**UNIVERZA V MARIBORU**
FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Marko Brakus

# ADAPTIVNA KLASIFIKACIJA S KOHONENOVO NEVRONSKO MREŽO

Diplomska naloga

Maribor, januar 2010

UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO
2000 Maribor, Smetanova ul. 17

Diplomska naloga univerzitetnega študijskega programa

# ADAPTIVNA KLASIFIKACIJA S KOHONENOVO NEVRONSKO MREŽO

**Študent:** Marko BRAKUS

**Študijski program:** univerzitetni, Računalništvo in informatika

**Smer:** Programska oprema

**Mentor:** red. prof. dr. Nikola GUID

**Komentor:** doc. dr. Damjan Strnad

Maribor, januar 2010

# Adaptivna klasifikacija
# s Kohonenovo nevronsko mrežo

**Ključne besede:** **Kohonen, SOM, klasifikacija, razdelitev v gruče, nevronske mreže**

**UDK:** 004.93:004.8(043.2)

## Povzetek

*V diplomskem delu bomo predstavili Kohonenovo nevronsko mrežo in preučili njeno strukturo, učenje z algoritmom SOM, parametre in uporabo predvsem pri klasifikaciji. Kohonenova nevronska mreža je ena od najpopularnejših zaradi svojih širokih lastnosti. Predvsem jo uporabljajo pri preučevanju in vizualizaciji podatkov.*

*Razložili bomo strukturo, učenje in parametre Kohonenove nevronske mreže in tudi način, s katerim algoritem SOM neznane podatke razdeli v gruče in izgradi klasifikator. Opisali bomo tudi testno programsko opremo.*

# Adaptive Classification
# with Kohonen Neural Network

**Keywords:** **Kohonen, SOM, classification, clustering, neural networks**

**UDK:** 004.93:004.8(043.2)

## Abstract

*In this thesis, we present the Kohonen neural network and study of its structure, parameters, learning with the SOM algorithm, and its use primarily in classification. The Kohonen neural network is one of the most popular because of its diverse features. It is used mainly in analysis and visualization of data.*

*We explain the strucuture and learning of the Kohonen neural network and the way, in which the SOM algorithm clusters unknown data and builds a classifier. A demonstration software is implemented and described as part of this work.*

# Kazalo

# Poglavje 1

# Uvod

Kohonenove nevronske mreže so posebna vrsta samoorganizirajočih preslikav (SOM - *Self Organising Maps*) ali SOFM (*Self Organising Feature Maps*). Ker so najbolj popularna oblika teh map, se večinoma izrazi SOM in Kohonenova nevronska mreža uporabljajo kot sinonimi. Uvedel jih je Teuvo Kohonen* leta 1982. Do sedaj je objavljeno več kot 7000 raziskovalnih člankov na to temo.

Obstajala je potreba po učinkovitem algoritmu, ki bi preslikal vzorce (*patterns*), ki so si po določeni metriki blizu v vhodnem prostoru podatkov, v značilke, ki so si topološko blizu v izhodnem prostoru (*features*). Pri tej preslikavi pride do zmanjševanja dimenzij večdimenzionalnega prostora v diskretni manjdimenzionalni prostor.

Struktura Kohonenove nevronske mreže imitira cerebralni korteks človeških možganov. Čeprav to nobena druga umetna nevronska mreža ne upošteva, je ta del možganov prostorsko urejen, kar pomeni, da sosednji deli korteksa obravnavajo sosednje dele telesa. Na podoben način sosednji deli Kohonenove nevronske mreže obravnavajo sosednje vhodne podatke.

Mreža postane topološko urejena skozi iterativen samoorganizirajoči proces (učenje), ki jo prilagaja vhodnim podatkom. Ta iterativen postopek učenja je tekmujoč, ker se v vsaki iteraciji izbere en zmagovalen nevron, ki je najbližje vhodnemu vzorcu in se potem skupaj s sosednjimi prilagaja temu vzorcu.

SOM je prispeval k vse večji popularnosti nevronskih mrež v začetku 80-tih let 20. stoletja. Postal je najpopularnejši algoritem za nenadzorovano učenje. Uporablja se v

* **Teuvo Kohonen,** dr. ing. (rojen 1934), finski akademik in raziskovalec, trenutno profesor emeritus v Finski Akademiji. Naredil je veliko v polju umetnih nevronskih mrež. Najbolj je znan po samoorganizirajočim mrežama (SOM) zaradi česa je tudi najbolj citiran finski znanstvenik.

razdeljevanju v gruče, klasifikaciji, kvantizaciji vektorjev, pri aktivnem učenju, v verjetnostnih modelih, pri optimizacijskih metodah, vizualizaciji podatkov, v podatkovnem rudarjenju in eksplorativni analizi podatkov.

Glavni namen Kohonenove mreže je torej preučevanje strukture neznanih podatkov in njihova grafična ponazoritev. Njene prednosti so enostavno implementiranje in razumevanje delovanja, čeprav matematičen opis delovanja ni trivialen. Od pomembnega pomena je tudi veliko znanstvenih člankov na to temo.

Diplomska naloga je razdeljena v šest poglavij. V drugem poglavju so prikazane osnove nevronskih mrež, biološka motivacija, različne arhitekture in načini učenja. V tretjem poglavju se ukvarjamo s Kohonenovo nevronsko mrežo. Omenjeni so načini priprave in preučevanja podatkov pred njihovo dejansko uporabo. Po razlagi strukture Kohonenove mreže smo se poglobili v načine in parametre učenja. Na koncu smo prikazali nekaj načinov grafičnega prikazovanja in ocenjevanja mreže. Četrto poglavje obravnava klasifikacijo z vidika Kohonenove nevronske mreže. Prvo je opisana razdelitev mreže v gruče. Potem je razložena uporaba SOM-a kot klasifikatorja in ocenjevanje njegove natančnosti. V petem poglavju opisujemo implementacijo demonstracijskega programa in merimo zmogljivosti naučene Kohonenove mreže kot klasifikatorja pri različnih podatkovnih množicah. V šestem poglavju je podan sklep in v sedmem seznam literature.

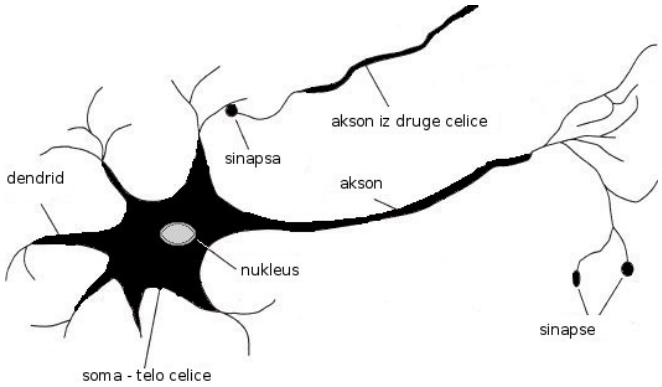
# Poglavje 2

# Osnove nevronskih mrež

## 2.1 Biološke osnove nevronskih mrež

Biološki nevronski sistem (nevronska mreža) se sestoji iz množice medsebojno povezanih nevronov. Število nevronov v možganih je reda 10^11, število spojev (sinaps) med aksoni je reda 10^15 in jih je več vrst.

Sinapsa (*synapse*) je enota, ki upravlja z interakcijo med nevroni. Obstaja jih več vrst. Najpogostejša je kemična sinapsa, ki pretvarja presinaptični električni signal v kemični signal in potem po sprejemu nazaj v električni signal.




**Slika 2.1:***Prikaz nevrona z aksonom, dendritom in sinapsom*

Akson *(axon)* je linija med nevroni, ki prenaša signale v električni obliki. Karakterizira ga visoka električna upornost in velika kapacitivnost. Lahko ga zamišljamo kot kabel, skozi katerega se napetost eksponentno zmanjšuje.

Dendrit (*dendrit*) je drevesu podobna razvejena izrastlina iz telesa nevrona. Ima vlogo

integriranja sinaptičnih vhodov.

Nevroni kot posamezni elementi so nekajkrat počasnejši kot silicijevi logični elementi. Možgani kot celota to kompenzirajo z ogromnim številom nevronov in povezav med njimi. Energetski gledano so možgani kljub številu elementov varčnejši od obstoječih računalniških procesorjev.

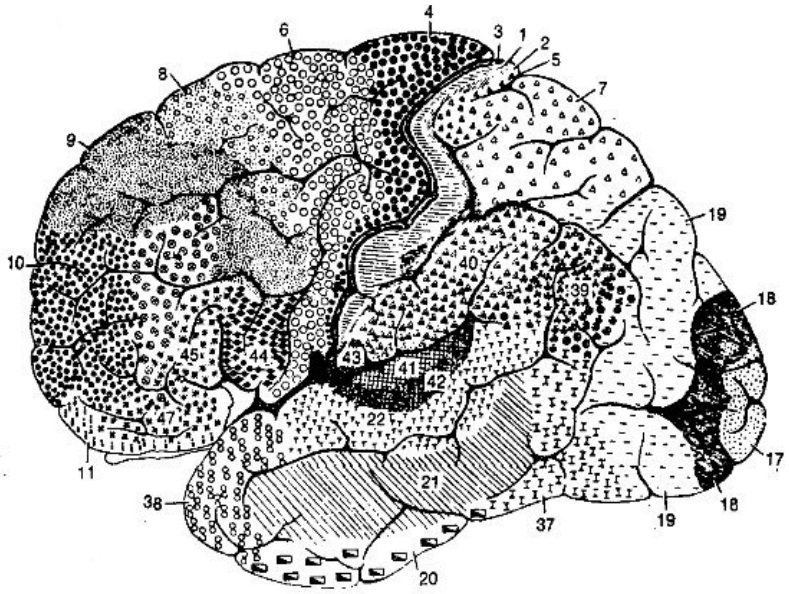
Pogosto se trdi da umetne nevronske mreže niso dovolj učinkovite, če ne imitirajo bioloških dovolj natančno. Veliko umetnih nevronskih mrež, to zlasti velja za nadzorovano učeče umetne nevronske mreže, ne imitirajo dovolj natančno bioloških procesov, ampak opravljajo svojo nalogo. Obstoječe umetne nevronske mreže so zelo primitivne v primerjavi z biološkimi.

Cerebralni korteks (*cerebral cortex*) (slika 2.2) je zunanji del možganov, ki je sestavljen iz nevronov in je sive barve. Ima ključno vlogo pri pomnjenju zavesti, refleksih, itd. Korteks je razdeljen na manjše dele, od katerih vsaki ima drugačno vlogo. Vsaki del korteksa je povezan z drugim sosednjim in spremembe v enem delu povzročajo spremembe v sosednjem delu. Ta lastnost, ki jo večji del umetnih nevronskih mrež ne upošteva, je uporabna iz več razlogov:

- če združimo dele s podobnimi funkcijami skupaj, bodo povezave med njimi krajše,
- manj bo medsebojno sekajočih povezav med regijami, če bodo regije s podobno funkcionalnostjo blizu ena drugi,
- za efektivno predstavitev znanja je pogosto potrebno podobne informacije imeti skupaj in imeti definirano metriko razdalje med njimi.

Struktura, kjer so deli medsebojno povezani in topološki urejeni in kjer sosednji deli korteksa preslikajo sosednje dele čutnih organov telesa, se imenuje možganska mapa.

Mapa se oblikuje skozi samoorganizirajoči proces, ki je lahko evolucija, učenje ali postnatalni rast. Obstajajo mreže adaptivnih (spreminjajočih se) nevronov, ki postajajo bolj podobni določenim lastnostim vhodnih podatkov. Nevroni postajajo med sabo topološki urejeni kot vhodni vzorci. Ker so take mreže običajno dvodimenzionalne, pravimo, da obstaja taka preslikava, ki obdrži topološko urejenost pri zmanjšanju dimenzij reprezentativnega prostora. Iz teoretičnega stališča nas zanima samoorganizirajoči proces, v katerem nastajajo preslikave, podobne možganskim. Večina takih algoritmov so matematični procesi, pri katerih biološke implementacije niso pomembne.




**Slika 2.2:** *Mapa cerebralnega korteksa človeških možganov. Različna področja (Brodmannova področja) so označena z različnimi tipi nevronov. Razvidna je topološka urejenost, kjer sosednja področja obravnavajo podobne funkcije. Motorične funkcije (4), premotorične (6), očesne funkcije (8), somatosenzorne funkcije (3, 1, 2), vizualne funkcije (17, 18, 19), slušne funkcije (41, 42)*

## 2.2 Matematični model nevronskih mrež

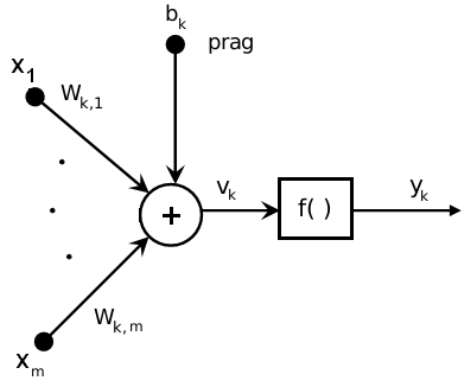
Opisati globalni meteorološki sistem okoli Zemlje je neprimerljivo lažje delo kot opis delovanja človeških možganov. Najbolj primerno orodje za opisovanje nevronskih sistemov so kompliciranе parcijalne diferencijalne enačbe.

V eksaktni znanosti še vedno ne obstaja kvantitativno natančen model procesiranja informacij za kompletni nevronski sistem. Obstajajo samo elementarni modeli.

Način obdelave informacij v možganih je različen od načina v digitalnih računalnikih. Možgani so kompleksen in adaptiven analogni računalnik. Ne nevroni, ne sinapse niso bipolarni pomnilniški elementi. Nevroni se obnašajo kot analogni integratorji z nekaj tisoč vhodov in pragom, ki se časovno spreminja. Taki nevroni niso dovolj natančni, da bi se obnašali kot Boolove funkcije. Vezja v možganih niso avtomati in ne izvajajo inštrukcij, podobnih strojnim.

Kljub temu so pri reševanju veliko vrst problemov možgani veliko hitrejši in učinkovitejši kot katerikoli obstoječi računalnik. Zato jih skušamo čim bolj natančno posnemati z računalniki.

Umetni nevroni se modelirajo kot elementi z več vhodi in enim izhodom, ki je utežna vsota vhodov.




**Slika 2.3:***Nelinearni model nevrona z utežmi, funkcijo vsote, aktivacijsko funkcijo in pragom.*

Na vhodih se nahajajo sinapse, od katerih ima vsaka svoj vhod $x_i$ in utež $w_{k,i}$ . Vse sinapse se skupaj združujejo v seštevalnik (*adder*), ki je linearni združevalnik (*linear combiner*):

Na linearni združevalnik se prišteva dodatni prag (*bias*), ki dodatno zvišuje ( $b_k > 0$ ) ali znižuje ( $b_k < 0$ ) izhod linearnega združevalnika.

Izhodna vrednost nevrona je dodatno omejena z aktivacijsko funkcijo (*activation function*): $y_k = f(v_k + b_k)$ . Vloga ji je držati izhodno vrednost nevrona znotraj določenega intervala, običajno [0,1] ali [-1, 1] in v izhod vpeljati nelinearnost. Nelinearnost je pomembna lastnost nevronske mreže, zaradi katere taka mreža med ostalim lažje imitira polinome višjih stopinj. Sigmoidne ali Gaussove aktivacijske funkcije so najpogostejše.

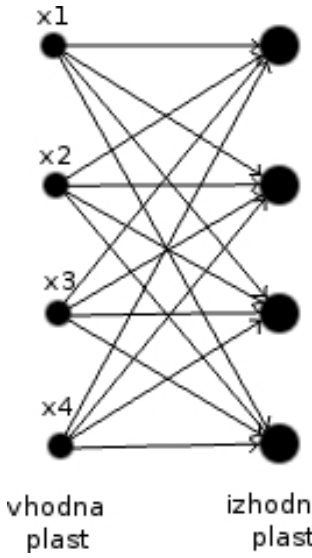
## 2.3 Arhitekture nevronskih mrež

Več povezanih nevronov skupaj tvorijo nevronsko mrežo, ki ima lahko različne arhitekture. Arhitektura nevronske mreže je tesno povezana z algoritmom učenja. Lahko

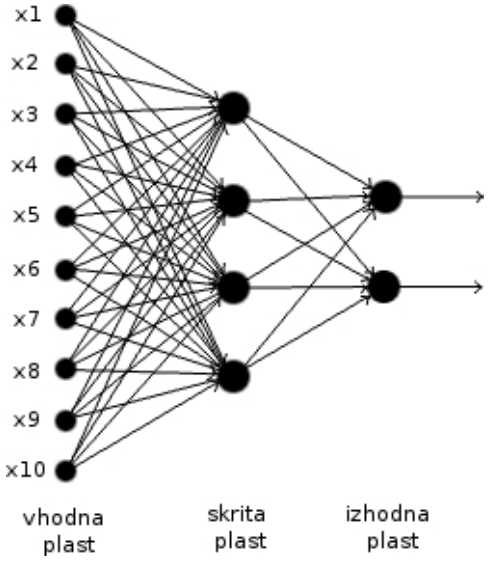
klasificiramo tri osnovne razrede arhitektur nevronskih mrež: enoplastne mreže *(single-layer feedforward networks),* večplastne mreže *(multilayer feedforward networks),* rekurzivne mreže (*recurrent networks*).

