporabo Vornoievih poligonov. Gre za to, da med vsakim parom sosednjih točk zarišemo mejo, ki je simetrala teh dveh točk. Na ta način dobimo topološko urejeno mrežo, ta pa nedvomno definira območje, ki je najbližje posamezni točki. Meje med območji bi nato lahko vlekel izključno po tem poligonu in tako bi se izognil dvoličnosti meja.

Menim, da je rezultat analize optimalen, kar se je potrdilo tudi po ogledu izrisanih vektorjev znotraj končnih meja. Rezultat je dober, ker ni povsem subjektivna ocena človeka, ampak temelji na nekih objektivnih rezultatih. Če bi človek le na videz določal meje med območji, bi po mojem mnenju dobil drugačen rezultat, za katerega pa ne bi imel nobene utemeljitve.

5 UGOTOVITVE

V tem poglavju želim na kratko opisati, kaj sem se naučil pri delu z nevronskimi mrežami.

Nevronske mreže so mogočno orodje, vendar pa je njihova uporaba v konkretnih primerih zelo kompleksna. Vsak problem zahteva določeno mero poznavanja problematike, še bolj pa delovanje nevronske mreže, ki jo želimo uporabiti. Mogoče je uporabiti tudi predhodno pripravljene programske pakete, ki pa so ponavadi »črna škatla«. Če želimo poleg rezultata ponuditi tudi utemeljitev oziroma prednosti, slabosti in pasti uporabljene metode, se moramo čim globlje potopiti v sistem delovanja metode.

V svojem diplomskem delu sem si prizadeval, da bi čimbolje razumel, kako mreža deluje. Vendar, če sem iskren, sem določen del algoritma pustil zaprtega v črni škatli. Na tisti stopnji se mi je zdelo bolj smiselno algoritem enostavno preizkusiti z različnimi oblikami podatkov, kot pa v podrobnosti raziskati delovanje funkcij *newc* in *train*.

Nevronske mreže glede na njihovo uporabnost lahko delimo na mreže za aproksimacijo, klasifikacijo in grozdenje. To pa še ne pomeni, da glede na problem izberemo eno od treh mrež in da nam ta vrne rezultat. Za postopke razvrščanja objektov v skupine sta najbolj znani mreži *SOM* in ART.

Slednje nisem obravnaval poglobljeno, ker sta pri njej bistvena sposobnost obravnave velikih količin podatkov in pomnilni prostor za ustvarjanje novih skupin, če se za to seveda pojavi potreba. Poleg tega metoda ART zahteva določene predhodne informacije o tem, kako različni elementi smejo pripadati isti skupini. Tega podatka mreži nisem želel podati, ker se mi zdi bistveno, da to ugotovi sama.

Mrežo *SOM* sem si ogledal podrobneje, vendar se mi njeni rezultati na splošno ne zdijo primerni za moj problem, zelo primeren pa se mi je zdel postopek

tekmovalnega učenja, ki ga uporablja; ta postopek sem nazadnje tudi uporabil za svoje delo.

Pokazal sem, da se vektorje da razvrščati v skupino z nevronsko mrežo, vendar so rezultati nepopolni. Izkazalo se je, da mreža vrne popoln rezultat le v primeru, ko so vektorji iz različnih skupin zelo različni med seboj in da poznamo pravo število skupin. Prvi pogoj ni odvisen od uporabnika - v mojem primeru je odvisen od »matere« Zemlje. Drugi pogoj je morda lažje rešljiv, vendar gotovo zahteva človekovo odločitev. Mislim, da je še vedno bolj objektivna človekova analiza rezultate mreže, kot pa poseganje v vhodne podatke.

Zdi se mi, da bi bilo vredno poskusiti še s spektralnim grozdenjem. Od vseh metod, opisanih v drugem poglavju, se mi zdi ta za obdelavo vektorjev hitrosti geodinamičnih premikov še najbolj perspektivna.

Popolna objektivnost pri delu z nevronsko mrežo ni mogoča. Vedno je treba podati vsaj velikost mreže in število epoh učenja, vendar sem prepričan, da ti dve stvari zelo malo vplivata na rezultat. Število nevronov naj bo nastavljeno na večjo vrednost, kot pričakujemo oziroma želimo skupin, za ugotovitev števila epoh pa je po mojem smiselno, da za posamezen problem opazujemo velikosti sprememb uteži. Kakorkoli že, prepričan sem, da preveliko število epoh ne vpliva na rezultat.

Tako je objektivnost postopka v veliki meri zagotovljena, vendar pa hkrati trdim, da takšna rešitev problema ni pravilna. Še vedno so nujni določeni postopki primerjave in združevanja rezultatov različnih variant vhodnih podatkov. Te postopke bi se morda dalo avtomatizirati, morda pa je še bolje, da jih opravi človek.

Rezultat, ki ga vrne nevronska mreža v mojem postopku, je odvisen tudi od števila točk, ki jih hkrati obdeluje, ter od porazdelitve gostote točk. Opazil sem, da je pri številu točk, manjšem od dvajset, zelo pomembno, da mreži podamo toliko nevronov, kolikor mora nastati skupin. Mislim, da je za tako majhne količine točk veliko bolj učinkovito, da grozdenje naredimo na pamet.

Izkazalo se je, da zgoščine točk v podatkih povzročijo formacijo svoje skupine. Ta grozd lahko dejansko pomeni svojo skupino, lahko pa je del okolice. Najverjetneje se je formiral zaradi pragovnega učilnega pravila. Večja skupina objektov z zelo podobnimi lastnostmi (tako položaj kot oblika) ima enostavno večje možnosti za ustanovitev svojega grozda. Problem rešujemo enostavno tako, da primerjamo oblike vektorjev iz zgoščine z oblikami okoliških vektorjev; zgoščino nato dodamo skupini z najbolj podobnimi vektorji, če pa se od vseh sosedov razlikuje, jo pustimo takšno, kot je.

Pri delu z 804 vektorji sta se pojavili dve takšni zgoščini, ena povsem na levem delu območja, druga pa na desni tretjini. Leva se je skoraj pri vseh grozdenjih razdelila na zgornji in spodnji del. Medtem je desna večkrat ustanovila svojo skupino, vendar je ta ponavadi imela »podružnice« tudi po drugih delih območja. Zato sem po analizi sklenil, da zgoščina pripada velikemu predelu, ki se razteza po sredini območja.

6 SKLEP

Sklepam, da sem glavni namen, ki sem si ga zadal v uvodu, izpolnil, kakor se je najbolje dalo. S pomočjo nevronskih mrež sem želel oblikovati skupine podobnih vektorjev geodinamičnih premikov. Uspelo mi je odkriti metodo, ki zna oblikovati skupine in jo v veliki meri preizkusiti ter ovrednotiti.

Trudil sem se čimbolj izključiti vpliv človekovega odločanja. Edina podatka, ki ju mora uporabnik mreži podati, sta število nevronov in število epoh učenja. Med delom sem ugotovil, da ta dva parametra relativno malo vplivata na končni rezultat, zato mislim, da je objektivnost samega orodja zagotovljena.

Problem pa nastane pri tem, ker nobena od preizkušenih oblik oziroma form podatkov ni vrnila objektivnega rezultata. Najboljši približek sem dobil s primerjavo rezultatov različnih form. Objektivnost postopka je zato na tej točki pod vprašajem. Menim pa, da bi bilo mogoče razviti tudi algoritme, ki bi opravili to analizo in s tem zagotovili objektivnost postopka.

Ena od morebitnih slabosti predstavljene metode je ponovljivost. Med samim delom na ta problem še nisem pomislil, vendar se mi zdi vreden omembe. Vprašanje je namreč, ali mreža pod povsem enakimi pogoji (parametri mreže, vhodni niz podatkov) vedno vrne enak rezultat. Do neke mere so si rezultati ponovitev gotovo podobni, morda pa se nekoliko razlikujejo predvsem na mejnih območjih, ki so za nas najbolj zanimiva. Do tega vprašanja sta me pripeljali dve dejstvi. Prvo je »lastna volja« mreže, ki smo jo pri določanju števila grozdov ocenili za uporabno. Drugo dejstvo pa je, da uteži nevronov, ki smo jih opazovali med učenjem, ne konvergirajo. Torej so gotovo tudi pri vsakem ponovnem učenju mreže različni.

V uvodu sem postavil tri hipoteze, na katere sedaj lahko odgovorim.

1. Nevronska mreža je sposobna razvrstiti objekte v skupine na podlagi njihovih lastnosti. Ostaja pa vprašanje, zakaj je oblikovala takšne skupine, kot jih je, in ali je to pravi in edini možni odgovor.

2. Ali lahko najdemo metodo, ki bo povsem splošna in uporabna za vse mogoče nize podatkov? Odgovor je ne. Vedno bomo morali, če že nič drugega, izbrati število nevronov. Menim pa, da je k splošnosti metode močno pripomogel postopek normiranja. Ne glede na velikostni razred posamezne komponente vhodnih vektorjev se z normiranjem velikost vpliva vseh izenači.

3. Rezultat oziroma odgovor mreže, kakšne naj bodo skupine, je sicer nedvoumen, vendar pa ni edini možen. Z različnimi formami podatkov pridemo do različnih rezultatov, zato je nemogoče reči, katera rešitev je pravilna. V nalogah, kjer bi morali z veliko gotovostjo poznati rezultat razvrščanja objektov v skupine, ne priporočam uporabe nevronske mreže. Postopek pa lahko odlično služi kot povsem nov pogled na obravnavano tematiko.

V povezavi z zadnjo hipotezo je morda smiselno omeniti še določena dejstva, o katerih do zdaj še ni bilo govora. Geologija kot znanost na splošno že pozna rešitve problemov, ki sem jih reševal v tej nalogi. Pomanjkljivost teh rešitev je pretežna subjektivnost, velika in morda prevladujoča prednost pa je globlje poznavanje narave samega problema.

Kljub temu, da metoda, ki sem jo obravnaval v nalogi, ni najbolj zanesljiva in morda niti ne kaj dosti uporabna, pa vsekakor pomeni določen korak v iskanju odgovora, kako je najbolje razvrščati vektorje premikov v deformacijski analizi. Menim, da je obravnavani način razvrščanja zares nekakšna slepa ulica. Vseeno pa bi na podlagi znanja, ki sem ga pridobil med študijem te tematike, lahko nadaljeval z iskanjem boljše metode.

Zagotovo bi bilo treba preizkusiti, kaj se pri spektralnem grozdenju dogaja s takšnimi podatki.

Druga zamisel je podrobnejše testiranje oziroma obravnavanje samoorganizirajočih se mrež. Rezultati tam res niso tako enolični, vendar pa tudi problemi niso takšni, da bi morali znati nanje enolično odgovoriti.

Naslednja možnost je raziskava oziroma razvoj algoritma, ki bi iz mojih rezultatov različnih form sestavil končni odgovor. Postopek bi moral vsebovati vsaj izločanje osamelih vektorjev in povezovanje pomešanih skupin.

Poleg vsega naštetega bi bilo dobro razmišljati tudi v smeri upoštevanja natančnosti določitve premikov, ki jih obdelujemo. Verjetno bi bilo smiselno, da bi podatki poleg slabše natančnosti oziroma zanesljivosti imeli tudi manjšo moč pri oblikovanju skupin.

VIRI

Uporabljeni viri

Blewitt, G. 2007. GPS and Space – Based Geodetic methods. Treatise on Geophysics, Vol 3., Los Angeles, University of California: 351-390.

Bezdek, James C. 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Kluwer Academic Publishers.

Brief history of neural networks
http://library.thinkquest.org/C007395/tqweb/history.html (15.8.2009)

Carpenter, G., Grossberg, S. 2002. Adaptive resonance theory. The Handbook of Brain Theory and Neural Networks, Second Edition. Cambridge, Massachusetts: MIT Press.

Cluster Analysis http://en.wikipedia.org/wiki/Cluster_analysis (24.7.2009)

Dobnikar, A. 1990. Nevronske mreže – teorija in aplikacije. Radovljica: Didakta.

Demuth, H., Beale, M., Haganet, M. 2009. Neural Network Toolbox™ 6 User's Guide. http://www.mathworks.com/access/helpdesk/help/pdf_doc/nnet/nnet.pdf (24.7.2009)

Haykin, S. 1994. Neural Networks. A Comprehensive Foundation. Macmillan, New York, NY.

Kangas, J., Kohonen, T. 1996. Developments and applications of the self-organizing map and related algorithms. Mathematics and Computers in Simulation 41, Issues 1-2, 3-12.

Kohonen, T, 1998. The self-organising map. Neurocomputing, Volume 21, Issues 1–3, 6, Pages 1–6.

Nevron http://sl.wikipedia.org/wiki/Nevron (14.8.2009)

StatSoft, Inc. 2007. Electronic Statistics Textbook. Tulsa, OK: Statsoft. Cluster Analysis http://www.statsoft.com/textbook/stcluan.html (24.7.2009)

Turk, G. 2005. Verjetnostni račun in statistika. Ljubljana: FGG.

Drugi viri

Chang, H., Kopaska-Merkel, D., Chen, H. 2000. Identification of lithofacies using Kohonen self-organising maps. Computers & Geosciences 28, Issue 2, 223-229.

Ersoy, O., Aydar, E., Gourgaud, A., Artuner, H. and Bayhan, H. 2007. Clustering of volcanic ash arising from different fragmentation mechanisms using Kohonen self-organizing maps. Computers & Geosciences 33, Issue 6, 821-828.

Zupan, J. 2007. Umetne nevronske mreže – intelektualni izziv. Spletni video. http://video.google.com/videoplay?docid=4273978201486981984 (17.5.2009).

Priloge

Priloga A, vhodni podatki

10 vektorjev

10	5	0.15	3.1
11	13	0.0	3
15	3	-0.2	2.8
17	10	0.1	3
27	4	-2.1	1.7
29	15	-2.7	2.7
23	22	-0.8	3.9
14	24	1	0.5
19	28	1	0.5
5	30	1.1	0.5

Kvadrat

50 vektorjev

9.2300	13.5500	0.5333	-1.2991
18.8400	26.6200	0.0108	-1.1191
15.5700	26.9100	0.5373	-0.9610
13.8200	34.4500	0.7188	-1.4010
32.4900	40.1700	0.1466	-1.0971
7.7200	45.9100	0.6890	-1.4368
28.6300	49.1400	0.3658	-1.1640
33.4400	49.3900	-0.0986	-1.2564
28.1300	2.9700	2.3567	-2.6882
19.3600	4.7400	1.9579	-2.5072
33.5100	14.6000	2.6389	-2.7461
26.9000	25.5000	2.8202	-3.3837
30.9800	30.1700	2.9047	-2.9319
41.4300	53.8800	2.4272	-16.0827
10.3900	61.2000	2.0193	-16.5431
40.6000	61.2600	2.2637	-16.3330
30.1500	70.3700	2.4585	-16.0608
24.4400	77.5800	1.7961	-16.9126
38.7800	81.7000	2.0055	-16.2485
26.3600	82.9200	1.6777	-16.8386
33.7300	83.5800	2.0052	-16.3847
34.3000	84.4400	2.3527	-16.9115
37.9400	91.1800	1.7289	-16.9620
6.0100	91.5100	1.9036	-16.1168
5.1400	92.3700	2.5440	-16.6685
7.2800	94.9900	1.9083	-16.4085
50.7300	92.2100	4.1393	-7.6535
51.4600	21.3800	4.1696	-7.7908
52.9100	96.1700	4.1608	-7.0774
53.8200	67.3000	3.8215	-7.2091
57.4900	24.1500	3.9009	-7.1356
61.5500	39.5300	4.0886	-7.7078
61.7500	82.0300	4.1089	-7.2198
64.5100	27.9200	3.5021	-7.6057
64.9100	54.9300	3.6685	-7.4021
75.6900	69.8600	4.4805	-7.2886
76.3700	7.1800	3.7987	-7.7034
78.1300	31.8700	4.3962	-7.2685
78.1900	10.6600	3.7918	-7.5912
78.8300	81.8300	4.2044	-7.6172
79.8200	63.4000	3.9041	-7.0417
81.9000	51.3800	3.7062	-7.8751
83.9700	31.1900	4.3870	-6.9322
89.1400	56.7700	3.9044	-7.5949
89.8700	8.4100	3.9928	-7.6811
90.3600	0.4400	3.5955	-7.2611
91.3000	81.7500	3.9014	-7.1682
93.4800	97.5700	3.5249	-7.1563
93.9200	55.7900	4.2721	-7.4735
99.7000	14.6000	4.4777	-7.4243