### 2.3.1 Enoplastna nevronska mreža *(single-layer feedforward network)*

Vsebuje eno izhodno plast nevronov, v katero se preslika ena vhodna plast podatkov. Ker ni povezav nazaj – samo v eni smeri naprej iz vhodne v izhodno plast, rečemo tej mreži enoplastna nevronska mreža (slika 2.4). Vhodno plast ne štejemo, ker v njej ni računanja.




**Slika 2.4:** *"Feedforward" nevronska mreža brez skritih nevronov.*




**Slika 2.5:** *Popolno povezana feedforward nevronska mreža z eno skrito plastjo nevronov.*

### 2.3.2 Večplastna nevronska mreža

Večplastna nevronska mreža (***multilayer feedforward***) vsebuje eno ali več skritih plasti nevronov (slika 2.5). Zaradi dodanih nevronov in več sinaptičnih povezav taka mreža premore statistiko višjega reda. Nevroni imajo za vhode izhode iz nevronov prejšnje plasti. Ni povezav nazaj v prejšnje plasti. Običajno se mreža z *m* vhodi, $h_i$ nevroni v *i*-tej plasti in *q* nevroni v izhodni plasti zapisuje kot $m-h_i-q$ nevronska mreža.

### 2.3.3. Rekurzivna nevronska mreža

Za razliko od večplastnih mrež rekurzivna mreža ***(recurrent neural network)*** vsebuje povezave nazaj, oziroma, vhod nevronov v plasti je odvisen tudi od izhodov nevronov te plasti, čeprav ne vsebuje zank, ki bi izhod nevrona direktno silili nazaj v vhod tega nevrona. Povratne zanke gredo skozi časovne zamike nazaj v nevron in tudi vse ostale nevrone. Lahko vsebuje nič ali več skritih plasti. Povratne povezave omogočajo mreži dinamično obnašanje.

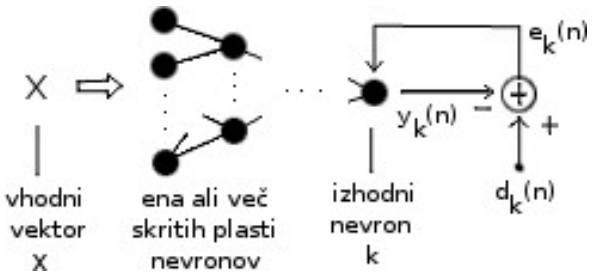
## 2.4 Učenje nevronskih mrež

Primaren proces pri umetni nevronski mreže je učenje – prilagajanje zunanjim podatkom, spodbudam, spremembam ali podobno. Obstaja veliko definicij samega učenja. Algoritem učenja je množica pravil, ki spreminjajo nevronsko mrežo ob stimulansu okolja. Algoritmi učenja se med sabo razlikujejo po načinu, kako prilagajajo sinaptične uteži nevronov.

### 2.4.1 Učenje s popravljanjem napak

Če je znani želeni izhod iz nevrona (nadzorovano učenje), lahko postopoma spreminjamo sinaptične uteži, dokler izhod iz nevrona ni dovolj podoben želenemu izhodu. *Signal napake* $e_k(n)$ pri koraku, oz. diskretnemu času *n* je razlika med dejanskim $d_k(n)$ in želenim izhodom nevrona $y_k(n) : e_k(n) = d_k(n) - y_k(n)$.

Z uporabo *delta-pravila (delta rule)* $\Delta w_{k,i}(n) = \eta e_k(n) x_i(n)$ se pri vsakem koraku *n* spreminjajo sinaptične uteži s hitrostjo učenja (*learning rate*) $\eta$ in proporcionalno produktu signala napake ter vhodov v nevron. Proces se izvaja dokler mreža ne doseže stabilno stanje *(steady state)*.




**Slika 2.5:***Učenje z popravljanjem napak z ponazorjenim signalom napake*

## 2.4.2 Hebbovo učenje

Hebbovo (Donald Olding Hebb*) učenje je eno izmed najpopularnejših in najstarejših učnih pravil in je večinoma nenadzorovano, kar pomeni, da ne potrebuje posebne učne množice podatkov. Zasnovano je na bioloških osnovah medsebojnega delovanja dveh sinaps nevronov. **Hebbova sinapsa** je sinapsa dveh nevronov, ki uporablja lokalen mehanizem za izboljšavo sinaptičnega delovanja kot funkcijo korelacije med predsinaptičnim in postsinaptičnim delovanjem. Lahko rečemo, da se celice, ki se skupaj prožijo (aktivirajo), skupaj tudi povezujejo (*Neurons that fire together; wire together*). Deluje s povečevanjem ali zmanjševanjem sinaptičnih uteži pri izvornu in ciljnu nevronu, ki sta ali nista sprožena istočasno.

Hebbovo pravilo ali učenje je določeno kot sprememba uteži: $\Delta w_{k,i} = \eta\, y_k(n)\, x_i(n)$ kjer je $\eta$ hitrost učenja, $y_k(n)$ je izhodna vrednost (postsinaptična aktivnost) in $x_k(n)$ je vhodna vrednost (predsinaptična aktivnost).

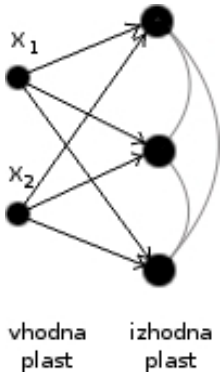
## 2.4.3 Tekmovalno učenje

Značilno za tekmovalno učenje *(competitive learning)* je, da je v določenem koraku (diskretnem času) samo en zmagovalni nevron aktiven. Vsi nevroni dobijo isti vhodni signal in med sabo tekmujejo, kdo bo zmagal. Vsak nevron sčasoma postane prilagojen za določeno vrsto vhodnih podatkov. Takim specializiranim nevronom pravimo **značilke (*feature detectors*)**, ker so

* **Donald Olding Hebb** (1904-1985) je bil kanadski znanstvenik psiholog, ki se je ukvarjal z nevrološko znanostjo in s svojimi postulati doprinesel tudi k razvoju umetnih nevronskih mrež.

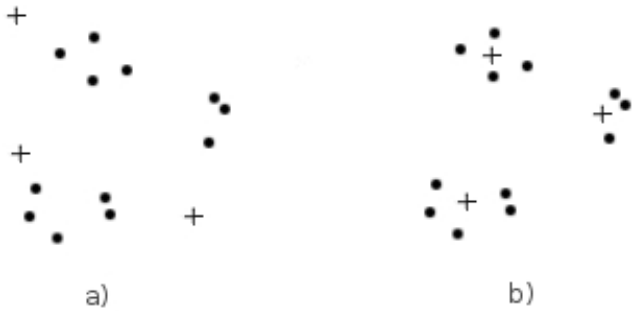
prilagojeni določeni podmnožici vhodnih podatkov in predstavljajo značilnosti te množice. Izhodni nevroni so med sabo povezani z **lateralnimi povezavami**.

Da bi nevron postal zmagovalni pri nekem vhodnem vzorcu *x*, mora imeti največjo vsoto vhodov v nevron. Zmagovalni nevron ima največji izhod. Ostale nevrone ugasne z negativnimi lateralnimi povezavami.




**Slika 2.6:** *Tekmovalna mreža s povezavami iz vhodnih nevronov v izhodne in lateralnimi povezavami med izhodnimi nevroni.*

Zmagovalni nevron se uči s spreminjanjem sinaptičnih uteži po tekmovalnemu pravilu: $\Delta w_{k,i} = \eta(x_i - w_{k,i})$, kjer je $\eta$ hitrost učenja, $x_i$ je vhodni vzorec in $w_{k,i}$ trenutne sinaptične uteži zmagovalnega nevrona. Učinek tega učnega pravila je premikanje uteži zmagovalnega nevrona še bližje vhodnemu vzorcu (slika 2.7). Pri Kohonenovi mreži je upoštevana tudi prostorska topološka urejenost možganskega korteksa in se pri tekmovalnemu učenju ne spreminja samo zmagovalni nevron ampak tudi njegovi sosednji nevroni v omejeni soseski.




**Slika 2.7:** *Geometrična dvodimenzionalna ponazoritev tekmovalnega učenja. Krogi so vhodni vzorci, križi so značilke - sinaptične uteži nevronov. Levo je mreža pred učenjem, ko nevroni niso prilagojeni vhodnim vzorcem. Desno je naučena mreža, ko nevroni postanejo značilke vhodnih vzorcev.*

### 2.4.4 Nadzorovano in nenadzorovano učenje

Pri **nadzorovanemu učenju** obstaja učna množica vhodov in pripadajočih izhodov iz nevronske mreže. Lahko se izračuna (kot pri učenju s popravljanjem napak – delta pravilo) odstopanje od pravilnega (želenega) izhoda in se to uporabi pri prilagajanju sinaptičnih uteži.

Prednost učne množice in poznavanja želenih izhodov za določene vhode je v obravnavanju funkcije (signala) napak kot večdimenzionalne funkcije, kjer so proste spremenljivke dimenzije uteži. Učenje je veliko pohitreno z iskanjem (po možnosti globalnega) minimuma te funkcije z **metodo gradienta**.

Pri **nenadzorovanemu (samoorganizirajočemu) učenju** ni učne množice in se nevronska mreža prilagaja, dokler ne zadosti nekemu podanemu kriteriju. Ko so nevroni dovolj prilagojeni statističnim lastnostim vhodnih podatkov, postanejo značilke in je mreža zmožna avtomatsko določiti razrede. Običajno prilagajanje poteka s tekmovalnim učenjem. Nevroni med sabo tekmujejo, kateri je najprimernejši (običajno najbližji) podanemu vhodnemu vzorcu. Nenadzorovano učenje se konča po določenemu številu iteracij ali ko izpolni podani kriterij.

# Poglavje 3

# Kohonenova nevronska mreža

## 3.1 Predobdelava podatkov

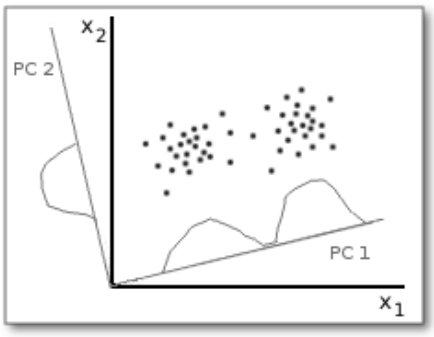
Predobdelava pripravi podatke v obliko, primerno za vhod v algoritem SOM. Pomaga tudi pri določanju primernih vrednosti parametrov algoritma.

### 3.1.1 Projekcije podatkov

Eden od načina poenostavljanja večdimenzionalnih podatkov je projekcija podatkov v točke na običajno dvodimenzionalnemu prikazu. Namen projekcije je zmanjšati dimenzionalnost ob ohranjevanju gruč in ostalih originalnih metričnih lastnosti podatkov.

#### 3.1.1.1 Linearna projekcija PCA

Pri linearni projekciji je vsaka komponenta projiciranega vzorca predstavljiva kot linearna kombinacija komponent originalne podatka. **PCA (*Principal Component Analysis*)** prikazuje večdimenzionalne podatke kot linearno kombinacijo manjdimenzionalnih na način, da ohrani varianco. Pri klasifikaciji je pomembno vedeti, koliko in če sploh lahko zmanjšamo (izberemo) dimenzije vhodnih podatkov, ne da bi izgubili veliko informacij. Ene dimenzije varirajo več kot ostale in so zaradi tega bolj pomembne, ker vsebujejo več informacij. Metoda PCA določi množico medsebojno ortogonalnih vektorjev, na katerih imajo ortogonalne projekcije podatkov največjo varianco (slika 3.1). Tehnika PCA se uporablja pri linearni inicializaciji Kohonenove nevronske mreže.



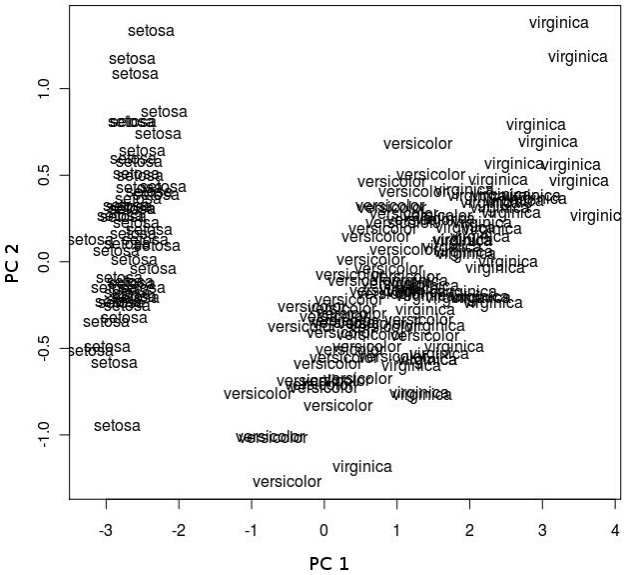

**Slika 3.1:** *Množica dvodimenzionalnih podatkov, ki so projicirani na dve osi - osnovne (principalne) komponente. Principalna komponenta 1 (PCA 1) ima očitno večjo varianco in je bimodalna (z dvema gručama).*

### 3.1.1.2 Nelinearna projekcija – Sammonova projekcija

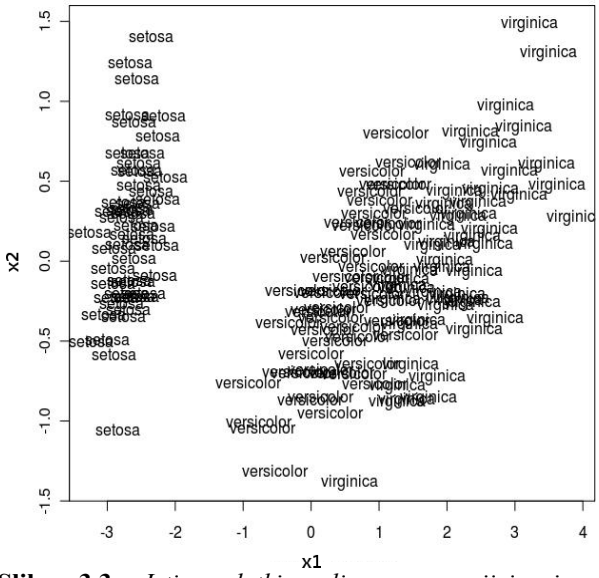
Nesimetrične večdimenzionalne podatke je lažje prikazati z nelinearno kot linearno projekcijo. Obstajajo različne nelinearne projekcije, katerim je skupna lastnost ohranjevanje metrike originalnih podatkov. **Sammonova nelinearna projekcija** je ena izmed takih metod **MDS (*Multidimensional Scaling*)**. Specifičnost te nelinearne preslikave je kvalitetna ohranitev originalnih razdalj v preslikanih dimenzijah nižjega reda (slika 3.3). Sammonova projekcija je zlasti uporabna za prikazovanje porazdelitve in prekrivanja razredov. Vedno je priporočena pred uporabo algoritma SOM [Kohonen, 2001].

### 3.1.1.3 SOM kot projekcija

Algoritem SOM kombinira metode klasterizacije in projekcije. Najbolj pomembne razlike glede na ostale projekcijske metode so, da SOM predstavlja odprto množico vhodnih podatkov s končno množico nevronov, podatki so projicirani v urejeno dvodimenzionalno mrežo (slika 3.4) in ni potrebe po ponovnem računanju projekcije pri dodajanju novega podatka, ki se enostavno dodeli najbližjemu nevronu.




**Slika 3.2:** *Podatkovna množica "Iris" s 4 dimenzijami. Podatki so linearno projicirani na dvodimenzionalno ravnino z PCA metodo.*




**Slika 3.3:** *Isti podatki nelinearno projicirani s Sammonovo metodo. Projekcije sta slučajno zelo podobne.*


**Slika 3.4:** *Isti podatki, projicirani s Kohonenovo nevronsko mrežo (distančna matrika).*

### 3.1.2 Standardizacija podatkov

Osnovni cilj teh treh operacij je spravljanje atributov ali vzorcev v določeni interval. Atribut z veliko večjo varianco bo pri učenju nevronske mreže *zasenčil* drugega, ki ima malo varianco. Pri algoritmu SOM standardizacija vzorcev ni nujna dokler se uporablja Evklidska razdalja za računanje podobnosti med vzorci in uteži nevronov. Priporočena je standardizacija vseh atributov množice ko določeni atributi imajo veliko večje vrednosti od ostalih. Obstaja več načinov standardizacije.