100 vektorjev

36.0300	54.8500	4.4671	3.3925
23.4400	54.8800	4.3971	3.9288
43.4400	56.2500	3.9245	3.3770
4.9300	57.1100	4.4615	3.1053
24.6000	58.7400	3.9248	3.9556
39.3200	59.1500	4.0507	3.6211
26.1800	59.7300	4.6243	3.8545
31.4300	60.8400	3.8079	3.0787
44.4200	62.0600	4.2902	3.1400
17.5000	62.1000	3.7076	3.1877
27.3100	62.6200	3.7014	3.3746
43.1900	63.4300	4.2720	3.3822
23.7900	64.5800	3.8523	3.0659
11.4600	66.4900	3.7314	2.9766
18.5900	70.0600	3.6929	3.6700
41.9900	75.3700	3.8930	3.3057
49.0200	81.5900	4.3065	3.6993
13.7000	81.8800	3.7854	3.6101
16.6200	83.3200	4.1023	3.8972
42.2500	85.6000	4.3469	3.1319
45.8600	86.9900	3.7566	3.1232
16.0300	87.2900	3.7238	3.0057
43.0200	89.0300	4.6007	2.9784
19.3900	90.4800	3.9872	3.0666
50.6100	46.4800	1.7417	6.5989
53.6900	5.9500	1.5273	7.0129
54.1400	94.2300	2.0512	7.2182
55.3400	29.2000	1.9912	6.4241
56.6800	82.3000	1.6773	6.9394
56.9200	63.1800	1.6893	6.8524
59.4700	56.5700	2.0255	6.4756
59.5500	2.8700	1.1855	7.1354
59.7200	16.1400	2.0235	6.5406
60.2000	25.3600	1.8264	7.2820
61.0900	7.1200	1.6571	7.3388
61.6600	11.3300	1.6493	7.0329
62.0800	73.1300	1.4081	7.3129
62.5200	73.3400	2.1448	6.6614
62.7300	69.9100	1.2960	6.5967
65.5200	83.7600	1.2498	6.8950
65.5500	39.1900	2.0304	6.4939
67.0000	20.0900	1.9803	6.6336
67.3900	99.9400	1.5258	6.8310
68.0200	5.3400	1.2621	6.3944
69.3200	65.0100	2.0563	6.5349
70.0900	96.2300	1.5876	7.1749
70.1500	9.2200	2.1496	6.7559
70.3600	48.5000	2.0597	7.2038
71.6500	51.1300	2.0541	7.1287
73.2700	42.2200	1.4015	7.0334
73.4900	68.7300	1.2153	6.7204
75.0500	74.0000	2.0440	6.8838
77.6400	48.9300	1.4497	7.3131

79.1100	81.5000	1.6975	6.9618
79.3900	92.0000	1.3651	7.3057
80.3000	8.3900	1.6026	6.9117
81.2100	61.0100	1.8234	6.4186
82.9500	95.6100	1.6978	6.9838
83.8600	45.1600	1.4324	6.7616
84.4700	36.7800	1.5484	6.8370
85.8000	33.5800	1.6966	6.9913
86.9900	76.9400	1.9781	6.3741
87.0400	0.9900	1.3501	6.8269
87.3500	51.3400	2.0381	6.3662
89.8300	75.4600	2.0354	7.2705
90.1600	0.5600	2.1446	7.1904
90.9000	59.6200	1.8622	7.3144
93.1600	33.5200	1.8885	7.0999
93.4200	26.4400	1.3574	6.8118
94.5500	91.5900	2.0037	6.5497
95.1700	64.0000	1.6546	6.5480
95.6600	14.7200	2.0942	7.2402
96.1400	7.2100	2.0238	7.2203
96.1600	5.8900	1.7666	6.8860
96.6900	66.4900	1.4724	6.7569
98.2700	80.6600	1.9418	6.8578
98.3000	55.2700	1.5009	6.8003
15.5600	19.1100	-5.7770	1.3446
8.9000	27.1300	-6.3513	1.5734
24.7300	35.2700	-6.0915	1.8485
31.4200	36.5100	-5.9035	1.3100
34.1800	40.1800	-6.3930	1.8977
30.7700	41.1600	-6.3153	2.1828
39.7200	41.3600	-6.6037	1.6294
37.1600	42.5300	-5.9700	1.9929
40.9300	46.3500	-6.3167	2.2221
40.9100	47.4000	-5.9143	1.5095
32.9000	47.8200	-6.3507	1.9307
18.7900	49.0600	-5.8847	1.3450
35.6700	49.8300	-6.7535	1.3535
37.5900	0.9900	-0.3259	-5.9122
11.9700	3.8100	0.4253	-5.2190
29.7400	4.9200	0.2731	-5.6561
36.5400	14.0000	0.5620	-5.6373
34.6100	16.6000	0.1759	-5.3182
40.0100	19.8800	-0.0468	-5.9251
43.9800	34.0000	0.2910	-5.2340
42.4900	37.5600	0.0881	-5.8596
45.0700	41.2200	-0.1033	-5.3846
46.0800	45.7400	-0.2866	-5.0637

100 vektorjev

16.7100	988.9200	0.7080	6.0500	726.3500	880.5200	4.6900	6.0950
20.4900	954.9600	0.5740	6.0070	726.4000	800.4300	4.5120	5.8410
25.9100	379.4100	2.1020	7.5570	754.2000	971.1300	4.1750	6.4420
26.7200	500.1700	-0.1920	5.8340	760.9900	879.1500	4.4160	5.8960
38.5500	279.1600	2.5560	8.5020	793.7100	883.6000	4.1540	6.3200
79.8100	555.9400	0.2960	6.4310	797.3400	707.6400	4.8220	6.4720
90.1500	614.0100	1.2630	4.9560	800.6900	506.9200	0.3410	3.6330
92.8500	664.7100	0.8630	4.9060	825.0500	1.7700	-0.2990	8.9160
93.4000	982.4100	0.9230	6.0080	843.2500	564.3200	0.7420	3.6710
118.5300	490.2500	-0.4150	5.8180	857.4000	255.8800	4.6120	7.4290
134.8300	919.3700	0.4170	6.0750	864.2500	603.8400	0.8420	3.8810
137.3500	963.4500	1.0720	5.7390	867.2200	159.2800	4.9590	8.0540
137.9700	109.3000	2.7880	7.8110	867.7400	453.5000	4.4520	7.9470
154.8800	479.4100	0.3750	6.1360	884.6700	518.4700	4.7070	7.8670
156.9500	812.0400	0.9100	6.1270	901.6000	388.6400	4.7410	7.9740
206.9100	244.1200	2.3940	8.4660	907.3600	486.6100	4.4000	8.1090
209.4000	381.8100	2.7150	7.5580	913.5300	538.8400	0.5030	3.3390
219.5800	710.8000	1.6020	5.4980	914.9600	154.2000	4.3880	7.9120
237.4500	553.6400	0.4850	5.8860	921.4600	920.9600	0.6290	3.5760
240.6300	567.0100	0.4850	6.5680	936.3900	184.5600	5.0300	8.1240
240.9200	489.5400	-0.1290	6.2390	945.1000	292.2900	4.9610	7.9140
244.3000	343.8400	2.8460	7.8170	945.3700	761.8300	0.4780	3.7080
265.2700	63.2100	2.0660	5.3400	961.2900	654.3400	0.7980	4.0200
272.4100	499.6300	0.3910	6.3250	989.7100	683.1300	0.6570	3.8260
285.4400	145.4100	2.4920	4.9590	993.2200	354.4600	0.3230	3.4090
316.1900	223.5700	4.5830	3.7370				
322.5000	355.7300	4.8790	4.0110				
325.4300	268.0400	4.7750	3.3020				
327.4300	125.0800	2.1750	4.8260				
332.6700	865.1600	4.7480	7.6520				
348.3200	495.0300	0.1250	6.0900				
350.0800	284.9600	4.4410	3.2270				
366.1100	616.9300	1.2200	5.1000				
370.7400	744.4700	0.9090	4.9240				
372.5300	16.9700	2.6170	4.9600				
378.8800	837.0300	4.1090	8.1370				
378.9700	751.0300	1.3850	5.1720				
379.9600	935.5900	3.7770	3.3360				
387.1300	898.1700	4.0550	3.0350				
402.9400	420.7600	0.4800	5.8440				
410.7900	60.4900	1.9920	4.8290				
428.1900	809.9300	4.1590	8.4230				
440.5000	437.5400	-0.0250	6.1940				
453.9000	444.0300	0.1230	6.1830				
458.4300	258.2300	0.2280	6.0010				
490.7500	773.1900	0.6700	5.4070				
497.6500	189.0500	4.8560	3.3430				
513.8400	709.2500	1.2020	5.7270				
520.4700	914.2000	4.7820	7.6010				
525.5600	422.6800	0.3060	8.8950				
533.2200	973.1100	4.5690	8.1890				
539.8700	442.6600	-0.0450	8.7980				
541.0400	294.3800	-0.0440	5.9210				
545.9200	521.3200	4.4240	6.2480				
545.9500	975.8200	4.3440	8.0920				
556.0400	840.3200	1.2530	5.4310				
556.4700	512.4500	3.8930	5.8570				
558.9900	830.1000	1.1870	5.6620				
559.3500	913.1900	4.8450	8.4970				
598.4700	664.8400	1.5710	5.5130				
600.7100	457.6800	0.0770	9.1710				
601.4500	810.1700	0.9710	5.5740				
621.2400	829.8900	1.4640	5.6480				
624.4100	401.5800	0.4650	9.2650				
633.1600	173.6500	4.8160	3.4940				
633.7300	619.9900	4.6080	5.5770				
638.1100	550.1800	4.0430	6.2540				
640.7200	340.3700	-0.0920	8.5200				
642.2000	729.3200	0.6650	5.6870				
642.2400	333.3500	0.5690	8.8790				
650.0100	837.1100	1.5610	5.0080				
689.5400	663.0100	4.3360	6.3260				
710.6500	750.5500	4.0450	6.0590				
719.2500	900.4200	3.9820	6.3620				
719.5700	632.9500	4.0980	5.8340				