**Skaliranje *(scaling)*** vektorja zajema katerokoli operacijo, ki vektorju prišteje (odšteje) konstanto in jo zmnoži (podeli) s konstanto. Primer je pretvorba temperature iz Celzijusov v Farenheite.

**Normalizacija (*normalization*)** najpogosteje pomeni delitev vektorja z njegovo normo (dolžino – običajno evklidsko):

$$x' = \frac{x}{||x||} \tag{1}$$

Vektor $x'$ postane enotski v intervalu [-1,1]. Normalizacija lahko še pomeni (zlasti pri nevronskih mrežah) ponovno skaliranje vektorja za minimum in rang kar omeji vse komponente vektorja na interval [0,1]:

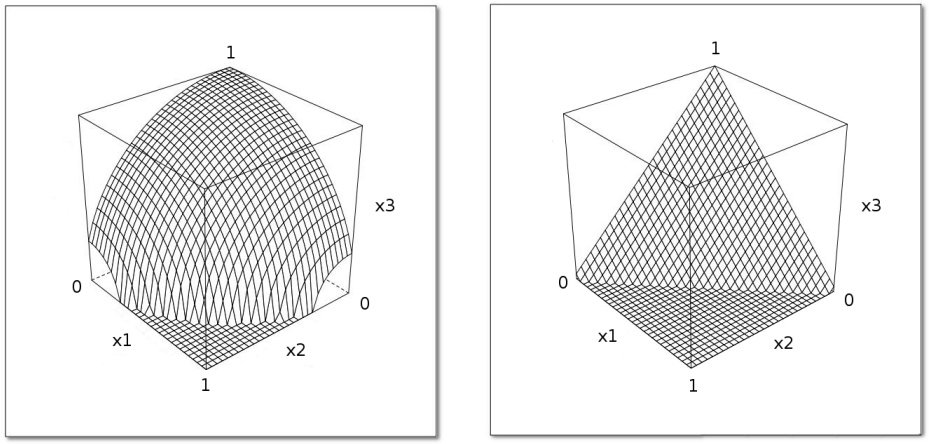
$$x' = \frac{x - min_x}{max_x - min_x} \tag{2}$$

Pogosto se tej normalizaciji reče **normalizacija min-maks (*min-max normalization*)**.

**Standardizacija (*standardization*)** je skaliranje vektorja z odštevanjem mere lokacije in deljenjem z mero skale. To običajno pomeni odštevanje povprečja in deljenje s standardno deviacijo.

$$x' = \frac{x - \mu_x}{\sigma_x} \tag{3}$$

Popularni naziv za to standardizacijo je standardizacija "***zero-mean and unit variance"***, kjer je povprečje nič in standardna deviacija ena in interval [-1,1]. Pogosto se vsem zgornjim metodam skupno reče kar standardizacija.




**Slika 3.5** *Normalizacijske ploskve za dva načina normalizacije, ki jih tvorijo pozitivni normalizirani vektorji. Levo je normalizacija vektorjev s njihovo normo. Desno je "min-maks" normalizacija.*

Učna množica tvori matriko, kjer so vrstice vzorci in stolpci atributi. Standardiziramo lahko vrstice (atribute), stolpce (vzorce) ali oboje. Različni načini standardizacije vhodne množice podatkov bodo imeli različni vpliv na učenje.

**Standardizacija atributov** je standardizacija posameznih stolpcev v matriki učne množice, kjer se standardizirajo posamezni atributi (komponente vektorja ) neodvisno od drugih. Vpliv posameznega atributa pri učenju nevronske mreže je odvisen od njegovega ranga ali variance. Atributi z večjim rangom imajo večji vpliv in če bi radi določili vsem atributom enak vpliv, jih je potrebno vse standardizirati. Standardizacija kateregakoli ranga, ki je centrirana okoli ničle, bo boljša kot tista, ki ni centrirana kot je recimo [0,1] ali še slabše [1,2]. Premajhen rang kot je npr. [-0.1, 0.1] je tudi neustrezen. Centriranost atributov je zlasti razvidna pri hiperravninah v učenju večplastnih nevronskih mrež, kjer se pogosto napačno standardizira v interval [0,1].

**Standardizacija vzorcev** je standardizacija posameznih vrstic v matriki učne množice. Standardizirajo se posamezni vektorji neodvisno eden od drugega. Podobno kot pri standardizaciji atributov imajo posamezni vzorci z večjo varianco ali rangom večji vpliv pri učenju. Taki vzorci lahko *zasenčijo* ostale. Za razliko od standardizacije atributov, ki je vedno ustrezna, je standardizacija vzorcev potrebna samo v nekaterih primerih. V primeru algoritma SOM, ki kot mero razdalje uporablja skalarni produkt (*dot-product SOM*) med vzorcem in

utežmi nevronov, je pomembno vhodni vzorec normalizirati. Če se za metriko pri SOM-u uporablja evklidska razdalja, ni potrebno vzorec standardizirati. Težava, ki nastane pri standardizaciji vzorca, je izguba informacij.

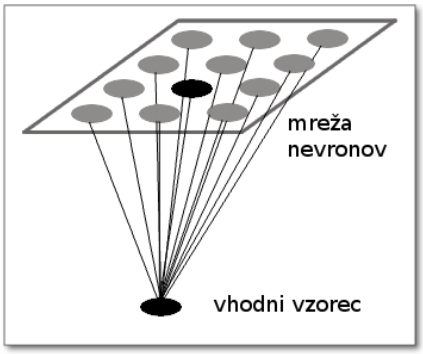
## 3.2 Struktura Kohonenove nevronske mreže

Samoorganizirajoča preslikava (SOM – *Self-Organizing Map* ali SOFM – *Self-Organizing Feature Map*) je specijalna vrsta umetne nevronske mreže. Sestavljena je iz ene (Kohonenova mreža) ali dveh (Willlshaw-von der Marburg* - 1976) plasti nevronov. Večplastne samoorganizirajoče mreže so manj pogoste.

Kohonenova nevronska mreža je posebna vrst samoorganizirajoče nevronske mreže s samo eno plastjo nevronov in je bolj popularna kot dvoplastni model. Temelji na že razloženih bioloških osnovah cerebralnega korteksa. Glavni namen je na adaptiven, samoorganizirajoči in topološko urejen način transformirati vhodne večdimenzionalne vzorce v eno ali dvodimenzionalno diskretno mapo.

---

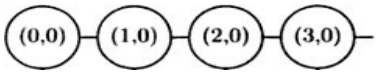
* **Willshaw, David** in **von der Malsburg, Christoph** sta prva postavila samoorganizirajočo nevronsko mrežo (dvoplastno) leta 1978, ampak ni postala tako popularna kot Kohonenova nevronska mreža (1982).



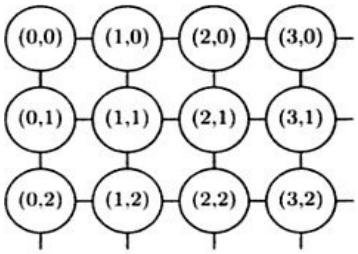

**Slika 3.6** *Enoplastni Kohonenov model samoorganizirajoče preslikave. En vektor na vhodu se primerja z vsemi nevroni v dvodimenzionalni mreži postsinaptičnih nevronov. Označen je en zmagovalni nevron.*

Vhodni vzorec gre kot vhod v vse nevrone, ki se nahajajo v eni plasti nevronov *(lattice)*. Sosednji nevroni so med sabo povezani, oziroma sosednji nevroni v plasti imajo sosednje koordinate.

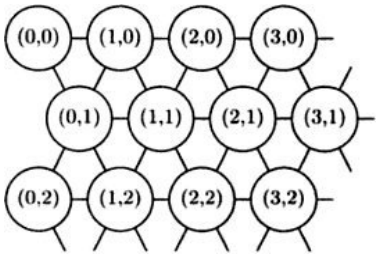
Obstaja več **topologij** plasti nevronov Kohonenove mreže. Pogoste so enodimenzionalne, dvodimenzionalne, dvodimenzionalne-heksagonske. Možne so tudi tridimenzionalne, ki niso pogoste.




**Slika 3.7** *Enodimenzionalna*




**Slika 3.8** *2D pravokotna*




**Slika 3.9** *2D Heksagonalna*

**Topološka urejenost** je določena z načinom razporejanja nevronov po utežeh in koordinatah znotraj plasti. Navadno pravokotno topologijo je lažje obravnavati kot heksagonalno. Kohonenova mreža je topološko urejena, če so nevroni, ki so si sosednji po koordinatah, sosednji tudi po sinaptičnih utežeh.

**Dimenzije mreže** pomenijo število nevronov v obeh smerih plasti. **Dimenzije sinaptičnih uteži** nevronov so vedno enake dimenzijam vhodnega vzorca (številu komponent vhodnega vektora).

Če vhodni vzorec ima, na primer, tri dimenzije, Kohonenova nevronska mreža pa je dvodimenzionalna in ima tridimenzionalne sinaptične uteži, se lahko zaradi takih uteži razpne v tridimenzionalnemu prostoru. Dvodimenzionalno nevronsko mrežo je lažje prikazovati kot tridimenzionalno.

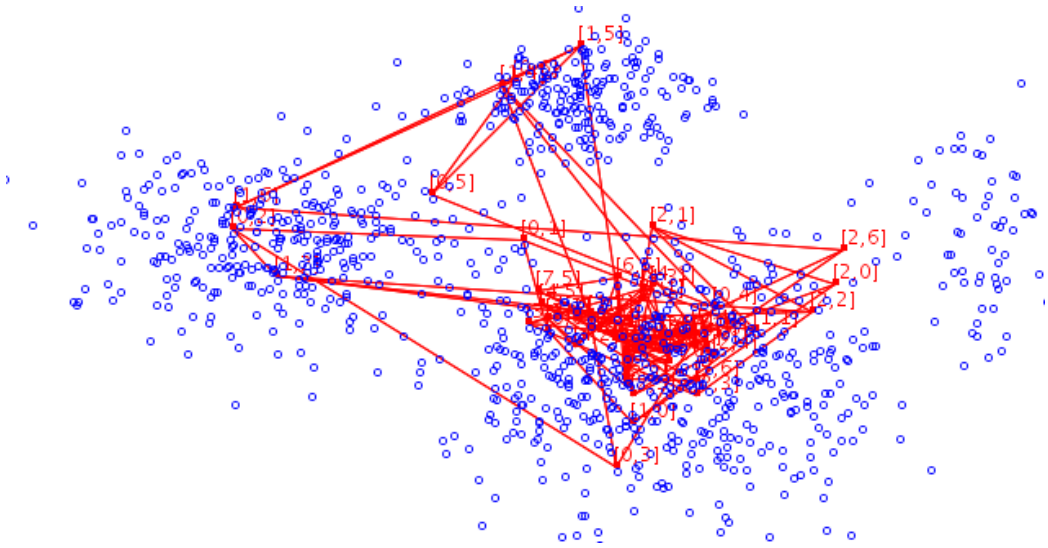
## 3.3 Učenje Kohonenove nevronske mreže

Končni cilj učenja Kohonenove nevronske mreže je čim boljše prilagajanje uteži nevronov podanim vhodnim podatkom ob ohranitvi topološke urejenosti. Na začetku je zaželjeno da so uteži nevronov inicializirane na strukturo ki čim bolj ponazarja vhodne podatke, čeprav tudi naključna inicializacija zadošča (slika 3.10).

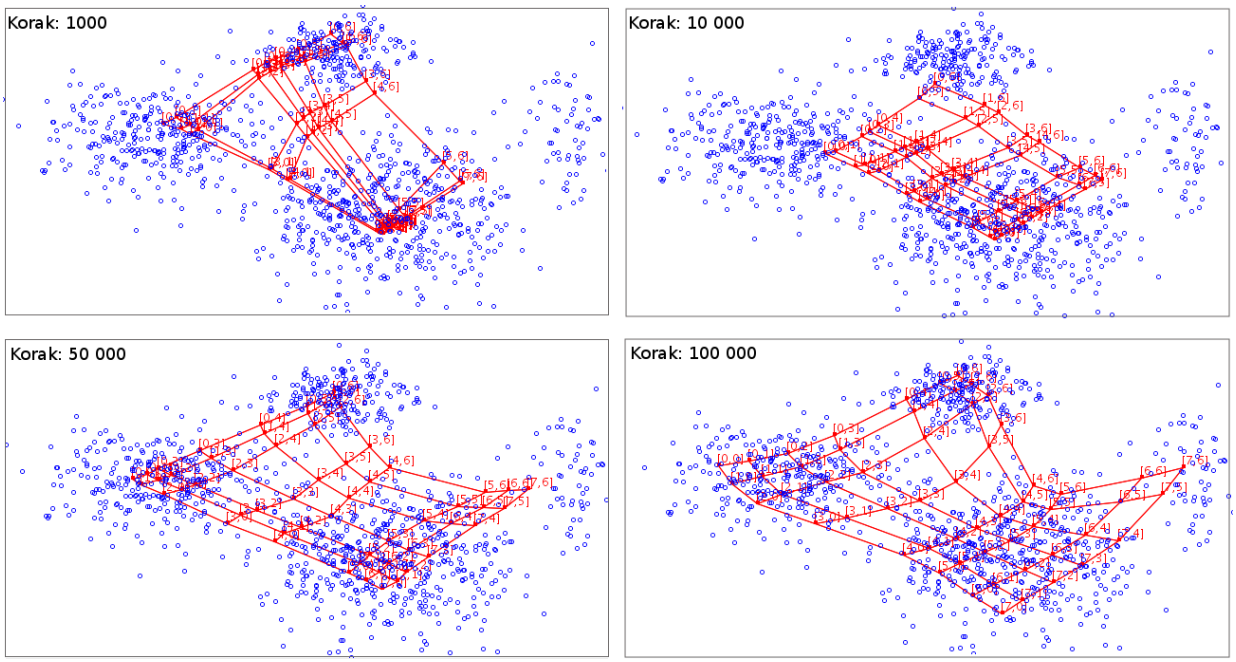
Uteži nevronov se pri vsakem koraku učenja spreminjajo. Pri vsakem koraku (iterativnega) učenja je izbran en vhodni vzorec in njemu najbližji zmagovalni nevron. Uteži tega izbranega nevrona in njemu sosednjih nevronov se spremenijo v smeri vhodnega vzorca. Na začetku so spremembe večje, ker je hitrost učenja večja in, ker je sosedstvo večje. Proti koncu učenja se spreminja samo še en zmagovalni nevron.

Enoplastna samoorganizirajoča adaptivna mreža uporablja tekmovalno učenje za prilagajanje sinaptičnih uteži nevronov vhodnim vzorcem na topološki urejen način.

Učenje je iterativen proces, ki je v osnovi sestavljen iz treh ponavljajočih se faz – kompetitivne, kooperativne in adaptivne (urejanje in konvergiranje) faze. Vse tri faze skupaj zaporedoma sestavljajo eno **epoho** (korak) učenja. Učenje se izvaja več tisoč epoh in se konča pri določenemu kriteriju kvalitete mreže.




**Slika 3.10** *Prvi korak pri učenju. Modro so prikazani vhodni podatki.Rdeče je obarvana Kohonenova nevronska mreža (s označenimi koordinatami nevronov), ki je inicializirana na naključno izbrane vzorce.*




**Slika 3.11** *Celotni postopek učenja Kohonenove nevronske mreže začeti v sliki 3.10.*

### 3.3.1 Tekmovalni proces

Tekmovalni proces je prvi del epohe učenja. V tem procesu se za naključno izbrani vhodni vzorec išče njemu najprimernejši nevron. Ta nevron se imenuje **zmagovalni nevron *(winning neuron)*** in je samo eden. Naključno izbrani vhodni vzorec $x=[x_1, x_2, \ldots, x_m]^T$ in uteži *i*-tega nevrona $w_i=[w_{i1}, w_{i2}, \ldots, w_{1m}]^T$ imata enake dimenzije. Zvezni prostor vhodnih vzorcev je preslikan v diskretni izhodni prostor vzorcev skozi proces tekmovanja med nevroni [Haykin, 1999].

### 3.3.2 Kooperativni proces

Pri kooperativnem procesu določamo sosedstvo in vpliv zmagovalnega nevrona na njo. Ta proces izhaja iz biološke lastnosti, ki pravi da nevron ki se sproži vpliva na nevrone v svoji bližji soseski (podobni razlogi kot pri Hebbovem učenju). Sproženi (zmagovalni) nevron ne vpliva enako na različno oddaljene sosednje nevrone, zato obstaja funkcija sosednosti $h_{j,i(x)}$ (glej podpoglavje 3.4.3), ki je odvisna od razdalje $d_{j,i}$ med zmagovalnim nevronom *i(x)* in *j*-tim

nevronom. Odvisna je tudi od diskretnega časa oz. epohe *n,* ker se velikost soseske običajno zmanjšuje z časom.