804 vektorji – California

34.726717	288.710006	0.044	-0.801	34.061526	286.337039	-0.097	-0.640	33.655830	284.954748	0.079	-0.115
35.323571	286.887837	0.005	-0.838	34.127931	285.985141	-0.071	-0.723	33.529642	285.162976	-0.003	-0.068
35.046835	287.872269	0.022	-0.831	34.863484	283.911407	0.029	-0.777	33.575434	285.002616	0.047	-0.071
35.209830	285.601833	-0.002	-0.800	34.914162	283.780176	0.004	-0.776	33.532127	284.956225	0.048	-0.063
35.209249	285.604212	0.014	-0.797	34.896192	283.750142	0.026	-0.757	33.532127	284.956225	0.044	-0.086
33.648182	289.564202	-0.003	-0.046	34.895554	283.744540	0.019	-0.793	33.635551	284.516968	0.034	-0.287
34.252977	286.915380	-0.025	-0.707	34.631157	284.220717	0.019	-0.758	33.598158	284.522881	0.038	-0.245
35.635038	282.481552	0.015	-0.852	34.666086	284.004592	0.007	-0.778	33.392957	285.000091	0.026	-0.034
35.645916	282.436309	-0.021	-0.863	34.735397	283.379571	0.021	-0.793	34.158610	283.520201	-0.013	-0.674
36.046065	281.135136	-0.007	-0.864	34.334997	285.051434	-0.007	-0.770	34.101237	283.401235	-0.014	-0.635
36.063919	281.071898	0.011	-0.868	34.531443	284.429881	0.009	-0.748	34.106871	283.375615	-0.024	-0.650
35.256128	281.925711	-0.011	-0.837	34.400915	284.600974	0.002	-0.752	34.042070	283.532590	-0.024	-0.602
35.612629	281.237579	0.004	-0.873	34.236021	285.371920	-0.023	-0.769	34.041362	283.478257	-0.035	-0.583
35.279479	281.800713	0.005	-0.831	34.176768	285.366992	0.067	-0.701	34.015127	283.512501	-0.012	-0.547
35.970607	279.628943	-0.001	-0.875	34.097478	285.562583	-0.073	-0.750	34.015127	283.512501	-0.018	-0.560
35.719400	279.895804	0.004	-0.845	34.209959	285.169245	0.019	-0.763	33.942540	283.699235	0.019	-0.490
35.717054	279.822085	0.004	-0.874	33.951163	285.765479	-0.059	-0.648	33.942540	283.699235	0.006	-0.502
35.033607	281.486510	0.079	-0.748	33.897332	285.675319	0.018	-0.568	33.770766	284.174244	-0.051	-0.397
35.034368	281.486500	0.012	-0.803	34.044699	285.304422	-0.013	-0.707	33.796279	283.933101	-0.026	-0.409
34.988875	281.546392	-0.054	-0.795	34.043967	285.303445	-0.014	-0.744	34.085591	283.396680	-0.032	-0.634
34.985224	281.631881	0.021	-0.790	34.156275	285.036088	0.002	-0.732	34.067976	283.422191	-0.085	-0.624
34.967315	281.642411	0.030	-0.788	33.913550	285.497460	-0.058	-0.533	34.067976	283.422191	-0.026	-0.623
34.990548	281.469251	0.023	-0.765	33.838361	285.555886	-0.067	-0.463	34.050128	283.448816	-0.031	-0.621
34.961827	281.525659	0.046	-0.785	34.440535	284.359750	0.040	-0.786	34.047341	283.450613	-0.028	-0.600
34.966850	281.465171	-0.025	-0.827	34.363456	284.538296	0.026	-0.752	34.025632	283.468817	-0.024	-0.590
34.952445	281.484653	0.027	-0.785	34.362065	284.539347	0.007	-0.756	34.014119	283.480870	-0.031	-0.588
34.899320	281.505684	0.014	-0.747	34.263060	284.645198	0.010	-0.736	34.019095	283.472273	-0.032	-0.568
34.940539	281.319066	0.008	-0.834	34.349768	284.123559	0.028	-0.751	34.007635	283.505269	0.004	-0.530
34.893184	281.368636	0.025	-0.759	34.291802	284.272220	0.007	-0.731	33.999514	283.516515	-0.051	-0.543
34.864332	281.470771	0.058	-0.755	34.256606	284.357715	0.058	-0.702	34.012837	283.487709	-0.004	-0.580
34.834396	281.500609	0.011	-0.750	34.319439	284.106207	-0.011	-0.726	34.011262	283.493302	-0.003	-0.575
34.931409	281.215413	0.023	-0.802	34.107224	284.584956	0.002	-0.700	34.031460	283.376922	0.020	-0.603
34.908493	281.180745	0.046	-0.747	34.061416	284.695589	0.004	-0.642	34.037519	283.320055	-0.016	-0.564
34.852185	281.335566	0.071	-0.736	34.131961	284.406873	0.017	-0.673	34.037519	283.320055	-0.030	-0.578
34.772773	281.514321	0.051	-0.730	34.008087	284.538080	0.009	-0.629	33.999074	283.400044	-0.020	-0.540
34.767552	281.471465	0.070	-0.660	33.914352	285.128758	-0.020	-0.660	33.999074	283.400044	-0.034	-0.558
34.805682	281.379962	0.057	-0.707	34.029238	284.864826	0.001	-0.696	33.971019	283.542407	-0.047	-0.566
34.768222	281.457027	0.046	-0.757	33.951102	285.057974	0.015	-0.679	33.966991	283.546437	-0.027	-0.557
34.757327	281.424868	0.032	-0.736	33.718079	285.535515	-0.022	-0.090	33.967871	283.545511	-0.034	-0.549
35.183552	280.552688	0.033	-0.804	33.878296	284.918633	0.019	-0.584	33.955969	283.563967	-0.023	-0.497
35.260340	280.285510	0.004	-0.830	33.817737	284.997474	-0.042	-0.438	33.948766	283.568983	-0.032	-0.422
35.032262	280.538096	0.046	-0.791	33.766834	285.178111	-0.025	-0.281	33.940754	283.565430	-0.015	-0.490
35.042529	280.435724	0.042	-0.813	33.694700	285.278820	0.088	-0.130	33.921782	283.610982	-0.005	-0.475
35.118636	280.766292	0.030	-0.793	34.223259	286.714696	-0.026	-0.695	33.904674	283.614257	-0.032	-0.478
35.032119	280.822916	0.009	-0.806	33.956709	287.038611	-0.033	-0.229	33.921775	283.542203	-0.036	-0.504
35.050724	280.674591	0.044	-0.802	34.053519	286.402803	-0.122	-0.651	33.906192	283.551887	0.047	-0.483
35.050724	280.674816	-0.164	-0.748	33.870190	286.918249	0.010	-0.211	33.906192	283.551887	-0.008	-0.468
34.980089	280.840712	0.039	-0.778	33.996851	286.292206	-0.098	-0.644	33.906192	283.551887	-0.028	-0.476
34.925778	280.979452	-0.007	-0.771	33.757789	286.349711	0.064	-0.092	33.903523	283.549190	-0.002	-0.506
34.896664	280.964853	0.031	-0.783	33.814681	286.080101	-0.000	-0.202	33.903829	283.548464	-0.023	-0.497
34.945786	280.883492	0.012	-0.792	33.646919	286.164991	0.013	-0.072	33.903700	283.548660	-0.029	-0.464
34.866680	280.938619	0.040	-0.793	33.630446	286.200137	0.017	-0.043	33.903829	283.548464	-0.021	-0.489
34.893703	281.170711	0.012	-0.819	33.401637	286.488590	-0.002	-0.050	33.903523	283.549190	-0.034	-0.480
34.844433	281.227584	0.055	-0.746	33.671836	285.802004	0.061	-0.035	33.900694	283.533800	-0.018	-0.465
34.868239	281.152972	0.022	-0.801	33.387248	286.233962	0.066	-0.094	33.900216	283.586815	-0.036	-0.517
34.815929	281.168437	0.015	-0.725	32.962435	286.755582	0.024	-0.012	33.882241	283.591462	-0.045	-0.517
34.780234	281.234479	0.045	-0.732	32.962435	286.755582	-0.028	-0.045	33.963770	283.372346	-0.024	-0.523
34.776079	281.228475	0.043	-0.749	34.751618	282.776196	0.006	-0.769	33.943219	283.398241	-0.022	-0.496
34.882848	281.019880	-0.016	-0.707	34.526828	283.230237	-0.001	-0.752	33.894227	283.526812	-0.017	-0.469
34.844642	281.021204	0.030	-0.765	34.567466	283.110677	0.041	-0.754	33.867395	283.581511	-0.001	-0.517
34.844893	281.021516	0.030	-0.765	34.390045	283.536709	0.002	-0.739	33.865005	283.439044	-0.043	-0.473
34.845198	281.020132	0.046	-0.734	34.323370	283.491168	0.012	-0.710	33.812128	283.513040	-0.044	-0.418
34.724109	281.293481	0.120	-0.689	34.749377	282.618461	0.012	-0.801	33.771418	283.696199	-0.029	-0.388
34.678836	281.284261	0.079	-0.680	34.782778	282.519687	0.006	-0.791	33.707594	283.823896	-0.003	-0.342
34.684934	281.225239	0.041	-0.693	34.761395	282.540857	0.014	-0.771	33.622490	283.945577	-0.012	-0.275
34.995759	280.617521	-0.002	-0.817	34.694481	282.695013	0.092	-0.831	33.637277	284.417964	0.035	-0.277
34.893930	280.870669	-0.011	-0.807	34.766408	282.432074	-0.003	-0.783	33.625105	284.433853	0.003	-0.228
34.927323	280.734135	-0.009	-0.823	34.762646	282.433223	0.001	-0.790	33.625105	284.433853	0.028	-0.246
34.918232	280.750916	0.044	-0.764	34.727652	282.394240	0.012	-0.790	33.599170	284.485478	0.008	-0.222
34.879755	280.827895	-0.026	-0.803	34.600244	282.664481	0.020	-0.773	33.598652	284.453773	0.032	-0.253
34.851031	280.873843	0.013	-0.775	34.758567	282.234479	0.013	-0.792	33.577527	284.520710	-0.033	-0.219
34.855370	280.745166	0.035	-0.742	34.753368	282.233833	0.029	-0.796	33.542122	284.578635	0.049	-0.129
34.849256	280.760281	0.003	-0.773	34.740137	282.226424	-0.002	-0.745	33.637808	284.237795	0.070	-0.265
34.822132	280.748332	0.004	-0.765	34.793750	282.040601	-0.027	-0.766	33.516681	284.575807	0.025	-0.110
34.809983	281.013038	0.089	-0.735	34.633308	282.430733	0.060	-0.758	33.624143	283.996626	0.025	-0.218
34.727065	281.046364	0.075	-0.712	34.452623	283.114252	0.031	-0.738	33.510650	284.284024	0.003	-0.145
34.728018	281.039380	0.047	-0.740	34.430450	282.971766	0.002	-0.727	33.514337	284.105857	0.019	-0.109
34.724139	281.020025	0.086	-0.715	34.294736	283.273639	-0.003	-0.697	33.553054	283.953240	0.037	-0.147
34.685120	281.047922	0.070	-0.725	34.224365	283.346003	-0.010	-0.696	33.410352	284.216479	0.091	-0.084
34.658862	281.162427	0.087	-0.690	34.244191	283.219391	0.007	-0.702	33.410352	284.216479	0.054	-0.115
34.603443	281.227712	0.068	-0.678	34.120943	283.363590	-0.023	-0.650	33.410352	284.216479	0.031	-0.067
34.744392	280.848163	0.076	-0.752	34.462546	282.692389	0.030	-0.729	33.245997	284.734429	0.011	-0.047
34.702544	280.880585	0.061	-0.709	34.457983	282.658052	0.002	-0.727	34.833625	281.867975	-0.007	-0.792
34.666200	281.021535	0.087	-0.689	34.279908	282.898873	-0.015	-0.690	34.839962	281.795170	-0.011	-0.758
34.629618	281.125889	0.063	-0.621	34.285037	282.962588	0.023	-0.694	34.720200	281.952959	0.006	-0.763
34.610960	281.044891	0.054	-0.695	34.149973	283.279351	-0.008	-0.684	34.787776	281.790427	0.021	-0.762
34.648532	280.978242	0.044	-0.697	34.145683	282.985574	0.021	-0.664	34.712105	281.861732	0.037	-0.754
34.646261	280.980954	0.045	-0.615	34.134621	283.139845	-0.023	-0.664	34.593405	281.971344	0.028	-0.758
34.587877	281.092600	0.154	-0.643	34.178677	284.151641	-0.003	-0.687	34.601958	282.179489	0.017	-0.734
34.586686	281.095319	0.138	-0.724	34.213093	284.015902	0.006	-0.697	34.512286	282.365138	0.008	-0.736
34.588009	281.093090	0.137	-0.734	34.302251	283.818666	0.023	-0.731	34.512286	282.365138	0.025	-0.704
34.929834	284.397192	-0.003	-0.773	34.074493	284.361611	0.012	-0.649	34.510532	282.363608	0.001	-0.735
34.819763	284.497618	-0.011	-0.745	34.000894	284.247099	-0.015	-0.598	34.607593	282.086230	0.045	-0.749
34.886485	285.238681	0.004	-0.769	34.209255	283.613416	0.003	-0.711	34.528081	282.112309	0.025	-0.723
34.362694	286.201675	-0.015	-0.753	34.015475	284.129753	0.050	-0.601	34.617717	281.843137	0.053	-0.697
34.420084	285.852430	-0.004	-0.764	33.900241	284.286144	-0.000	-0.461	34.701728	281.589470	0.050	-0.702
34.204929	286.422703	-0.019	-0.725	33.820044	284.236045	-0.008	-0.420	34.615361	281.696248	0.040	-0.720
34.590055	284.854773	0.008	-0.764	33.840372	284.700505	-0.003	-0.424	34.573701	281.694537	0.007	-0.685
34.349992	285.623954	-0.016	-0.759	33.857760	284.667814	-0.012	-0.475	34.493548	282.062627	0.021	-0.718
34.116328	286.191584	-0.059	-0.702	33.822596	284.723926	0.031	-0.407	34.493831	282.064533	0.013	-0.711

34.460566	282.120157	0.023	-0.702
34.353448	282.263229	0.013	-0.674
34.404644	282.119696	0.018	-0.691
34.365647	282.210560	0.016	-0.682
34.504733	281.896729	0.029	-0.702
34.504733	281.896729	0.036	-0.698
34.379662	282.039698	-0.019	-0.666
34.314096	282.660411	0.022	-0.696
34.314514	282.649851	0.027	-0.693
34.244755	282.745143	0.014	-0.677
34.320632	282.451220	0.025	-0.678
34.264783	282.542647	-0.000	-0.663
34.146602	282.663778	0.017	-0.643
33.981994	282.989223	-0.044	-0.489
33.981994	282.989223	-0.091	-0.527
34.286274	282.412575	0.002	-0.674
34.207882	282.538276	0.004	-0.629
34.188183	282.477718	0.022	-0.659
34.238546	282.295609	-0.005	-0.656
34.086335	282.442557	0.023	-0.579
33.979232	282.874755	-0.009	-0.514
34.044327	282.621404	0.019	-0.548
34.619869	281.539552	0.043	-0.675
34.616659	281.480960	0.046	-0.705
34.522513	281.685375	0.032	-0.694
34.567941	281.536802	0.057	-0.773
34.623271	281.309343	0.043	-0.653
34.561663	281.428171	0.097	-0.713
34.511870	281.585493	0.007	-0.634
34.511744	281.585678	0.084	-0.673
34.516409	281.533398	0.094	-0.739
34.497389	281.515483	0.106	-0.732
34.345918	281.996569	0.020	-0.680
34.295330	281.973387	0.017	-0.640
34.241982	282.022039	0.016	-0.653
34.241982	282.022039	0.021	-0.648
34.267282	281.878652	0.013	-0.666
34.571793	281.328459	0.068	-0.678
34.541756	281.378580	0.081	-0.697
34.465436	281.461522	0.084	-0.675
34.564805	281.228724	0.037	-0.666
34.500645	281.178157	0.086	-0.650
34.500434	281.175692	0.058	-0.666
34.358126	281.662417	0.047	-0.627
34.337952	281.666541	0.062	-0.627
34.371291	281.587658	0.063	-0.645
34.351051	281.609848	0.083	-0.615
34.208173	281.829864	0.065	-0.593
34.252437	281.678629	0.036	-0.620
34.321371	281.457107	0.104	-0.595
34.163524	281.721690	0.042	-0.559
34.110372	282.387272	0.022	-0.623
34.064050	282.340669	0.005	-0.537
34.064050	282.340669	0.005	-0.533
34.064050	282.340669	-0.025	-0.522
34.036579	282.363407	-0.008	-0.521
34.208927	282.059030	0.013	-0.622
34.148817	282.032034	0.038	-0.601
34.110823	282.151651	0.015	-0.615
34.022739	282.344883	0.018	-0.528
33.999245	282.328548	0.077	-0.553
33.954098	282.537871	0.051	-0.531
33.954098	282.537871	-0.052	-0.488
34.101809	281.834079	0.113	-0.572
33.994365	283.160876	-0.029	-0.554
33.842053	283.254614	-0.046	-0.448
33.752306	283.569408	-0.018	-0.388
33.737195	283.568729	-0.026	-0.388
33.701563	283.538700	-0.025	-0.341
33.632104	283.673927	-0.015	-0.133
33.632104	283.673927	0.007	-0.251
33.634367	283.649320	-0.013	-0.255
33.688894	283.399433	-0.207	-0.403
33.688894	283.399433	-0.060	-0.349
33.596195	283.562732	-0.003	-0.205
33.772579	283.212249	-0.082	-0.402
33.715999	283.227488	0.004	-0.352
33.715999	283.227488	-0.091	-0.390
33.811606	282.946167	-0.054	-0.459
33.660291	283.169335	-0.065	-0.368
33.649002	283.179041	-0.083	-0.392
33.604131	283.359313	-0.060	-0.200
33.508825	283.578940	-0.029	-0.204
33.526006	283.372023	-0.042	-0.216
33.500968	283.371599	-0.062	-0.190
33.492305	283.951715	-0.022	-0.164
33.177875	284.469065	0.008	-0.030
33.324597	283.940311	-0.017	-0.053
33.579404	283.000296	-0.092	-0.267
33.387602	283.310166	-0.038	-0.159
33.426282	283.176353	-0.066	-0.184
33.504167	282.883686	-0.089	-0.226
33.504167	282.883686	-0.133	-0.097
33.678094	282.516477	-0.061	-0.392
33.305247	283.101183	-0.064	-0.136
33.377975	283.515269	0.008	-0.114
33.292893	283.446553	-0.038	-0.122
33.229268	283.359592	-0.020	-0.022
33.198875	283.209590	-0.048	-0.100
33.414420	285.261449	0.031	-0.016
32.950595	285.497632	-0.054	-0.008
32.993599	285.032556	0.011	-0.027
33.103275	284.675573	0.023	-0.058
32.823434	285.312192	0.015	-0.044
32.632786	285.359654	-0.007	-0.036
32.854400	284.197869	-0.010	-0.010
32.725833	284.118643	0.020	0.001
32.389463	284.953345	-0.010	-0.017
36.001416	278.431426	0.062	-0.867
35.351836	279.471357	0.093	-0.740
35.372524	279.274856	0.075	-0.808
35.372725	279.262173	0.049	-0.807
35.904371	278.376791	0.034	-0.868
35.904115	278.376692	0.021	-0.893
35.904371	278.376791	0.092	-0.907
35.904371	278.376791	0.077	-0.875
35.904237	278.376513	0.039	-0.870
35.901698	278.347979	-0.010	-0.873
35.881385	278.328183	0.024	-0.872
35.664898	278.328304	-0.015	-0.908
35.573609	278.438751	0.061	-0.860
35.674562	278.824309	0.030	-0.830
35.399328	279.203094	0.095	-0.910
35.169005	279.125381	0.063	-0.802
35.264185	279.875389	0.087	-0.789
35.255323	279.883153	0.076	-0.761
35.132252	280.152396	0.099	-0.809
34.977593	280.411813	0.057	-0.773
35.013083	280.291640	0.065	-0.828
34.972892	280.310097	0.059	-0.775
35.120915	280.068426	0.039	-0.761
35.118707	280.022509	0.051	-0.775
35.055018	280.120009	0.060	-0.809
34.951330	280.267663	0.078	-0.814
34.914926	280.252805	0.071	-0.816
35.071900	279.702521	0.141	-0.721
34.989405	280.074699	0.137	-0.668
34.989405	280.074699	0.076	-0.740
34.948397	280.121614	0.137	-0.798
34.962393	279.994645	0.102	-0.735
34.962452	279.994779	0.103	-0.737
34.897392	280.108073	0.108	-0.719
34.847463	280.078486	0.066	-0.664
34.884580	280.578062	0.022	-0.774
34.852770	280.526405	0.065	-0.792
34.848491	280.583049	0.032	-0.773
34.810151	280.695566	0.018	-0.756
34.776965	280.697424	0.064	-0.713
34.766318	280.703881	0.033	-0.689
34.729810	280.679810	0.050	-0.657
34.726507	280.681267	0.092	-0.665
34.697721	280.792496	0.038	-0.696
34.587769	280.944782	0.087	-0.651
34.523417	281.003001	0.020	-0.598