### 3.3.3 Adaptivni proces

V adaptivnem procesu se spreminjajo sinaptične uteži zmagovalnega nevrona in sosednjih nevronov. Hebbovo učenje je neprimerno pri samoorganiziranju, ker se spremembe v utežih dogajajo samo v eni smeri. Ta težava se uredi z vpeljavo novega pozitivnega izraza $g(y_i)$ odvisnega od izhoda nevrona.

$$\Delta w_i = \eta y_i x - g(y_i) w_i ,$$

kjer je prvi člen izraza na desni strani Hebbovo učenje, drugi pa je novo vpeljani člen.

Namesto $g(y_i)$ lahko vpeljemo linearno funkcijo $g(y_i) = \eta y_i$ in potem vpeljemo še funkcijo sosednosti namesto izhoda nevrona $y_i = h_{j,i(x)}$.

Po vpeljavi diskretnega časa, oziroma zaporednega števila epohe se dobi končna enačba:

$$w_i(n+1) = w_i(n) + \eta(n) h_{j,i(x)}(n)(x(n) - w_i(n)) \qquad (4)$$

ki se nanese na vse nevrone, ki so znotraj soseske zmagovalnega nevrona *i(x)*.

Z iterativno uporabo zgornje formule se mreža postopoma topološko ureja v smislu, da nevroni, ki so sosednji v mreži, imajo sosednje sinaptične uteži.

Adaptiven proces lahko razdelimo v dve fazi – fazo urejanja in potem fazo konvergiranja.

V **fazi urejanja (*ordering phase*)** adaptivnega procesa mreža skozi približno tisoč epoh dobi grobo topološko urejenost. Hitrost učenja začenja pri 0.1 in se zmanjšuje do približno 0.01. Funkcija sosednosti na začetku vključuje skoraj vse nevrone in se zmanjšuje na koncu do samo enega zmagovalnega nevrona [Haykin. 1999].

**Faza konvergiranja (*converging phase*)** adaptivnega procesa sledi fazi urejanja in je sestavljena iz več epoh, običajno 500-krat več kot je število nevronov v mreži. Hitrost učenja ostaja majhna okoli 0.01. Soseska zmagovalnega nevrona je zmanjšana na en nevron [Haykin, 1999].

### 3.3.4 Inkrementalni algoritem SOM

Algoritem SOM sestavljajo štirje glavni deli: inicializacija, vzorčenje, iskanje zmagovalnega nevrona in posodabljanje mreže. Trije zadnji koraki sestavljajo eno epoho učenja, ki se ponavlja:

1. **Inicializacija.** V izgrajeni nevronski mreži se nastavijo parametri in uteži nevronov po eni od metod.
2. **Vzorčenje.** Izbere se naključni vhodni vzorec $\boldsymbol{x}$.
3. **Iskanje zmagovalnega nevrona.** Vhodni vzorec $\boldsymbol{x}$ se po podani metriki razdalj primerja z vsemi nevroni in se izbere samo en najbolj primerni zmagovalni nevron.
4. **Posodabljanje mreže.** Posodobi uteži zmagovalnega nevrona in nevronov, ki so znotraj njegove soseske po formuli:

$$w_i(n+1) = w_i(n) + \eta(n) h_{j,i(x)}(n)(x(n) - w_i(n))$$

## 3.4 Parametri učenja Kohonenove nevronske mreže

Če mreža SOM ni zelo velika (manj kot nekaj stotin nevronov), izbira parametrov ni zelo pomembna.

### 3.4.1 Topologija in dimenzije

Število nevronov je fiksno skozi učenje. Običajno je privzeto število nevronov $N$ določeno kot:

$$N = 5\sqrt{n}\,, \tag{5}$$

kjer je $n$ število vzorcev [SOMToolbox, 2001]. Mreže z manj nevroni se hitreje učijo (časovna zahtevnost je kvadratična). Pri določanju števila nevronov pomaga ena od projekcijskih metod, kot je PCA ali Sammonova projekcija. Manjša težava nastane, če več nevronov določa eno gručo, kot če je premalo nevronov v mreži za določanje vseh gruč.

Obstajajo enodimenzionalne, dvodimenzionalne, tridimenzionalne mreže in tudi večdimenzionalne, med katerimi so prve dve najbolj pogoste. Dvodimenzionalna se deli na pravokotno in na heksagonalno topologijo.

Heksagonalna topologija je priporočena za vizualizacijo, ker nima poudarjenih horizontalnih in vertikalnih dimenzij kot pravokotna topologija. Z druge strani, popolnoma okrogla topologija ne bi ustrezala, ker ne bi imela orientacije v prostoru. Praviloma je razlika med heksagonalno in pravokotno dimenzijo stvar okusa.

Pravokotna mreža je lažja za implementacijo kot heksagonalna. Ker nevroni aproksimirajo porazdelitev vhodnih vzorcev, je priporočeno imeti dimenzije pravokotne mreže podobne (proporcionalne) glavnim dimenzijam porazdelitve vhodnih podatkov v smeri dveh principalnih komponent.

Tridimenzionalne Kohonenove mreže so tudi možne, ampak niso pogoste v praksi zaradi kompleksnosti. V veliko primerih je dvodimenzionalna mreža boljša za predstavitev tridimenzionalnih podatkov [Haykin, 1999].

### 3.4.2 Začetne vrednosti uteži nevronov

Končni cilj Kohonenove mreže je čim boljši približek porazdelitve vhodnih vzorcev. Več ko je nevronov, bolj se mreža prilagodi vzorcem. S pametnejšo inicializacijo uteži se lahko izognemo prvotni urejevalni fazi učenja. **Naključna inicializacija** je zagotovo najenostavnejša, ampak ni priporočena v praksi. Mišljena je prvenstveno za demonstracijo algoritma SOM, ki lahko katerekoli uteži uredi, če se dovolj dolgo uči ob pravilnih parametrih. Če vnaprej poznamo vhodne podatke, se splača uteži inicializirati na naključno izbrane vhodne vzorce.

Boljša kot naključna je urejena inicializacija, kjer se uteži inicializirajo v dvodimenzionalno topološko urejeno mrežo. Velja pravilo, da če so utežni vektorji inicializirani v katerokoli dvodimenzionalno sekvenco, mreža konvergira večkrat hitreje in zanesljivejše kot z naključno inicializacijo [Kohonen, 2001].

Pomembna lastnost mreže nevronov je, da se z urejanjem prilagajajo dimenzijam vhodnih podatkov, ki imajo največje variance. Zaradi tega lahko uporabimo **linearno inicializacijo** uteži. Pri linearni inicializaciji, kot je že razloženo, se prvo določita dve principalni komponenti

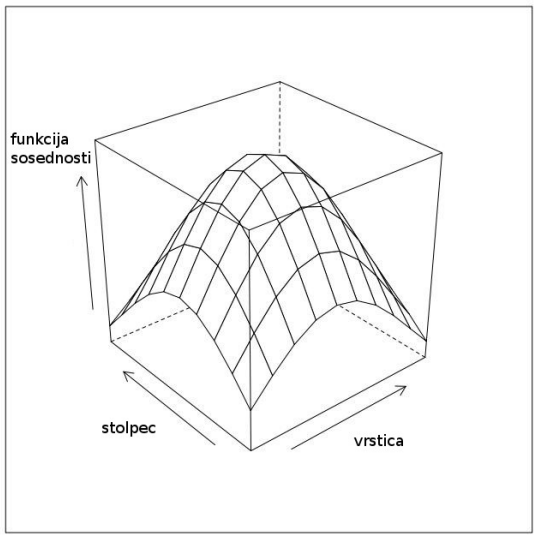
avtokorelacijske matrike vhodnih podatkov, ki imata največji lastni vrednosti. Ta dva lastna vektorja se razprostirata v dvodimenzionalnem linearnem podprostoru. Dvodimenzionalna Kohonenova mreža je definirana v tem podprostoru s centroidom, enakim povprečju vhodnih podatkov.

### 3.4.3 Funkcija sosednosti

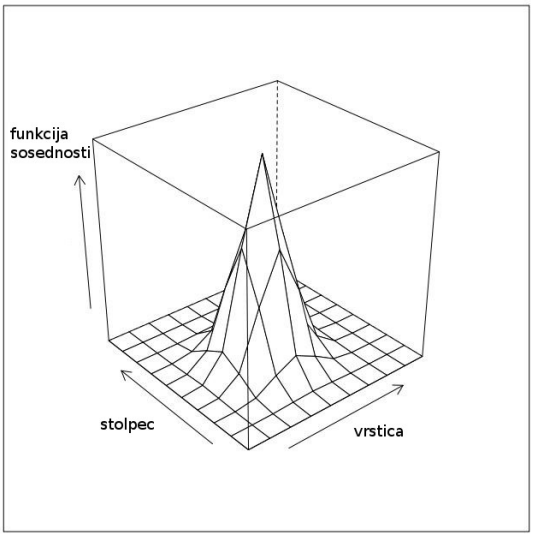
V adaptivnem procesu, kjer se prilagajajo uteži nevronov (formula 4), ima funkcija sosednosti **(*neighborhood function*)** $h_{j,i(x)}(n)$ pomembno vlogo kot jedro *(kernel),* definirano preko množice nevronov v mreži. Funkcija sosednosti pri učenju določa, koliko se zmagovalni nevron in njemu sosednji nevroni spreminjajo in kako čvrsto so nevroni povezani med sabo. Največ se spreminja zmagovalni nevron, najmanj pa nevroni bolj oddaljeni od tega nevrona. Nevroni, ki se ne nahajajo znotraj jedra funkcije sosednosti, se sploh ne spreminjajo. Za konvergiranje mreže je pomembno, da funkcija sosednosti monotono upada proti 0:
Funkcija sosednosti lahko ima različne oblike. Najpopularnejša je Gaussova funkcija:

$$h_{j,i(x)}(n) = e^{\frac{-\|i(x)-j(x)\|^2}{2\sigma^2(n)}} \tag{6}$$

kjer je $\sigma(n)$ monotono upadajoča funkcija ki določa širino jedra.




**Slika 3.12:** *Funkcija sosednosti na začetku učenja dvodimenzionalne Kohonenove nevronske mreže s 10 krat 10 nevronov s zmagovalnim nevronom v [5,5].*




**Slika 3.13:** *Ista funkcija sosednosti pri koncu učenja. Soseska očitno zajema samo zmagovalni nevron in njemu dotikajoče nevrone.*

Taka funkcija, ki je konveksna v središču, pomika (kopiči) sosednje nevrone proti središču te funkcije. Zaradi tega se nevroni, bližji zmagovalnemu, pomikajo več in tisti daljši manj.

Najbolj enostavna oblika funkcije sosednosti, ki se uporablja pri paketni mreži SOM je mehurčna funkcija, ki je določena samo z množico nevronov, sosednjih zmagovalnemu nevronu. Množico indeksov nevronov $N_c(n)$ v času $n$ sestavljajo nevroni znotraj polmera okoli zmagovalnega nevrona. Funkcija sosednosti $h_{j,i(x)}$ je enaka 1, če se nevron nahaja znotraj soseske $N_c(n)$ in 0, če se ne nahaja. Polmer množice se monotono zmanjšuje skozi fazo urejanja.

Veliko pomembnost ima začetna **velikost jedra** funkcije sosednosti. Če se začne z premajhnim jedrom, mreža ne bo globalno urejena. Začetni polmer je običajno večji od polovice premera mreže. Velikost jedra se skozi urejevalno fazo učenja zmanjšuje do enega nevrona in dalje skozi fazo konvergiranja lahko vsebuje samo en nevron.

Če je začetna množica $N_c(0)$, oziroma deviacija funkcije $h_{j,i(x)}$ dovolj velika (večja kot

polmer največje dimenzije mreže), ni rizika, da bi mreža končala v metastabilnih stanjih.

**Oblika jedra** je pomembna enako kot velikost. Pri konveksnih funkcijah ne obstajajo stabilna stanja razen urejenih. Pri konkavnih obstajajo metastabilna stanja, ki lahko upočasnijo proces učenja mreže. Če je funkcija sosednosti na začetku učenja konveksna, tako kot središčni del Gaussove funkcije pri veliki standardni deviaciji, urejanje gotovo doseženo. Večinoma se po urejevalni fazi v fazi konvergiranja funkcija sosednosti zmanjša na en nevron.

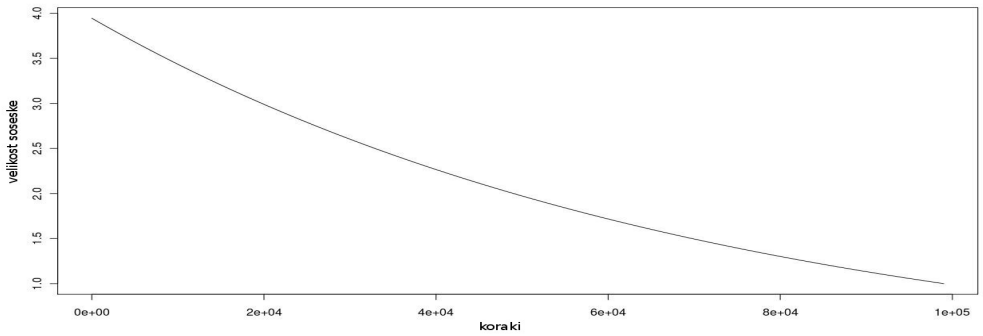
Poleg oblike funkcije sosednosti in velikosti jedra ima veliki vpliv **hitrost spreminjanja** (zmanjševanja) funkcije sosednosti (jedra). Če se jedro prehitro zmanjšuje, lahko mreža konvergira v neurejeno stanje.

Pri Gaussovi funkciji sosednosti velikost soseske eksponentno upada (slika 3.14):

$$\sigma(n)=\sigma_0 e^{\frac{-n}{\tau_1}}, \tag{7}$$

kjer je $\sigma_0$ začetna velikost jedra (običajno polovica diagonale mreže), $\tau_1$ pa je časovna konstanta določena kot:

$$\tau_1=\frac{število\ \ iteracij}{\log(\sigma_0)}. \tag{8}$$




**Slika 3.14:** *Eksponentno upadanje velikosti soseske (ista funkcija sosednosti kot v sliki 3.12 in 3.13) s začetno velikostjo 4 in 100 000 korakov učenja*

### 3.4.4 Hitrost učenja

V procesu učenja (formula 4) je **hitrost učenja (*learning rate*)** $\eta(n)\in(0,1)$ monotono

upadajoča funkcija, odvisna od časa, ki skupaj s funkcijo sosednosti določa velikost spremembe, ko se posodabljajo uteži nevronov v mreži. Če so uteži nevronov inicializirani na naključne vrednosti, hitrost učenja naj bi imela visoke vrednosti (blizu 1) na začetku skozi fazo urejevanja. Vrsta upadanja (linearno, eksponentno, inverzno-časovno) ni zelo pomembna. Po fazi urejevanja (v fazi konvergiranja) naj bi hitrost učenja imela malo vrednost (manj kot 0.02) skozi dolgo periodo. V tej fazi način zmanjševanja tudi ne igra velike vloge.

Eksponentno zmanjševanje (*exponential decay*):

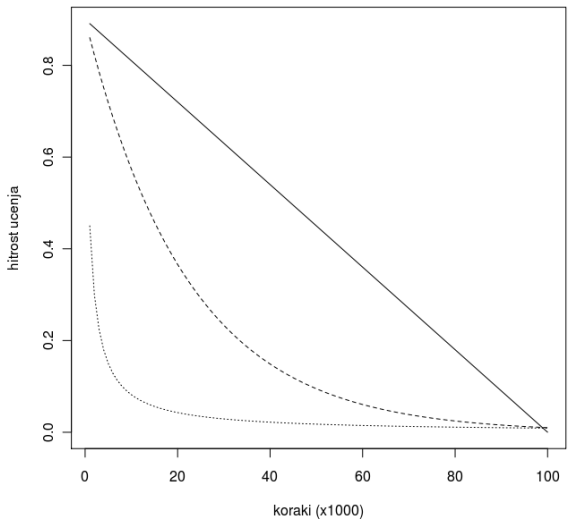
$$\eta(n)=\eta_0\left(\frac{\eta_{končni}}{\eta_0}\right)^{\left(\frac{n}{število\,korakov}\right)} \tag{9}$$

Inverz časa [SOMToolbox, 2001]:

$$\eta(n)=\eta_0\left(\frac{C}{C+n}\right), \quad C=\frac{število\ \ korakov}{100} \tag{10}$$

Linearna hitrost učenja:

$$\eta(n)=\eta_0\left(1-\frac{n}{število\ \ korakov}\right) \tag{11}$$




**Slika 3.15:** *Prikaz različnih funkcij hitrosti učenja s začetno hitrostjo 0.9 in za 100 000 korakov. Linearna hitrost učenja (polna črta v diagonali), eksponentna (črtkana - vmes) in inverz časa (točkasta - spodaj).*

Pri velikih mrežah je potrebno minimizirati čas učenja in izbira optimalne hitrosti učenja je pomembna – običajno je funkcija inverza časa (slika 3.15). Izbira optimalne hitrosti učenja v fazi urejanja ni enostavna, ker se tedaj spreminja tudi širina funkcije sosednosti. V fazi konvergiranja je lažje določiti optimalno hitrost učenja, ker se tedaj širina funkcije sosednosti ne spreminja.