34.653697	280.653683	0.084	-0.642
34.653757	280.653798	0.079	-0.645
34.641648	280.669565	0.079	-0.678
34.784921	280.335321	0.084	-0.749
34.687869	280.476549	0.097	-0.645
34.711385	280.236660	0.091	-0.614
34.568814	280.701679	0.068	-0.634
34.561699	280.712764	0.085	-0.647
34.451500	280.893546	0.079	-0.638
34.477293	280.837487	0.062	-0.608
34.477232	280.837573	0.067	-0.636
34.451506	280.707247	0.037	-0.589
34.451506	280.707247	0.036	-0.594
35.187464	279.441199	0.124	-0.779
35.042708	279.592481	0.121	-0.723
35.116362	279.349476	0.120	-0.754
35.020582	279.465106	0.106	-0.688
35.011851	279.489147	0.123	-0.688
34.930258	279.663454	0.144	-0.726
35.024706	279.352105	0.136	-0.706
35.055398	279.146721	0.077	-0.711
35.055398	279.146721	0.147	-0.683
35.055469	279.146616	0.075	-0.723
34.953315	279.287516	0.094	-0.637
34.890581	279.642133	0.092	-0.638
34.833649	279.514717	0.106	-0.604
34.712237	280.071845	0.117	-0.623
34.712368	280.071888	0.118	-0.608
34.712368	280.072109	0.118	-0.608
34.662339	280.266951	0.136	-0.629
34.669634	280.187698	0.095	-0.581
34.710833	280.071842	0.110	-0.624
34.591438	280.185834	0.081	-0.601
34.472701	280.441625	0.049	-0.584
34.446465	280.614093	0.061	-0.636
34.392778	280.591453	0.020	-0.561
34.354204	280.512582	-0.016	-0.472
34.359015	280.454284	0.041	-0.518
34.665119	279.865944	0.085	-0.583
34.517289	280.071120	0.039	-0.544
34.517162	280.071067	0.045	-0.536
34.594975	279.824080	0.093	-0.537
34.271073	280.508161	-0.005	-0.471
34.384186	280.138713	-0.011	-0.471
34.384186	280.138713	-0.008	-0.461
34.216385	280.376067	-0.006	-0.418
35.683146	277.940517	0.027	-0.873
35.522000	278.379541	0.035	-0.871
35.446771	278.126174	0.018	-0.841
35.162334	278.837632	0.059	-0.772
35.162334	278.837632	0.057	-0.772
35.126465	278.483554	0.056	-0.745
35.056516	278.508547	0.069	-0.686
35.056391	278.508496	0.062	-0.683
35.548723	277.706669	-0.038	-0.910
35.344593	278.018378	0.034	-0.847
35.265723	277.885370	0.026	-0.821
35.226056	278.179690	0.048	-0.782
35.226056	278.179690	0.036	-0.776
35.151214	278.173818	0.030	-0.763
35.084843	278.250117	0.042	-0.733
34.877963	278.590894	0.003	-0.516
35.012625	278.307186	0.028	-0.708
35.168217	278.016181	0.040	-0.774
35.038056	278.029645	0.034	-0.713
34.976926	278.141173	0.045	-0.673
34.903785	278.262599	0.056	-0.559
34.898150	278.161220	0.037	-0.547
34.998492	279.006226	0.066	-0.668
34.809403	279.369700	0.073	-0.602
34.809403	279.369700	0.064	-0.595
34.722143	279.582852	0.100	-0.588
34.658214	279.618606	0.064	-0.545
34.626265	279.608132	0.070	-0.525
34.686933	279.480123	0.078	-0.558
34.599631	279.469116	0.055	-0.503
34.953429	278.900786	0.065	-0.624
34.905426	278.987052	0.063	-0.589
34.897338	278.909258	0.043	-0.561
34.897338	278.909258	0.056	-0.562
34.709682	279.082631	0.013	-0.441
34.600551	279.222088	0.046	-0.469
34.600551	279.222088	0.124	-0.411
34.606913	279.685855	0.042	-0.537
34.600677	279.701829	0.058	-0.529
34.600749	279.701968	0.060	-0.518
34.431857	279.908749	-0.014	-0.445
34.431984	279.908790	-0.011	-0.439
34.248329	280.275981	-0.048	-0.400
34.318839	279.973164	-0.029	-0.346
34.168231	280.167128	-0.008	-0.397
34.436562	279.578125	0.021	-0.421
34.396792	279.693283	-0.005	-0.400
34.252759	279.677097	0.029	-0.391
34.097034	280.023704	-0.045	-0.366
34.097094	280.023601	-0.026	-0.367
34.101821	279.994618	0.008	-0.374
34.414078	279.007360	0.056	-0.318
34.327077	279.440311	0.023	-0.380
34.232628	279.585663	-0.008	-0.367
34.407454	279.243549	0.075	-0.392
34.302093	279.363981	0.045	-0.366
34.010432	279.909201	-0.019	-0.342
34.140031	279.610865	-0.014	-0.322
34.075791	279.735166	-0.003	-0.345
34.187436	279.218240	0.029	-0.337
35.576298	276.821552	0.043	-0.872
35.630671	276.314714	-0.002	-0.896
35.555431	275.700194	0.024	-0.876
35.533448	277.120882	0.004	-0.868
35.533377	277.120756	-0.000	-0.866
35.410369	277.365624	0.065	-0.842
35.431228	277.202480	0.022	-0.858
35.431169	277.202361	-0.006	-0.832
35.202704	277.608768	0.019	-0.753
35.308353	277.217777	0.034	-0.799
35.308353	277.217777	0.010	-0.817
35.207313	277.498526	0.047	-0.767
35.410687	277.039381	0.001	-0.861
35.171970	277.193632	0.019	-0.759
35.171970	277.193632	0.000	-0.772
35.050308	277.292731	0.012	-0.722
35.012922	277.862677	0.044	-0.705
34.910875	277.991685	0.025	-0.563
34.822347	277.991922	-0.009	-0.427
34.822347	277.991922	0.006	-0.407
34.733712	278.055020	-0.004	-0.335
34.975311	277.537470	0.009	-0.622
34.935768	277.666242	0.029	-0.589
34.935709	277.666101	0.025	-0.605
34.870306	277.563298	0.017	-0.478
35.069083	277.212740	0.010	-0.745
35.016434	277.291233	0.024	-0.669
35.016434	277.291233	0.027	-0.681
35.050278	277.090365	0.008	-0.728
35.050278	277.090365	-0.001	-0.730
34.960152	277.139120	0.006	-0.647
35.050522	276.891708	-0.015	-0.749
35.050522	276.891708	-0.008	-0.754
35.003529	276.899435	0.020	-0.748
34.986546	277.019452	0.002	-0.681
34.986546	277.019452	-0.006	-0.687
34.927168	277.079387	0.001	-0.522
34.925092	277.080705	0.010	-0.535
34.990858	276.819983	-0.009	-0.730
34.990858	276.819983	-0.021	-0.727
34.986004	276.820900	0.000	-0.719
34.990679	276.749567	0.011	-0.748
34.933013	276.904011	-0.015	-0.514
34.933013	276.904011	-0.038	-0.499
34.904220	276.977186	-0.015	-0.494
34.895995	276.892524	-0.027	-0.413
34.913518	277.253848	0.031	-0.521
34.913524	277.253618	0.009	-0.521
34.758783	277.654529	0.007	-0.377