V teoriji lahko posodobimo hitrost učenja pri vsakem koraku za vsak nevron *j* po pravilu:

$$\eta_j(n+1)=\frac{\eta_j(n)}{1+h_{j,i(x)}\eta_j(n)}, \tag{12}$$

kar je teoretična špekulacija, ki v praksi ne deluje vedno na najboljši način.

Lahko določimo povprečno optimalno hitrost učenja isto za vse nevrone:

$$\eta(n)=\frac{A}{n+B}, \tag{13}$$

kjer so A in B primerno izbrane konstante, kar nam daje funkcijo inverza časa (10). Pri taki hitrosti učenja ima vsaki vhodni vzorec približno enako pomembnost. V primeru paketnega algoritma SOM ni potrebno določiti hitrosti učenja, ker tega parametra ni [Kohonen, 2001].

### 3.4.5 Metrike razdalje

Vzorci, ki so obravnavani v algoritmu SOM, se nahajajo v n-dimenzionalnem prostoru, ki ima definirano **metriko (*metric*)**, ki meri razdaljo (podobnost) med vektorji. Razdalja je definirana kot funkcija dveh elementov prostora. Obstaja več različnih metrik razdalje med dvemi vektorji v n-dimenzionalnemu prostoru. Vsaka razdalja mora zadoščati naslednjim pogojem:

**1)** $d(x,x)=0$

**2)** $d(x,y)\geq 0$, kjer enakost velja, samo če je $x=y$

**3)** $d(x,y)=d(y,x)$

**4)** $d(x,y)\leq d(x,z)+d(z,y)$, (najkrajša pot med dvemi točkami je ravna črta)

Standardna razdalja med vzorci $\boldsymbol{x}=(x_1,x_2,\ldots,x_n)$ in $\boldsymbol{y}=(y_1,y_2,\ldots,y_n)$ v algoritmu SOM je **Evklidska razdalja**:

$$d_E(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum (x_i - y_i)^2} \tag{14}$$

Podobne razdalje so **Minkowski razdalja**, ki je posplošitev evklidske:

$$d_M(\boldsymbol{x}, \boldsymbol{y}) = \left( \sum_{i=1}^{n} |x_i - y_i|^{\lambda} \right)^{1/\lambda}, \tag{15}$$

kjer (za $\lambda = 1$ dobimo **Manhattansko razdaljo**, za $\lambda = 2$ evklidsko razdaljo).

**Mahalanobis razdalja** je evklidska razdalja utežena z inverzom kovariančne matrike $\psi$ :

$$d_{MH} = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T \psi (\boldsymbol{x} - \boldsymbol{y})}, \tag{16}$$

ki je uporabna v primerih, ko so vhodne spremenljivke med sabo korelirane.

Možno je uporabljati različne metrike v primerjanju vzorcev, ampak tedaj je potrebno tudi ustrezno spremeniti učno pravilo algoritma, da bo kompatibilno z metriko primerjave.

## 3.5 Algoritem SOM s notranjim produktom

Metrika razdalje oziroma podobnosti med vzorci je pogosto korelacija (*correlation*), kar je tudi najenostavnejša mera podobnosti med dvema vektorjema. Nenormalizirana korelacija $C$ med dvema vzorcema $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ in $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ je:

$$C = \sum_i^n \xi_i \eta_i \tag{17}$$

Če sta $\boldsymbol{x}$ in $\boldsymbol{y}$ obravnavana kot evklidska vektorja, je $C$ njihov skalarni ali notranji produkt (*dot product*).

Taka korelacijska mera podobnosti je najbolj primerna za primerjavo vzorcev, onesnaženih z Gaussovim šumom. Algoritem SOM, ki za metriko razdalje med vzorci uporablja korelacijo (skalarni produkt ali *dot product*), je popularno imenovan SOM z notranjim produktom (*„Dot-Product SOM“*).

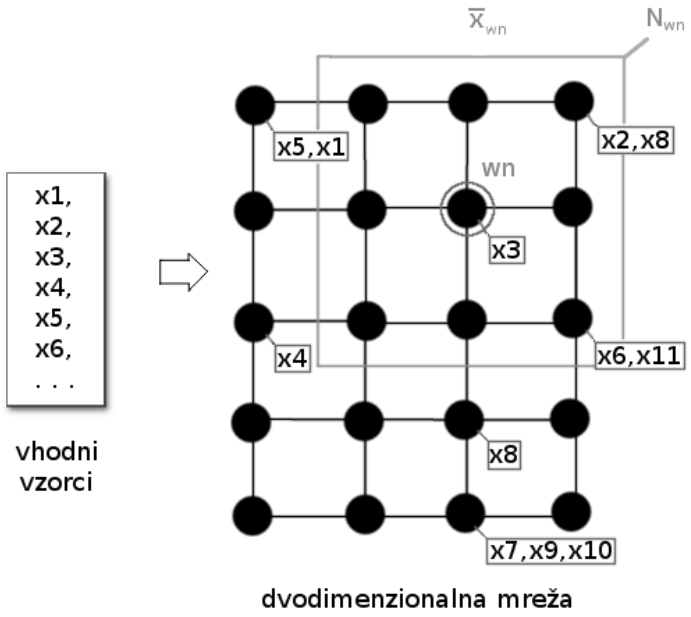
Vzorce ni obvezno normalizirati pred *vhodom* v algoritem s tako metriko, toda je zaželeno, da bi vektorji imeli isti dinamični rang (podpoglavje 3.1.2). Pri spremembi metrike je potrebno ustrezno prilagoditi tudi učno pravilo, ki adaptira nevrone. Korelacijska metrika je hitra

in uporabna v veliko primerov in je skupaj z evklidsko najpogostejša v praksi.

## 3.6 Paketni algoritem SOM

Ob standardnem inkrementalnem algoritmu SOM obstaja še **paketni algoritem SOM (*batch SOM algorithm*)**, ki ni tako preučevan, kot je inkrementalen, ampak ima svoje prednosti. Imenovan je paketni, ker so vsi vzorci podani na enkrat in razdeljeni po nevronih, preden se naredi adaptacija. Bolj je deterministični kot inkrementalni in so njegovi rezultati zato lažje ponovljivi. Pri večjih problemih je vidno hitrejši, ker hitreje konvergira. Ne uporablja parametra hitrosti učenja kar je en parameter manj. Ima manj težav s konvergiranjem in proizvaja stabilnejša stanja nevronov kot originalni inkrementalni algoritem [Fort et al.]. Paketni algoritem SOM je zlasti učinkovit, ko so začetne uteži nevronov že približno urejeni, tudi če še ne aproksimirajo porazdelitve vzorcev.

Paketni algoritem SOM (slika 3.16) je iterativen proces, v katerem se vsi vhodni vzorci razdelijo v pripadajoče(najbližje) jim nevrone. Potem se uteži vsakega nevrona posodobijo glede na vse vzorce ki mu pripadajo. Velikost soseske posameznega nevrona se zmanjša in se postopek ponovi. Funkcija sosednosti je istega obnašanja kot pri standardnem inkrementalnem algoritmu.




**Slika 3.16.** *Prikaz paketnega algoritma SOM, v katerem so vhodni vzorci pri vsakem koraku uvrščeni v sezname pripadajočih vzorcev nevronov, ki so jim najbližji. Kot primer je označen en zmagovalni nevron "wn" in njegova soseska "Nwn".*

Osnovni koraki paketnega algoritma SOM:

0. **Inicializacija.** Uteži nevronov se inicializirajo po enakih pravilih kot pri inkrementalnem algoritmu. Zaželena je inicializacija na katerekoli vzorce.
1. **Razdelitev vhodnih vzorcev.** Vsak vhodni vzorec $x$ se primerja z utežmi vsakega nevrona in doda v seznam vzorcev nevrona, čigavi uteži so najprimernejše (najbližje) temu vhodnemu vzorcu.
2. **Računanje povprečja in posodabljanje nevronov.** Ko so se vsi vhodni vzorci razdelili po pripadajočih nevronih, vsakem nevronu posodobimo utežni vektor glede na povprečno vrednost vseh vzorcev $\overline{x}_{wn}$ znotraj njegove množice sosednosti $N_{wn}$ (slika 3.16).
3. **Zmanjševanje funkcije sosednosti.** Po zmanjševanju funkcije sosednosti se koraki od 1 do 3 ponovijo.

## 3.7 Vrednotenje Kohonenove nevronske mreže

Za iste vhodne podatke lahko dobimo različne mreže, če uporabimo različne parametre učenja. Žal ne obstaja univerzalna najboljša metoda določanja kvalitete mreže.

Ker SOM sodi v kategorijo metod vektorske kvantizacije, je ena metoda računanje **kvantizacijske napake (*quantization error*)**:

$$d(x, n_w) = min_j \left\{ d(x, n_j) \right\}, \tag{18}$$

kjer je $d(x, n_j)$ funkcija razdalje med vhodnim vzorcem $\boldsymbol{x}$ in pripadajočim nevronom $n_j$, ki je vzorcu najbližji v podatkovnemu prostoru [Kohonen, 2001]. **Povprečna kvantizacijska napaka** mreže je dobljena kot povprečje kvantizacijskih napak vseh vhodnih vzorcev:

$$e_q = \frac{1}{N} \sum_{i=1}^{N} d(x, n_w), \tag{19}$$

Kvantizacijska napaka meri, koliko natančno Kohonenova mreža aproksimira porazdelitev vhodnih podatkov. Pričakovano je, da ima najboljša mreža najmanjšo povprečno kvantizacijsko napako. Če mreža ni še dosegla stabilnega stanja v učnem procesu, je kvantizacijska napaka veliko višja kot v stabilnem stanju pri urejenemu optimumu.

Pomemben parameter v algoritmu SOM je funkcija sosednosti $h_{j,i(x)}$, ki opisuje, koliko so zmagovalni nevron in njemu sosednji nevroni povezani.

**Mera distorzije** je definirana kot vsota vseh razdalj od vhodnega vzorca *x* do vseh ostalih nevronov uteženih s funkcijo sosednosti:

$$e = \sum_{j \in L} h_{j,i(x)} d(x, n_j), \tag{20}$$

kjer je *L* množica indeksov vseh nevronov mreže.

**Povprečna pričakovana mera distorzije** mreže je:

$$E = \int ep(x)\, dx = \int \sum_{j \in L} h_{j,i(x)} d(x, n_j) p(x)\, dx, \tag{21}$$

kjer je *p(x)* porazdelitev vhodnih podatkov.

V primeru *N* diskretnih vhodnih podatkov in *M* nevronov je povprečna pričakovana mera distorzije:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{M} h_{j,i(x)} d(x_i, n_j) \quad (22)$$

Lahko rečemo, da je SOM množica nevronov, ki globalno minimizirajo povprečno pričakovano distorzijo – *E*. Natančna optimizacija te distorzije je ekstremno zahtevna za računanje.

Kot meritev topološke kvalitete mreže uporabljamo **topološko urejenost** definirano kot :

$$e_t = \frac{1}{N} \sum_{j}^{N} u(x_i) , \quad (23)$$

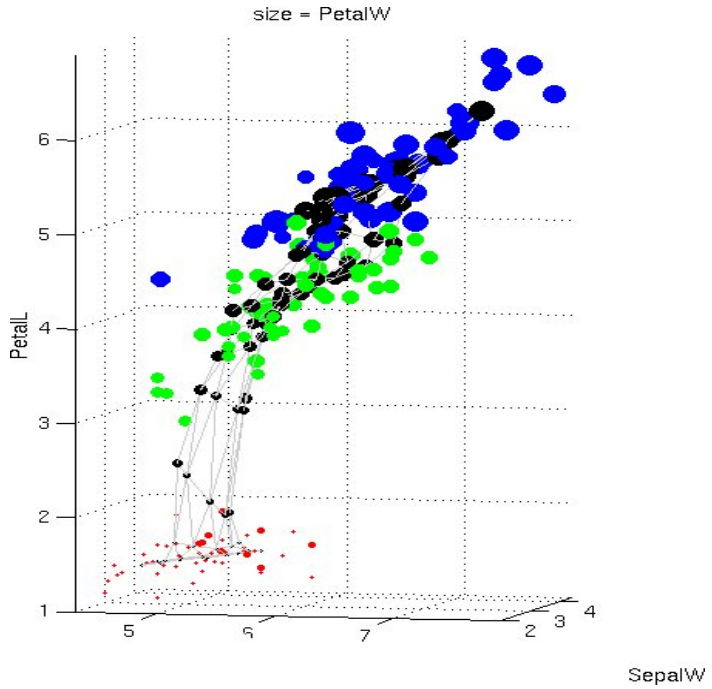
kjer je funkcija $u(x_i)$ enaka 1, če sta najbližji nevron in drugi najbližji nevron sosednja, drugače je 0 [Kiviluoto, 1996]. V tej metodi za vse vhodne vzorce najdemo prvi najbližji nevron in drugi najbližji nevron. Če ta dva najbližja nevrona nista sosednja, topologija mreže ni ohranjena.

Kvaliteto mreže SOM lahko ocenjujemo tudi s primerjavo z ostalimi nelinearnimi projekcijskimi metodami, kot je recimo Sammonova projekcija [Kohonen, 2001].

## 3.8 Grafična ponazoritev Kohonenove nevronske mreže

Glavni namen algoritma SOM je vizualizacija kompleksnih (večdimenzionalnih) podatkov v dvodimenzionalnem prostoru in ustvarjanje abstrakcij kot pri klasterizacijskih in projekcijskih metodah. Namen vizualizacije Kohonenove mreže je ugotavljanje strukture podatkov. Obstaja veliko metod grafične ponazoritve. V primeru manjdimenzionalnih vzorcev je Kohonenovo mrežo možno prikazati kot na sliki 3.17.

Pri večdimenzionalnih podatkih ali pri zahtevnejši analizi gruč je nujno uporabljati metode projekcije in vektorske kvantizacije. Projekcije, kot so PCA, Sammonova ali tudi sam algoritem SOM, zadoščajo namenu približnega ponazarjanja gruč v podatkih. Za bolj natančno predstavitev mej med razredi ustreza ponazoritev **U-matrika (*U-matrix*** ali ***Unified distance matrix*)**. Pri tej metodi so razdalje med sosednjimi nevroni (kvantizacijskimi vektorji) ponazorjene z različnimi odtenki barv (slika 3.18a). Svetlejši odtenki označujejo manjšo povprečno razdaljo med sosednjimi nevroni (taki nevroni skupaj tvorijo gručo), temnejši pa označujejo večjo razdaljo (meja med gručami).

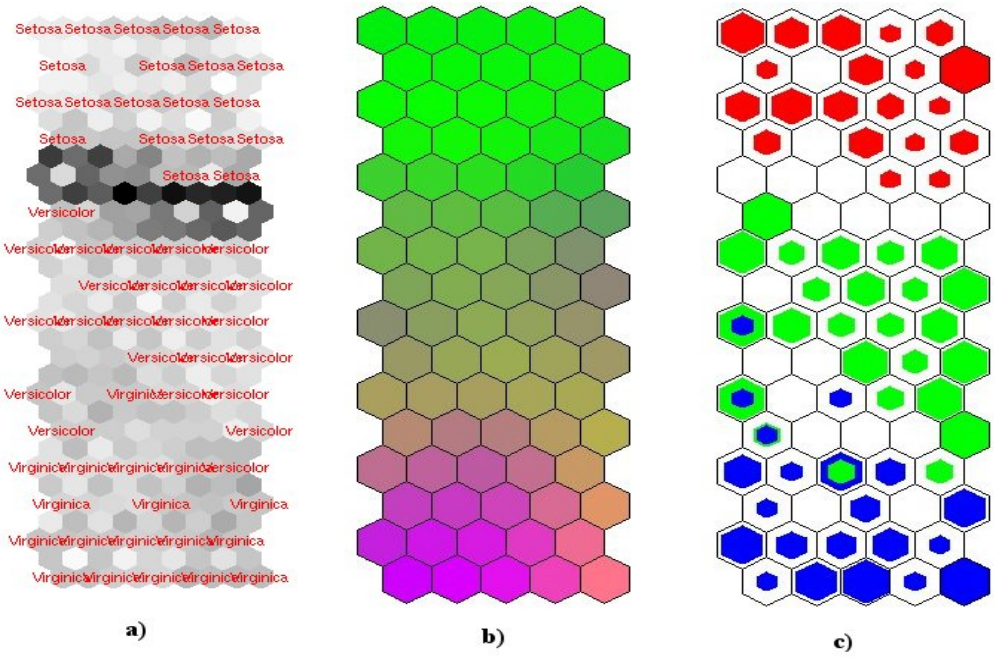



**Slika 3.17.** *[SOMToolbox, 2001] Primer štiridimenzionalnih (velikost kroga je četrta dimenzija) vzorcev podatkovne množice "Iris" razdeljene v treh gručah. Nevroni (povezani krogi) so prilagojeni vzorcem (naučeni).*

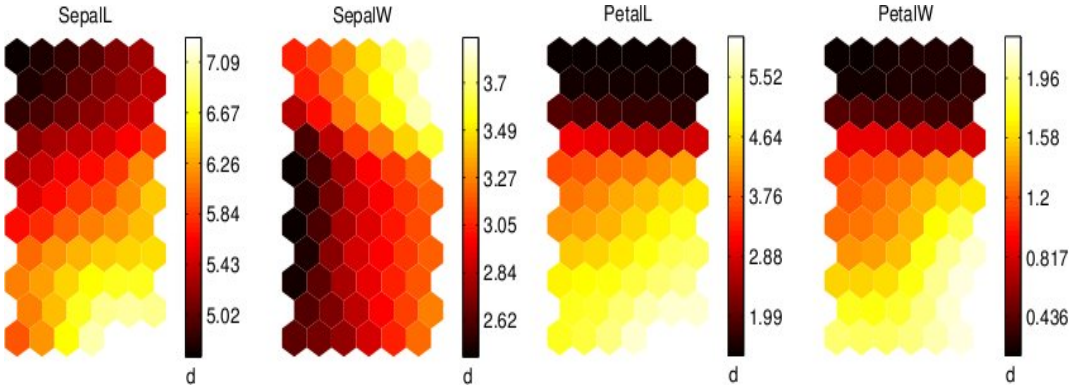
Pri metodi **podobnostnega barvanja *(similarity coloring)*** (slika 3.18b) lahko za vsako gručo uporabimo posebno barvo, kjer imajo nevroni znotraj ene gruče različne odtenke iste barve. Temnejši odtenki barve označujejo večjo pripadnost gruči (manjšo povprečno razdaljo do sosednjih nevronov), svetlejši pa manjšo.