34.878429	277.075113	-0.020	-0.485	33.732263	281.811834	0.036	-0.415	33.482940	279.795868	0.005	-0.290
34.878429	277.075113	-0.014	-0.462	33.625284	282.011598	0.040	-0.421	33.455910	279.735920	-0.000	-0.281
34.595745	278.347354	0.047	-0.354	34.363142	280.742824	0.071	-0.575	33.325884	279.795932	0.026	-0.279
34.246970	277.888379	0.014	-0.350	34.328427	280.697120	0.026	-0.548	33.270757	280.196756	0.003	-0.248
33.862345	278.844235	0.162	-0.371	34.238959	280.932474	-0.019	-0.484	33.054676	280.762874	-0.007	-0.158
33.847535	278.817697	0.058	-0.294	34.231207	280.916198	-0.002	-0.484	32.996195	280.628750	0.000	-0.161
33.867567	278.598462	0.059	-0.298	34.231207	280.916198	-0.021	-0.483	32.994941	280.632551	0.033	-0.223
35.087267	276.429803	-0.020	-0.805	34.231207	280.916198	0.035	-0.517	32.996195	280.628750	0.031	-0.170
34.957887	276.748920	0.008	-0.728	34.130348	280.895696	0.033	-0.466	32.996195	280.628750	0.032	-0.176
34.969127	276.727467	-0.015	-0.719	34.139471	280.776849	-0.026	-0.404	33.098235	280.245358	0.007	-0.204
34.969127	276.727467	-0.018	-0.726	33.843022	281.215346	0.020	-0.394	33.437836	279.448566	0.006	-0.289
34.968507	276.701432	-0.011	-0.724	34.088796	280.831038	0.019	-0.440	33.566380	279.213013	0.032	-0.312
34.948582	276.754840	-0.008	-0.690	33.994230	280.694407	0.086	-0.434	33.384480	279.498170	0.005	-0.297
34.948528	276.754245	-0.028	-0.694	33.946173	280.806047	-0.037	-0.358	33.413496	279.022997	0.020	-0.291
34.986117	276.631936	-0.022	-0.717	33.946173	280.806047	-0.007	-0.397	33.209127	279.567869	0.039	-0.261
34.993579	276.611043	-0.036	-0.774	33.787615	281.531021	0.025	-0.397	33.286845	279.259243	0.071	-0.328
34.993579	276.611043	-0.021	-0.774	33.833884	281.293910	0.020	-0.439	33.071459	280.067205	0.017	-0.209
34.959318	276.669403	0.027	-0.707	33.720048	281.414916	0.013	-0.367	33.102044	279.659076	0.018	-0.215
34.938327	276.721598	-0.022	-0.469	33.549124	281.946156	0.020	-0.425	33.028076	279.664312	0.023	-0.183
34.955324	276.664454	0.072	-0.719	33.685101	281.185301	0.063	-0.384	32.638049	281.240668	0.050	-0.059
34.945607	276.683811	-0.074	-0.455	33.577774	282.517231	-0.017	-0.261	32.402084	281.666272	0.038	-0.070
34.924233	276.828413	0.023	-0.567	33.577774	282.517231	-0.209	-0.388	32.001160	281.876200	0.008	-0.042
34.848987	276.915542	-0.035	-0.444	33.480873	282.654051	-0.050	-0.259	32.346110	280.554826	0.025	-0.079
34.864804	276.853842	-0.056	-0.455	33.456214	282.685049	-0.039	-0.196	33.055503	279.359763	0.050	-0.208
34.864739	276.853688	-0.027	-0.425	33.456858	282.500476	-0.076	-0.335	33.273091	278.762923	0.006	-0.277
34.879677	276.792783	-0.037	-0.439	33.368110	282.853140	-0.054	-0.097	33.224164	278.792097	0.009	-0.213
34.879677	276.792783	-0.020	-0.421	33.204251	282.967915	-0.095	-0.120	33.213074	278.810750	-0.008	-0.224
34.904949	276.707725	-0.060	-0.466	33.523434	282.335159	-0.067	-0.386	32.957734	279.202599	0.034	-0.197
34.953619	276.611281	0.032	-0.662	33.541609	282.257004	-0.035	-0.383	32.940856	279.556246	0.023	-0.183
34.993478	276.508705	-0.030	-0.811	33.479926	282.201794	-0.071	-0.334	32.925365	279.426576	0.029	-0.164
34.995372	276.507896	-0.020	-0.783	33.451276	282.132372	-0.002	-0.366	32.794990	279.484363	0.040	-0.155
34.976086	276.564916	-0.017	-0.788	33.406537	282.141645	0.015	-0.346	33.150781	278.569707	0.049	-0.231
34.936377	276.649854	-0.039	-0.439	33.374102	282.154470	-0.016	-0.307	33.037179	278.858139	0.084	-0.193
34.918816	276.681746	-0.045	-0.437	33.288158	282.245983	0.021	-0.117	32.928255	278.825955	0.055	-0.202
34.901033	276.639491	-0.085	-0.423	33.333839	282.140446	0.046	-0.218	32.878074	278.787705	0.024	-0.177
34.901033	276.639491	-0.062	-0.418	33.302510	282.154419	0.023	-0.174	32.918566	278.663854	0.041	-0.164
34.908935	276.558865	-0.127	-0.459	33.261592	282.148578	0.031	-0.154	32.761059	278.815628	0.018	-0.184
35.021427	276.340927	0.008	-0.801	33.367322	281.845245	0.047	-0.271	32.676503	278.916587	-0.008	-0.158
35.021362	276.341003	-0.003	-0.820	33.087577	282.140340	0.007	-0.058	32.631083	279.112538	0.026	-0.111
34.936031	276.348519	-0.034	-0.374	32.834729	282.961991	0.035	-0.010	32.631146	278.858687	0.036	-0.138
34.936031	276.348519	-0.028	-0.335	32.962573	282.600657	0.013	-0.054	32.285939	279.568271	0.044	-0.057
34.703888	277.090324	-0.055	-0.339	32.843823	282.536815	0.051	-0.041	32.145814	279.805094	0.020	-0.060
35.008946	276.007301	0.075	-0.806	33.890822	280.325050	-0.006	-0.362	32.125218	279.612751	0.037	-0.058
33.407747	278.617881	0.036	-0.273	33.821958	280.565887	-0.016	-0.370	34.523104	275.972786	0.002	-0.245
33.974307	277.204010	0.032	-0.309	33.665516	280.608782	-0.040	-0.314	33.591875	276.716410	0.016	-0.293
33.682799	277.852543	0.055	-0.320	33.664444	280.681059	0.024	-0.306	32.972571	277.482937	0.007	-0.215
34.518102	281.101450	0.083	-0.603	33.551207	281.190446	0.000	-0.350	32.762113	276.734515	0.030	-0.198
34.411999	281.059750	0.072	-0.632	33.386863	281.274184	0.011	-0.302	32.768512	276.723063	0.043	-0.202
34.387360	281.067594	0.071	-0.641	33.633603	280.795594	-0.046	-0.289	32.762384	276.732983	0.008	-0.178
34.246733	281.507985	0.009	-0.558	33.483807	280.958891	-0.022	-0.298	32.655401	276.792243	0.057	-0.169
34.231614	281.274249	0.048	-0.527	33.828639	280.112193	-0.010	-0.338	32.528733	276.269337	0.026	-0.143
34.129039	281.422252	-0.026	-0.538	33.875508	279.788001	0.013	-0.346	32.868731	278.196027	0.074	-0.201
34.041905	281.588834	0.021	-0.495	33.023702	280.886124	0.020	-0.163	33.040004	277.800583	0.038	-0.233
34.386109	280.986260	0.060	-0.575	33.111972	281.980216	0.038	-0.108	32.501413	278.714939	0.031	-0.107
34.255865	281.109016	0.067	-0.576	33.146197	281.798831	0.063	-0.155	32.421268	278.680406	0.059	-0.122
34.217497	281.062446	0.026	-0.517	32.414120	283.025765	0.010	-0.017	32.325928	278.744502	0.022	-0.117
34.162095	281.324745	0.051	-0.541	32.923417	281.583525	0.068	-0.102	32.838344	277.363105	0.057	-0.200
34.144825	281.091641	0.003	-0.460	32.243998	282.473149	0.037	-0.026	32.241931	278.663746	0.024	-0.075
33.984902	281.830979	0.017	-0.500	31.963325	283.750853	0.021	-0.046	32.169208	278.606407	0.059	-0.090
33.740540	282.248592	-0.052	-0.515	33.671824	279.621307	0.036	-0.314	32.031152	278.721654	0.026	-0.096
33.839305	281.771552	0.056	-0.482	33.558207	279.996218	-0.026	-0.287	32.086652	278.570910	0.017	-0.094
33.931174	281.486700	0.047	-0.410	33.490560	280.028236	-0.022	-0.270	31.747098	279.470128	0.016	-0.039
33.776854	281.709042	0.034	-0.392	33.432376	279.967869	-0.009	-0.295	31.850173	278.963634	0.039	-0.059
33.576588	282.206965	-0.052	-0.399	33.716553	279.396080	0.037	-0.316	31.653798	278.998200	0.011	-0.049
33.732263	281.811834	-0.053	-0.400	33.514819	279.483952	0.040	-0.304	31.551911	279.026193	0.029	-0.025