**Histogram zadetkov** ponazarja porazdelitev vzorcev med nevroni (referenčnimi vektorji) (slika 3.18c). Za vsaki nevron je prikazan histogram ki kaže kolikokrat je ta nevron bil zadet (izbran kot zmagovalni) in iz katerega razreda je bil vzorec. Običajno je z barvo prikazano kateremu razredu je ta vzorec sodil. Polja nevronov, ki niso nikoli bili zmagovalni, so prazna.

Za pregled vrednosti posameznega atributa (spremenljivke) v različnih delih mreže se uporablja prikaz **komponentnih ravnin (*component planes*)** (slika 3.19). Iz komponentnih ravnin se vidi vrednost atributa, prikazanega z odtenkom barve v vsakemu nevronu.obrazložitveno moč. Iz komponentnih ravnin se zlahka vidi korelacija med atributi.




**Slika 3.18** *[SOMToolbox, 2001] Množica štiridimenzionalnih podatkov z tremi gručami prikazana na treh načinov.* ***a)*** *U-matrika ponazarja povprečne razdalje med nevroni z odtenki ene barve. V tem prikazu so najbolj razvidne meje (skupine temnejših polj) med gručami (Setosa, Versicolor in Virginica).* ***b)*** *Pri podobnostnem barvanju Kohonenove mreže dodelimo vsaki gruči eno barvo kjer so nevroni iste gruče iste barve in različnih odtenkov.* ***c)*** *Porazdelitev vzorcev glede na referenčne vektorje je ponazorjena s histogramom zadetkov.*




**Slika 3.19.** *[SOMToolbox, 2001] Komponentne ravnine vseh štirih atributov množice podatkov iz slike 3.18. Z odtenki so prikazane vrednosti izbranega atributa v vsakem nevronu.*

## 3.9 SOM kot vektorska kvantizacija

**Vektorska kvantizacija** ***(vector quantization)*** je metoda, ki tvori kvantizirano aproksimacijo porazdelitve vhodnih podatkov s končnim številom vektorjev ***(codebook vectors)***

[Kohonen, 2001]. Poljuben podatek aproksimiramo tako, da poiščemo njemu najpodobnejši referenčni vektor:

$$d(x, n_w) = min_j\{d(x, n_j)\}, \tag{24}$$

pri čemer je $d(x, n_w)$ posplošena razdalja (običajno evklidska) med vhodnim podatkom $x \in \mathbb{R}^n$ in utežnim vektorjem nevrona $n_w$. Očitno vsaki kvantizacijski vektor ima svoje pripadajoče območje okoli sebe v podatkovnem prostoru. To je lahko ponazorjeno z **Voronoijevo razdelitvijo (*Voronoi tessellation*)**.

Množica kvantizacijskih vektorjev je predstavljena s točkami, od katerih ima vsaka okoli sebe omejeno območje. Območja so med sabo razdeljena z linearnimi mejami (hiperravninami v višjih dimenzijah). Vsi vektorji, ki imajo določeni kvantizacijski vektor kot najbližji, tvorijo skupaj eno **Voronoievo množico(*Voronoi set*).** Če so Voronoieve množice optimalnih velikosti, imajo minimalno distorzijo (glej podpoglavje 3.7)– odstopanje pri aproksimaciji.

Algoritem SOM je kvantizacijska metoda, ki tvori Voronoievo razdelitev, kjer so kvantizacijski vektorji uteži nevronov. Dodatna lastnost je, da so nevroni topološki urejeni..

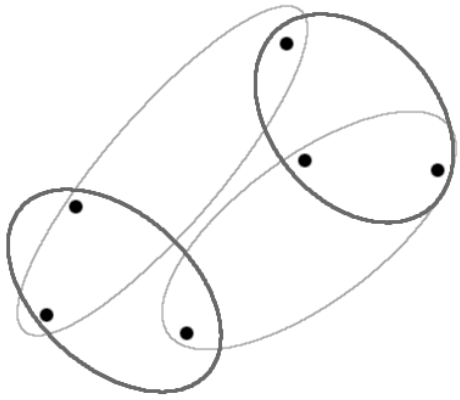
# Poglavje 4

# Klasifikacija s Kohonenovo nevronsko mrežo

## 4.1 Razdelitev v gruče

### 4.1.1 Uvod

Razdelitev podatkov v gruče (razrede, skupine, klasterje) je proces združevanja objektov, ki so si podobni med sabo in različni od objektov v drugih gručah. Splošna ideja, upoštevana pri vseh metodah je minimiziranje razdalje med objekti znotraj gruč in maksimiziranje razdalj med gručami. Razdelitev podatkov v gruče nikakor ni enoumna. Isti podatki se lahko razdelijo na več načinov, tudi ob isti kvaliteti razdeljevanja.



**Slika 4.1.** *Primer razdelitve šest vzorcev v dva razreda. Optimalna razdelitev je označena s temnejšimi elipsami, ker maksimizira razdaljo med gručami in minimizira disperzijo znotraj gruč.*

S statističnega stališča gre pri razdeljevanju v gruče za obravnavanje približka verjetnostne porazdelitvene funkcije vzorcev in določanje mej, ki naj bi tvorile gruče. Meje med gručami so določene s prostori, kjer se nahaja najmanj vzorcev. Kohonenova mreža po učenju določa približek porazdelitvene funkcije vzorcev in nas zanimajo predvsem nevroni v katere se preslika dovolj vzorcev. Pogosto razdelitvi v gruče sledi nadzorovana metoda, ki dodatno izboljša nastale gruče za namene klasificiranja.

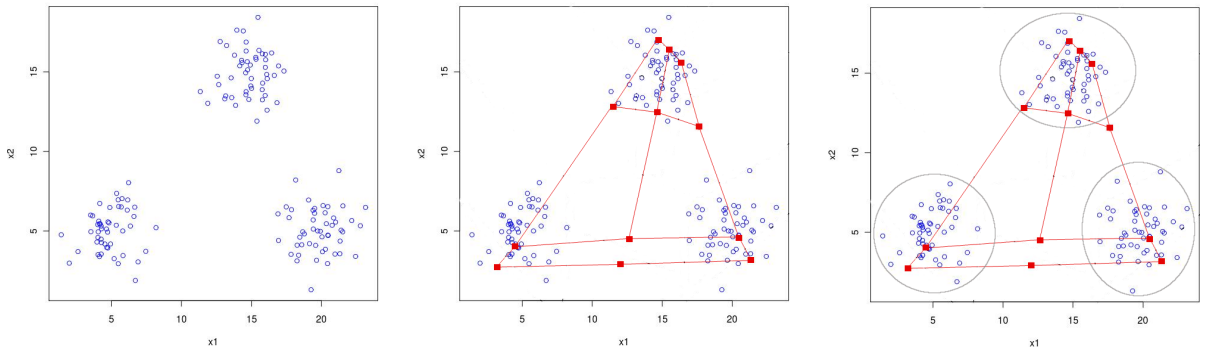
### 4.1.2 Razdelitev SOM-a v gruče

SOM projicira večdimenzionalne vhodne podatke v nevrone manjdimenzionalne mreže, ki efektivno grafično ponazarja lastnosti podatkov. Iz grafične ponazoritve naučene Kohonenove mreže so zlahka razvidne potencijalne gruče z mejami med njimi. Toda, za kvantitativno obdelavo je potrebno skupinam nevronov natančno določiti gruče. Zaželeno je da je ta proces avtomatičen, čeprav je to možno narediti ročno iz grafične ponazoritve kot je distančna matrika.

Če so nevroni naučene Kohonenove mreže dovolj prilagojeni vhodnim vzorcem, preverimo z metodami vrednotenja, kot je kvantizacijska napaka (podpoglavje 4.2.6). Taki prilagojeni nevroni ali prototipi zmanjšujejo število podatkov za razdelitev v gruče ob ohranitvi strukture. Na naučenih nevronih se uporabijo prilagojeni klasični algoritmi. Prilagojeni so zaradi

posebnih lastnosti, ki jih ponuja Kohonenova nevronska mreža. Te lastnosti so topološka urejenost in število podatkov, ki se preslikajo v posamezni nevron mreže.

Tak **dvostopenjski način razdeljevanja v gruče** (slika 4.2) je hitrejši kot direktna razdelitev v gruče vhodnih podatkov, kar je njegova glavna prednost [Vesanto in Alhoniemi, 2000]. Tudi za manjše število vzorcev postanejo algoritmi (zlasti hierarhični) zelo zahtevni in se dvostopenjski način splača. Druga prednost razen hitrosti je zmanjševanje šuma med vzorci. Nevroni so lokalna povprečja podatkov in so zaradi tega manj občutljivi na naključne variacije. Z druge strani je to nezaželeno v primerih, ko nas zanimajo prav take zunaj ležeče vrednosti *(outliers)*. Dvostopenjski način razdelitve v gruče prvo v gruče razdeli nevrone Kohonenove mreže, potem dodeli vsak vhodni vzorec v gručo v katero sodi nevron v kateri se ta vzorec preslika. Pri **nadzorovani razdelitvi mreže** uporabljamo množico klasificiranih vzorcev, in vsaki nevron razdelimo v gručo v katero sodi večina v njega preslikanih vzorcev.

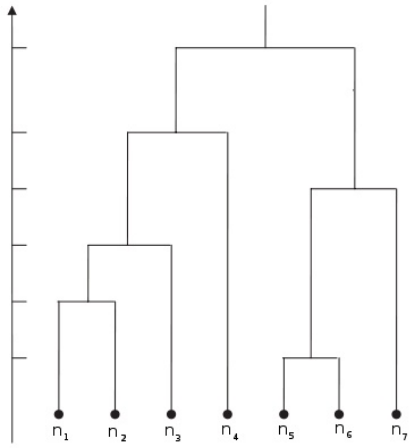



**Slika 4.2.** *Shema dvostopenjske razdelitve v gruče. Namesto da se bi vhodni podatki razdeljevali v gruče, se prvo nauči Kohonenova nevronska mreža, potem se nevroni razdelijo v gruče. V prvi sliki so prikazani vhodni vzorci, v središnji je ponazorjena naučena Kohonenova nevronska mreža (rdeče) in v tretji razdeljevanje nevronov mreže v gruče (elipse).*

Za **nenadzorovano razdelitev mreže** v gruče uporabljamo isto učno množico neklasificiranih vzorcev kot za učenje mreže. Pri nenadzorovanem pristopu se ne zavedamo obstoječih razredov vhodnih podatkov in obravnavamo samo razdalje med nevroni in število v njih preslikanih vzorcev. Možno je uporabiti oba glavna pristopa - partitivni algoritem ali aglomerativni hierarhični algoritem [Vesanto in Alhoniemi, 2000]. Partitivni algoritem naredi ravninsko (*flat*) razdelitev v gruče, tipično ob minimiziranju neke funkcije napake ali zadoščanju pogoja. Popularen partitivni algoritem je metoda **k-povprečja (*k-means*)**. Mi smo v

implementaciji (poglavje 5) uporabili aglomerativni hierarhični. Pri hierarhični razdelitvi v gruče nastane drevesna struktura – **dendrogram**, ki ne določa delitve v gruče enoumno (slika 4.3). Dejanska razdelitev v gruče se dobi z rezanjem dendrograma v določenih nivojih. Pri aglomerativnih hierarhičnih se gruče gradijo od spodaj navzgor *(bottom-up),* kjer se nevroni postopoma združujejo v večje gruče, dokler ena gruča ne vsebuje vse nevrone.

[Taşdemir in Merényi, 2005] sta razložila aglomerativni hierarhični algoritem razdelitve naučene Kohonenove nevronske mreže v gruče, ob upoštevanju topološke urejenosti in števila vzorcev, ki se preslikajo v vsaki nevron. Uporabljeni kriterij za ocenjevanje kvalitete gruče je pravilo: "*Točka v gruči je bližja določeni točki znotraj te gruče, kot katerikoli drugi točki znotraj katerekoli druge gruče.*". Ta kriterij omogoča bolj natančno določanje gruč kot kriteriji uporabljeni v [Vesanto in Alhoniemi, 2000].




**Slika 4.3.** *Shema dendrograma izgrajenega iz sedem nevronov.*

Celotni proces razdelitve nevronov v gruče je razdeljen v **štiri glavne faze**.

**Prva faza** algoritma izgradi dendrogram. Pred izgradnjo dendrograma se naredi **selekcija nevronov**. Dendrogram izgradimo samo iz nevronov v katere se preslika zadostno število vhodnih vzorcev. Pri tem je uporabljen naslednji pogoj:

$$RF_i > \mu_{RF} - \alpha\,\sigma_{RF}\,, \tag{25}$$

kjer je $RF_i$ (*receptive field size*) število vzorcev ki se preslikajo v nevron *i,* $\mu_{RF}$ je povprečno število vzorcev preslikanih v en nevron mreže, $\sigma_{RF}$ je standardna deviacija in $\alpha$ je konstanta ki določa odstotek vzorcev sprejetih v razdelitev v gruče. Večji ko je $\alpha$, več nevronov je sprejetih,

kar ustreza pri bolj kompleksnih podatkih. Manjši $\alpha$ je ustrezen pri enostavnejših podatkih, ker za ponazarjanje njihove strukture zadošča manj nevronov.

Dendrogram se pri aglomerativnem hierarhičnem algoritmu, kot je omenjeno gradi od spodaj navzgor. Na začetku se iz vsakega nevrona naredi ena gruča. Potem se v vsakemu koraku dve najbližji gruči spajata v eno. Postopek spajanja se ponavlja dokler ne ostane samo ena gruča:

Za meritev razdalje (razpršenosti) znotraj gruč se uporablja metoda **maksimalnega najbližjega soseda *(maximum nearest neighbour)***. Pri tej metodi, izmed vseh najmanjših razdalj med nevroni v gruči iščemo največjo:

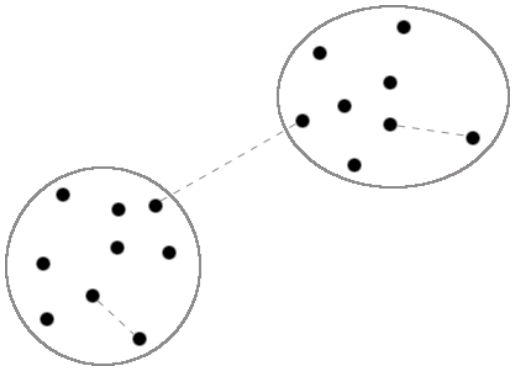
$$d_{nn\_max} = max_i \, min_{j;\, j \neq i} \left[ d(x_i, x_j) \right], \tag{26}$$

kjer je $d(x_i, x_j)$ razdalja med nevroni $x_i$ in $x_j$ iz iste gruče. Zaželeno je uporabljati enako metriko razdalje kot pri učenju Kohonenove nevronske mreže.

Za meritev razdalje med gručami se uporablja **enojna povezava** (***single linkage***). Pri tej metodi iščemo najmanjšo razdaljo med nevroni dveh različnih gruč:

$$d_{slink} = min_{i,\, j} \left[ d(x_i, x_j) \right]. \tag{27}$$

kjer je $d(x_i, x_j)$ razdalja med nevroni $x_i$ in $x_j$ iz dveh različnih gruč.



**Slika 4.4** *Dve gruče s označeno razdaljo med gručami (single link) in razdaljo znotraj posamezne gruče (maximum nearest neighbour)*

V **drugi fazi** algoritma se v dendrogramu označijo potencialne gruče. Element dendrograma je potencialna gruča, če zadošča naslednjem pogoju:

$$\frac{max(d_{nn\_max}(A), d_{nn\_max}(B))}{d_{slink}(A, B)} > \frac{nivo+1}{N-1}, \tag{28}$$

kjer je *nivo* nivo v dendrogramu (koren dendrograma je nivo 0), kjer se nahaja element

(združitev) katerega obravnavamo. *N* je število nevronov uporabljenih v izgradnji dendrograma.

V **tretji fazi** algoritma se potencialne gruče izbrane v prejšnjemu koraku preverjajo, če ustrezajo dodatnem pogoju o številu vzorcev vsebovanih v podgručah in v prostih nevronih. Podgruče (A in B) so otroci potencialne gruče (C) ki so tudi označeni kot potencialne gruče. Prosti nevroni so listi dendrograma ki niso vsebovani v nobeni gruči.

$$\sum_{i \in A} RF_i + \sum_{j \in B} RF_j \; > \; \sum_{k \in (C-(A \cup B))} RF_k \,, \tag{29}$$

Če je število vzorcev ki se preslikajo v nevrone v podgručah večje od števila vzorcev ki so v prostih nevronih (enačba 28), obdržimo to potencialno gručo kot gručo. Če ta pogoj ne zadošča, dalje delimo to potencialno gručo na manjše. Med potencialne gruče so dodani tudi prosti listi.

V zadnji **četrti fazi,** določamo končne gruče s spajanjem gruč nastalih v prejšnji tretji fazi. Prvi pogoj ki se preverja je koeficient maksimalne notranje razdalje in razdalje med gručami:

$$\frac{max(d_{nn\_max}(C_k), d_{nn\_max}(C_l))}{d_{slink}(C_k, C_l)} > 1 \,, \tag{30}$$

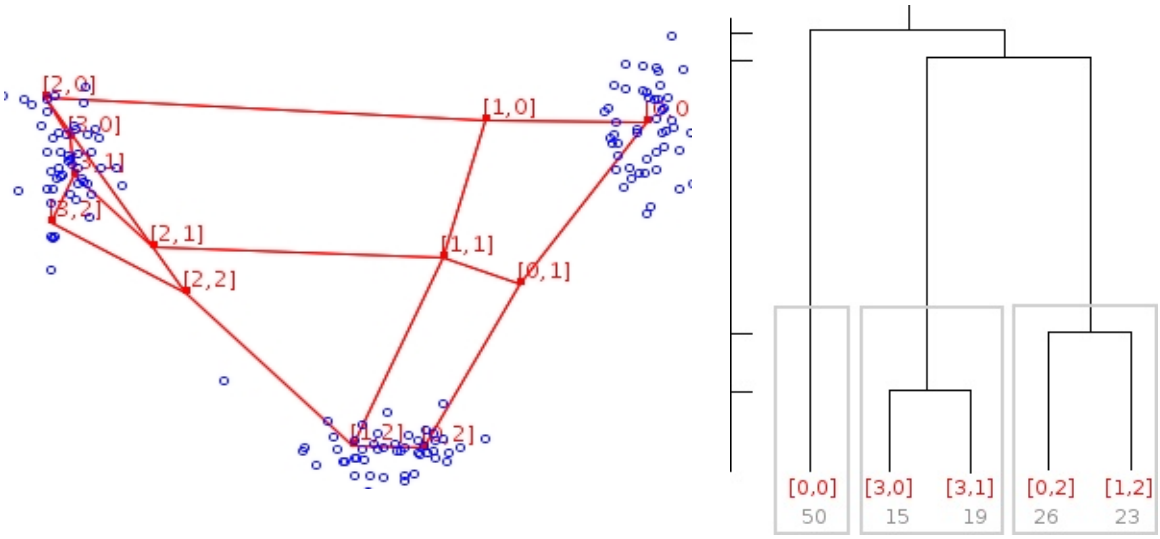
Če pogoj velja, se gruče $C_k$ in $C_l$ združita skupaj. Če pogoj ne velja in če vsaj ena od gruč vsebuje samo en nevron, se preverja še **moč povezanosti** (*connectivity strength*). Moč povezanosti *CONN(i,j)* dveh nevronov *i* in *j,* je število nevronov za katere je nevron *i* ali *j* zmagovalni in drugi nevron drugi zmagovalni [Taşdemir in Merényi, 2005].

$$CONN(i, j) \; > \; \gamma \left( \frac{RF_i}{n_i} + \frac{RF_j}{n_j} \right), \tag{31}$$

kjer je sta $n_i$ in $n_j$ števila sosedov nevronov *i* in *j* , $\gamma$ je konstanta ki z zvišanjem lahko poveča granularnost. Če neenačba velja, se gruči združita.

**Granularnost** (***granularity***) je mera ki opisuje željeno število gruč. Večja granularnost pomeni več gruč, oziroma rezanje dendrograma na višjih nivojih (bližje listom). V zgoraj opisanem algoritmu iz [Taşdemir in Merényi, 2005], je granularnost možno določati v več fazah. Na začetku, pri izbiri nevronov v neenačbi (24), granularnost lahko zvišamo z zvišanjem konstante $\alpha$. V drugi fazi algoritma je možno granularnost povečati z zvišanjem praga na desni strani neenačbe (27). Verjetno najbolj običajen način zvišanja granularnosti je v četrti fazi, pri

računanju moči povezanosti (neenačba (30)), kjer lahko povečamo konstanto $\gamma$ .



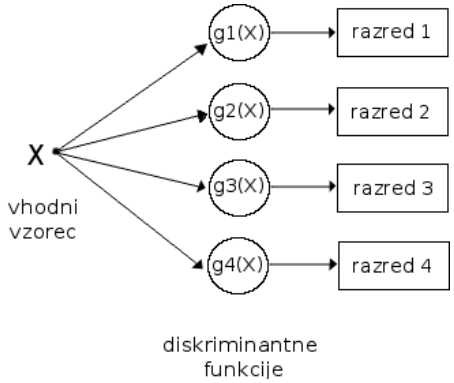

**Slika 4.4** Levo je primer naučene Kohonenove nevronske mreže (rdeče) za tri jasno ločene gruče 150 dvodimenzionalnih podatkov (modro). Izmed 12 nevronov smo izbrali 5 nevronov iz katerih smo v prvi fazi algoritma izgradili dendrogram (desno). V listih dendrograma so narisane koordinate nevronov v dvodimenzionalni mreži skupaj s številom vzorcev, ki se v njih preslikajo. Ker ni bilo potencialnih gruč, smo nevrone združevali šele v četrti fazi algoritma. Sivi pravokotniki v dendrogramu označujejo tri dobljene gruče.

Opazovane objekte je pogosto možno opisati z več njihovimi lastnostmi, toda klasifikacija, ki bi jih vse upoštevala, ni učinkovita zaradi računske zahtevnosti in tudi nezaželjenih pojavov, kot je **prekomerno prilagajanje *(overfitting).*** Ustrezno podmnožico lastnosti (dimenzij) izberemo s postopkom **zmanjševanja dimenzij (*dimensionality reduction*)**. Njegov namen je pred klasifikacijo izbrati podmnožico dimenzij ob čim manjši izgubi informacij. Zmanjševanje dimenzij se realizira na dva glavna načina - skozi postopke projekcije

Matematično gledano je klasifikator preslikava iz zveznega ali diskretnega **prostora značilk *(feature space)*** v diskretno množico razredov (imen razredov). Kako je ta preslikava realizirana, je odvisno od pristopa izgradnje klasifikatorja.

Postopku izgradnje klasifikatorja rečemo **učenje *(learning)***. Učenju, kjer se uporablja **učna množica *(learning set),*** ki je množica že klasificiranih objektov, pravimo **nadzorovano učenje *(supervised learning)***. Če klasifikacija uporablja **nenadzorovano učenje *(unsupervised learning),*** ji popularno rečemo **razdelitev v gruče *(clustering).*** Kohonenova mreža je naučena na nenadzorvani način, čeprav obstajajo nadzorovane metode.

Funkcije, ki opisujejo meje med razredi, imenujemo **diskriminantne funkcije *(discriminant functions)*** *(slika 4.4)*. Klasifikator lahko opišemo kot mrežo ali stroj, ki za podani vzorec ***X*** na vhodu, izbere ustreznu diskriminantnu funkciju *g(X) in* tako določi razred, v katerega ta vzorec sodi.




**Slika 4.4** *Prikaz klasifikatorja kot množice diskriminantnih funkcij. Na vhodu je vzorec ki gre v vsako od diskriminantno funkcijo. Vzorec se klasificira v tisti razred za katerega diskriminantna funkcija da optimalni izhod.*

Nevroni naučene Kohonenove mreže, kot centri Voronoijevih celic, predstavljajo diskriminantne funkcije.

S statističnega stališča se na problem klasifikacije lahko gleda kot na problem določanja približka verjetnostne porazdelitvene funkcije v *n*-dimenzionalnem prostoru in razdeljevanja tega prostora v razrede. Zaradi tega statistika tvori osnove klasifikacije.

Objekt (vzorec) je obravnavan kot naključni vektor ***X*** s komponentami kot naključnimi spremenljivkami $x_i$, ki predstavljajo lastnosti objekta. Torej, vsak objekt je predstavljen kot vektor v *n*-dimenzionalnemu prostoru in množica objektov tvori porazdelitev vektorjev ***X.*** Če poznamo porazdelitve posameznih razredov objektov, lahko tvorimo diskriminantne funkcije ***g(X)*** ki tvorijo meje med porazdelitvami razredov.

Obstajata dva glavna tipa statističnih klasifikatorjev – **parametrični *(parametric)*** in **neparametrični *(nonparametric)***. Pri parametričnih klasifikatorjih je verjetnostna porazdelitvena funkcija posameznih razredov znana. Med takimi je znan **Bayesov klasifikator *(Bayes classifier)***, ki je v vseh primerih optimalen – minimizira verjetnost napačne klasifikacije. Če poznamo priorno (*a priori)* verjetnost $P(\omega_j)$ vsakega razreda $\omega_j$ in pogojne verjetnostne

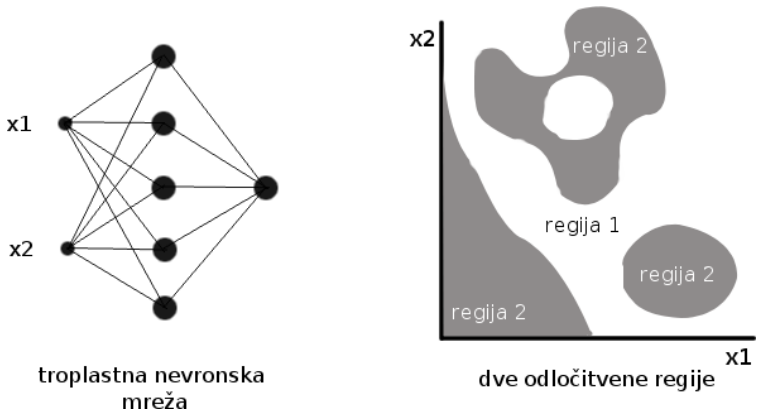
gostotne funkcije $p(x|\omega_j)$, lahko z **Bayesovim pravilom** ***(Bayes rule)*** določimo posteriorne (*posteriori)* verjetnosti $P(\omega_j|x)$ - verjetnosti da podani vzorec ***x*** pripada razredu $\omega_j$

$$P(\omega_j|x)=\frac{p(x|\omega_j)P(\omega_j)}{\sum(p(x|\omega_j)P(\omega_j))}. \tag{32}$$

Bayesovo pravilo tvori teoretične osnove za večino klasifikatorjev. Pogosto se namesto Bayesovega klasifikatorja uporabljajo enostavnejši ampak manj natančni parametrični klasifikatorji, kot so **linearni** ***(linear)*** ali **kvadratični** ***(quadratic)*** [Duda et al., 2004]. Pri neparametričnih klasifikatorjih ne poznamo porazdelitvene funkcije objektov in jo skušamo približno oceniti na nestrukturirani način iz množice vzorcev. V praksi zelo redko vzorci natančno ustrezajo eni od verjetnostnih porazdelitvenih funkcijah. Saj tudi končni namen klasifikatorja ni določanje natančnega približka te porazdelitve, ampak določanje razredov vzorcem ob čim manjši klasifikacijski napaki. Eni neparametrični klasifikatorji delujejo na način, da skušajo določiti približek porazdelitve vzorcev, in če je dovolj natančen se uporablja na parametričen način. Drugi direktno določajo diskriminantne funkcije oziroma posteriorne verjetnosti. Verjetno najbolj znana neparametrična klasifikatorja sta **Parzenov klasifikator** in **metoda najbližjih sosedov (*nearest neighbors method)***.

Nevronske mreže so močno in enostavno orodje za izgradnjo klasifikatorjev. Obstaja veliko primerov, kjer linearne diskriminantne funkcije niso zadostne. Ob nepoznavanju zadostnih parametrov klasifikacije, ni znanih statističnih avtomatiziranih metod, ki bi določile nelinearne diskriminantne funkcije. Prednost večplastnih nevronskih mrež je v enostavnem učenju (prilagajanju uteži nevronov) nelinearnosti. V principu je troplastna nevronska mreža zmožna klasifikacije kateregakoli problema. Kar pomeni, da lahko ob pravilnem številu skritih nevronov in pravilnih utežih lahko ponazarja katerokoli posteriorno verjetnost.

Nevroni v vhodni plasti predstavljajo komponente vhodnega vzorca. Aktivacijske nelinearne funkcije izhodnih nevronov so te, ki predstavljajo različne diskriminantne funkcije. V tipičnem primeru se vhodnemu vzorcu dodeli razred, ki predstavlja izhodni nevron, katerega aktivacijska funkcija daje največji izhod. Običajno je v izhodni plasti enako število izhodnih nevronov, kot je razredov, v katere klasificiramo.




**Slika 4.5** *Troplastna (ali več) nevronska mreža je zmožna implementirati katerokoli odločitveno mejo ne glede na obliko in povezanost*

Pri nadzorovanemu učenju, kjer se v učni množici za vsaki vzorec poda tudi razred, v kateri ta vzorec sodi,de je eden od popularnejših algoritmov nadzorovanega učenja algoritem delta pravila *(delta rule)*.

Glavna težava pri uporabi nevronskih mrež je določanje topologije in kompleksnosti mreže. Vhod je določen z dimenzijo vzorcev, ampak število nevronov v notranjih skritih plasteh pri nadzorovanem učenju ni. Preveč notranjih nevronov lahko vodi prevelikem prilagajanju (*overfitting*), premalo pa k nezmožnosti pravilnega učenja.

Nevronske mreže ponujajo enostaven in močen način izgradbe klasifikatorjev. Žal nas nikakor ne osvobodijo potreb po globljem poznavanju podatkov in domen problema.

### 4.2.2 LVQ

**LVQ (*Learning Vector Quantization*)** je metoda nadzorovanega učenja, ki premika Voronoieve vektorje v podani Voronoievi razdelitvi z namenom optimiziranja velikosti Voronoievih celic [Kohonen, 2001]. Metodo je predložil Kohonen leta 1986. SOM izgradi Voronoievo razdelitev, ki ni vedno optimalna in zaradi tega tudi klasifikacija ni tako dobra, kot bi lahko bila. LVQ izboljša to teselacijo s premikanjem mej med Voronoijevimi celicami (spreminja uteži nevronov).

Vsem vhodnim vzorcem naj bi bili dodeljeni razredi na podlagi že obstoječe klasifikacije. Za vsaki naključno izbrani vhodni vzorec najde njemu pripadajoči (najbližji) referenčni vektor v

Voronoijevi teselaciji, oziroma nevron v mreži SOM. Če izbrani vhodni vzorec in pripadajoči mu referenčni vektor pripadata istemu razredu se vektor premakne k vzorcu. Če ne pripadata istemu razredu, se premakne vektor stran od vzorca.

### 4.2.3 Metoda k-najbližjih sosedov

Pri **metodi najbližjih sosedov (*nearest neighbour method*)** se neznani vzorec klasificira v razred, v katerega sodi njemu najbližji referenčni vektor pri vektorski kvantizaciji oziroma nevron v Kohonenovi nevronski mreži. Taka metoda predstavlja najenostavnejšo obliko klasifikacije.

Od velikega značaja je predhodna kvantizacija z algoritmom SOM, ker bi metoda drugače bila računsko veliko zahtevnejša, če se bi uporabila direktno na veliko klasificiranih vzorcih namesto na manjšem številu referenčnih vektorjev. Pri klasifikaciji s Kohonenovo mrežo bomo uporabljali metodo 1-najbližjih-sosedov.