Priloga B, lastnosti mreže

Podatki mreže

```
net =
    Neural Network object:
    architecture:
         numInputs: 1
         numLayers: 1
       biasConnect: [1]
      inputConnect: [1]
      layerConnect: [0]
     outputConnect: [1]
     targetConnect: [0]

        numOutputs: 1  (read-only)
        numTargets: 0  (read-only)
    numInputDelays: 0  (read-only)
    numLayerDelays: 0  (read-only)
    subobject structures:
            inputs: {1x1 cell} of inputs
            layers: {1x1 cell} of layers
           outputs: {1x1 cell} containing 1 output
           targets: {1x1 cell} containing no targets
            biases: {1x1 cell} containing 1 bias
      inputWeights: {1x1 cell} containing 1 input weight
      layerWeights: {1x1 cell} containing no layer weights
    functions:
          adaptFcn: 'trains'
           initFcn: 'initlay'
        performFcn: (none)
          trainFcn: 'trainr'
    parameters:
        adaptParam: .passes
         initParam: (none)
      performParam: (none)
        trainParam: .epochs, .goal, .show, .time
    weight and bias values:
                IW: {1x1 cell} containing 1 input weight matrix
                LW: {1x1 cell} containing no layer weight matrices
                 b: {1x1 cell} containing 1 bias vector
    other:
          userdata: (user stuff)
```

Uteži in pragi pred učenjem

```
wts =                                                   40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500              40.7742
  504.9650  495.3450    2.3075    6.1500
```

```
biases =
```

Uteži in pragi po učenju

wts =

788.0076	854.9771	3.4567	5.6082
284.3880	113.4420	2.4022	5.8773
464.3909	433.1426	0.3992	6.6978
100.6748	934.9744	0.8682	6.0199
905.0087	224.7170	3.5899	7.4646
475.3654	879.1308	3.4952	6.6009
576.6727	438.2549	0.3833	8.8039
610.4918	299.4112	1.3380	6.7948
308.8533	650.8912	1.0216	5.5367
282.4609	426.4632	1.8071	6.3086
584.9874	538.1255	4.0207	6.0277
401.2842	251.8925	3.5875	4.0430
891.1116	556.4346	1.9433	5.1470
81.4645	498.0843	0.8390	6.2466
637.6917	719.3324	2.3818	5.7522

biases =

28.9371
38.1506
80.9788
44.9848
33.9840
22.0522
75.7037
67.9121
45.6193
45.2643
71.2816
55.1534
30.1694
33.7769
32.3635

Priloga C, prosojnica za nalogo 3