Porazdelitve različnih gruč oz. razredov vhodnih podatkov se pogosto medsebojno prekrivajo. Metoda najbližjih sosedov ($k = 1$) tega ne upošteva in je priporočeno uporabiti metodo $k$-najbližjih sosedov ($k > 1$). Pri tej metodi se neznani vzorec katerega klasificiramo primerja ne samo s enim ampak s $k$ najbližjimi sosedi in se izbere razred ki je v večini. Lahko se pokaže, da so tako določene meje med razredi po zmogljivostih zelo podobne statistično optimalnim mejam po Bayesovi teoriji [Kohonen, 2001].

### 4.2.4 Kohonenova nevronska mreža kot klasifikator

Po učenju Kohonenove nevronske mreže (iterativno ali paketno učenje) dobimo približek porazdelitve vhodnih podatkov iz učne množice. Nevronov mreže je manj kot vhodnih podatkov, kar je prednost, ki omogoča hitreje obravnavanje iste strukture podatkov.

Taki mreži je možno določiti gruče (podpoglavje 4.1.2) hitreje kot bi jih določili na izvirnih podatkih. Gruče lahko določimo na nenadzorovan ali nadzorovan način. Pri nenadzorovanem načinu ne poznamo pripadnost vhodnih podatkov v razrede, in se zanašamo na razdalje med nevroni (U-matrika), število vzorcev ki se preslikajo v posamezni nevron in tudi na

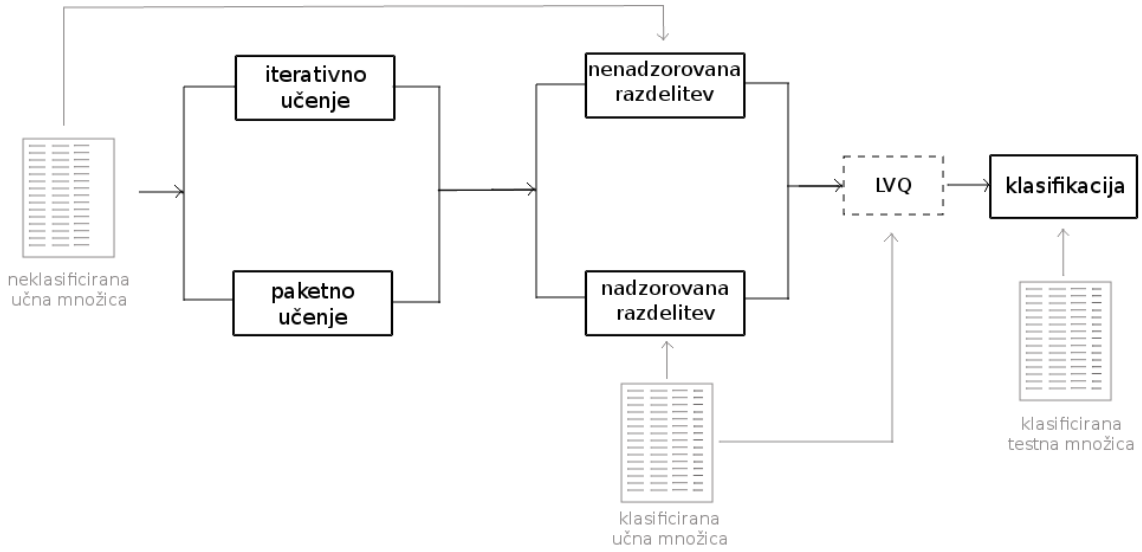
topološko urejenost mreže. Pri nadzorovanem načinu razdeljevanja nevronov uporabljamo klasificirano učno množico in je razdeljevanje natančnejše in zanesljivejše.

Naučeno Kohonenovo mrežo in njeno razdelitev lahko dodatno izboljšamo s uporabo nadzorovane metode LVQ (podpoglavje 4.2.3) ki izboljša uteži nevronov (meje med celicami zamišljene Voronoieve teselacije).

Kombinacija samoorganizirajoče mreže in dodatne nadzorovane metode učenja tvori **adaptivno klasifikacijo** ***(adaptive pattern classification)***ki je hibridna [Haykin, 1999]. Za klasifikacijo je mišljena nadzorovana metoda LVQ, ki je inicializirana s Kohonenovo nevronsko mrežo. Adaptivnost je lastnost vseh nevronskih mrež, ki omogoča spreminjanje (adaptiranje) sinaptičnih uteži glede na spremembe v okolju (vhodni učni vzorci). Kohonenova nevronska mreža se kot prvi korak adaptivne klasifikacije prilagaja (adaptira) vhodnim podatkom in ponazarja njihovo strukturo. Potem LVQ metoda, kot drugi del adaptivne klasifikacije izboljša mrežo (meje zamišljenih Voronoievih celic) in s tem izboljša natančnost klasifikacije.

Kohonenova nevronska mreža je, kot nelinearna projekcija zmožna tudi izbire značilk *(feature extraction),* ki je prvi korak pri klasifikaciji. Po razdelitvi Kohonenove nevronske mreže v gruče dobimo določeno število nevronov, za katere vemo v katere gruče sodijo. Množica vseh teh nevronov tvori klasifikator. Pri klasificiranju določenega vhodnega vzorca poiščemo njemu najbližji nevron in dodelimo vzorec v razred (gručo) v kateri sodi ta nevron. To je algoritem 1-najbližjih sosedov uporabljen na množici nevronov Kohonenove nevronske mreže.

Natančnost take klasifikacije je odvisna od optimalnosti zamišljene Voronoieve teselacije, ki jo tvori naučena Kohonenova nevronska mreža kot vektorska kvantizacija. In še več od razdelitve nevronov v gruče. Če nevron sodi v napačno gručo, bo tudi klasificiral vse vhodne podatke ki se v njega preslikajo v to napačno gručo.




***Slika 4.6*** *Prikaz celotnega postopka uporabe Kohonenove nevronske mreže za klasifikacijo. Mrežo učimo s neklasificirano učno množico na iterativni ali paketni način. Razdelitev mreže v gruče poteka na nenadzorovan ali nadzorovan način. Končno razdeljeno mrežo je možno izboljšati s LVQ metodo, preden jo uporabimo kot klasifikator.*

### 4.2.6 Vrednotenje klasifikacije

Ne obstaja najboljši univerzalni klasifikator in je običajno potrebno preizkusiti več metod in modelov pri klasifikacijski nalogi.

Pri izgradnji klasifikatorja obstaja kompromis med **prevelikim prilagajanjem (*overfitting*)** in **posplošitvijo *(generalization)***. Možno je izgraditi klasifikator, ki 100% natančno klasificira podane podatke, ampak pogosto ob ceni kompleksnosti. Preprilagojeni klasifikator ima prekompleksne meje med razredi, ki niso zmožne pravilno določiti razred neznanim vzorcem. Preveč prilagojen je na primer polinom previsoke stopnje, nevronska mreža s preveč skritimi nevroni ali preveliko odločitveno drevo. Ena od najpomembnejših težav v klasifikaciji vzorcev je določanje primerne kompleksnosti modela. Model naj ne bi bil preveč enostaven za opisovanje razlik med učnimi podatki in ne toliko kompleksen, da ne bi bil napačno klasificiral

nove neznane vzorce [Duda et al., 2004].

Tako kot ne obstaja najboljši univerzalni klasifikator, žal tudi ne obstaja splošna univerzalna metoda primerjanja klasifikatorjev. Uspešnost klasificiranja neznanih vzorcev je ocenjena z **napovedovalno natančnostjo *(predictive accuracy)*,** ki pomeni razmerje uspešno klasificiranih (napovedovanih) neznanih vzorcev in je pogosto najpomembnejši kriterij pri ocenjevanju. Ker je neznanih vzorcev neskončno veliko, napovedovalno natančnost približno ocenimo z množico vzorcev, ki so klasificirani, ampak niso uporabljeni pri učenju klasifikatorja. Eni od glavnih načinov za tovrstno ocenjevanje so delitev množice vzorcev v učno in testno – **treniraj-in-testiraj** (***train-and-test***) in **navzkrižna validacija *(cross validation)***.

Pri delitvi množice vzorcev v učno in testno uporabimo učno množico za izgradnjo klasifikatorja in testno množico pozneje za ugotavljanje števila pravilno klasificiranih vzorcev. Napovedovalna natančnost $p = C / N$ je dobljena kot razmerje med številom pravilno klasificiranih vzorcev $C$ in številom vseh vzorcev testne množice $N$. Pomanjkljivost te metode je, ker ne uporablja vseh možnih vzorcev za učenje, čeprav pri velikih podatkovnih množicah (dosti več kot 1000 vzorcev) to ni velika težava.

Metoda navzkrižne validacije *(cross validation)* je bolj primerna za manjše število vzorcev. V osnovi se množica vzorcev razdeli v $k$ podmnožic in se vsaka enkrat klasificira s klasifikatorjem, ki je zgrajen z uporabom ostalih $k-1$ podmnožic kot učnih. Napovedovalna natančnost $p = K / N$ je dobljena kot razmerje skupnega števila pravilno klasificiranih vzorcev $K$ in števila vseh vzorcev $N$.

**Preglednica 4.1** *Matrika zmedenosti za* ***a)*** *sedem razredov, kjer so v diagonali števila pravilno napovedovanih vzorcev, ob diagonali pa števila napačno klasificiranih vzorcev.* ***b)*** *Matrika zmedenosti za dva razreda - pozitivnega in negativnega*

| Pravilna klasifikacija | Klasificirano kot | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 5 | 6 | 7 |
| 1 | 52 | 10 | 7 | 0 | 0 | 1 |
| 2 | 15 | 50 | 6 | 2 | 1 | 2 |
| 3 | 5 | 6 | 6 | 0 | 0 | 0 |
| 5 | 0 | 2 | 0 | 10 | 0 | 1 |
| 6 | 0 | 1 | 0 | 0 | 7 | 1 |
| 7 | 1 | 3 | 0 | 1 | 0 | 24 |

a)

| Pravilna klasifikacija | Klasificirano kot | |
|---|---|---|
| | + | − |
| + | pravilno pozitiven | krivo negativen |
| − | krivo pozitiven | pravilno pozitiven |

b)

Pri ponazoritvi uspešnosti klasifikatorja pomaga **matrika zmedenosti** ***(confusion matrix)***, ki je matrični način prikazovanja rezultatov testiranja klasifikatorja. Matrika vsebuje eno vrstico in stolpec za vsaki možni razred, kjer so vrstice pravilne klasifikacije in stolpci napovedovalne vrednosti. Element matrike $c_{ij}$ vsebuje število vzorcev iz razreda $i$, ki so klasificirani v razred $j$. V idealnem primeru ko, ni napačnih klasifikacij, imajo vsi elementi matrike razen dijagonalnih vrednost nič [Bramer, 2007].

V primeru klasifikacije samo v dva razreda, običajno enemu rečemo **pozitiven** ***(positive)*** in drugemu **negativen** ***(negative)***. Tudi ko je razredov več, jih lahko razdelimo v en pozitiven in več negativnih razredov. Diagonala (preglednica 4.1) vsebuje število vzorcev iz pozitivnega razreda pravilno klasificiranih v pozitivni razred *(true positives)* in število vzorcev iz negativnega razreda pravilno klasificiranih v negativni razred (*true negatives*). Ob diagonali so števila napačnih klasifikacij. Dva tipa napačnih klasifikacij so *false positives* in *false negatives.* Pogosto se uporabljajo okrajšane oznake (TP, FP, TN, FN). Napovedovalna natančnost:

$$p = \frac{(TP + TN)}{(P + N)} \tag{33}$$

je iz matrike zmedenosti, dobljena kot razmerje med številom pravilno klasificiranih (dijagonala matrike) in številom vseh vzorcev.

V matriki je zlahka vidna neuravnoteženost med razredi, ko je število napačnih

klasifikacij veliko večje pri enem razredu kot pri ostalim. Ocena napovedovalne natančnosti je odvisna od tega in se nezaželeno spreminja ob spremembi razmerja med pozitivnimi in negativnimi razredi. Zato obstajajo ostale mere zmogljivosti klasifikatorja, kot so **razmerje TP (*TP rate)*** = TP/P in **razmerje FP *(FP rate)*** = *FP/N,* ki so razmerja med številom pravilno klasificiranih pozitivnih vzorcev ali napačno klasificiranih negativnih vzorcev in številom vseh vzorcev. Prednost kombinacije takih mer, splošno dobljenih kot razmerje iz različnih vrstic matrike zmedenosti, je ta, da za razliko od napovedovalne natančnosti niso odvisne od relativnega števila pozitivnih in negativnih vzorcev. Razmerje TP in razmerje FP se lahko ponazorita v **ROC grafu *(Receiver Operating Characteristics graph)*** kjer je razmerje TP ordinata, in razmerje FP abcisa.

# Poglavje 5

# Implementacija in primeri

Večino snovi, razložene v prejšnjih poglavjih, smo implementirali v demonstracijski programski opremi, napisani v programskem jeziku Java. Kohonenovo nevronsko mrežo je možno naučiti, ob različnih parametrih učenja, na oba opisana načina - iterativni in paketni. Naredili smo osnovne grafične prikaze vhodnih podatkov in naučene Kohonenove nevronske mreže, kot so komponentne ravnine, distančna matrika in razpršeni graf. Grafično se prikaže tudi pripadnost nevronov v gruče po opravljeni razdelitvi mreže v gruče.. Razdeljeno Kohonenovo nevronsko mrežo smo uporabili za adaptivno klasifikacijo vhodnih podatkov.

## 5.1 Učenje

Oba načina učenja - iterativni in paketni sta si podobna in uporabljata isto strukturo Kohonenove nevronske mreže. Implementirali smo dvodimenzionalno (kar zajema tudi enodimenzionalno) mrežo pravokotne topologije. Vsak nevron je določen z dvodimenzionalnimi koordinatami. Vsak nevron, razen vektorja uteži, vsebuje tudi seznam vseh vhodnih vzorcev, v katerega se preslikajo, kar je potrebno pri paketnem načinu učenja, ocenjevanju topološke urejenosti in pri razdelitvi v gruče.

```
nastavi minimalno razdaljo na maksimalno vrednost
for vsaki nevron 'n' v mreži
      izračunaj razdaljo med nevronom in vzorcem
      if razdalja < minimalna razdalja
            minimalna razdalja <- razdalja
            zmagovalni nevron <- nevron 'n'
      end if
end for
```

**Slika 5.1** *Iskanje zmagovalnega nevrona je enako pri paketnem in iterativnem učenju*

```
for korak od 1 do maksimalnega števila korakov
      (razdeli vse vhodne vzorce po nevronih:)
      for vsaki vhodni vzorec x
            najdi zmagovalni nevron wn za vzorec x
            dodaj vzorec x v seznam vzorcev nevrona wn
      end for
      (posodobi vsak nevron s povprečno vrednostjo vzorcev, ki se preslikajo v njega in v
      nevrone znotraj njegovega sosedstva:)
      for vsaki nevron n mreže
            inicializira vsota na 0.0
            for vsaki nevron n2 mreže
                  if n2 v sosedstvu od n
                        doda seštevek vseh vzorcev nevrona n2 v
                        vsota
                  end if
            end for
            določi povprečje iz vsota
            nastavi uteži nevrona n na povprečje
      end for
      zmanjša velikost soseske
end for
```

**Slika 5.2** *Paketno učenje (podpoglavje 3.6) v vsakem koraku obravnava vse vzorce*

```
for s od 1 do maksimalnega število korakov
      izberi naključni vhodni vzorec
      najdi zmagovalni nevron 'wn'
      določi hitrost učenja na trenutnemu koraku s
      določi velikost in obliko sosednosti na koraku s
      for vsaki nevron 'n' v mreži
            if nevron 'n' se nahaja znotraj soseske
                  adaptiraj nevron 'n'
            end if
      end for
end for
```

**Slika 5.3** *Iterativno učenje (podpoglavje 3.3.4)*

Program za iterativno učenje mreže se zažene s parametri iz ukazne vrstice:

```
java IterativniSOM datoteka_vzorcev stolpci vrstice začetna_soseska
hitrost_učenja maks_korakov [prag_kvant_napake prag_top_urej]
```

Prvi parameter določa datoteko vhodnih vzorcev, naslednja dva sta število stolpcev in vrstic mreže. Sledita trije parametri učenja: začetna velikost soseske, začetna hitrost učenja in maksimalno število korakov izvajanja. Dva zadnja, neobvezna parametra, sta prag kvantizacijske napake in prag topološke urejenosti. S tema parametroma lahko ustavimo učenje, ko dobimo nevronsko mrežo ustrezne kvalitete.

Podobno zaženemo paketno učenje, ki ima en parameter manj (hitrost učenja):

```
java PaketniSOM datoteka_vzorcev stolpci vrstice začetna_soseska
maks_korakov [prag_kvant_napake prag_top_urej]
```

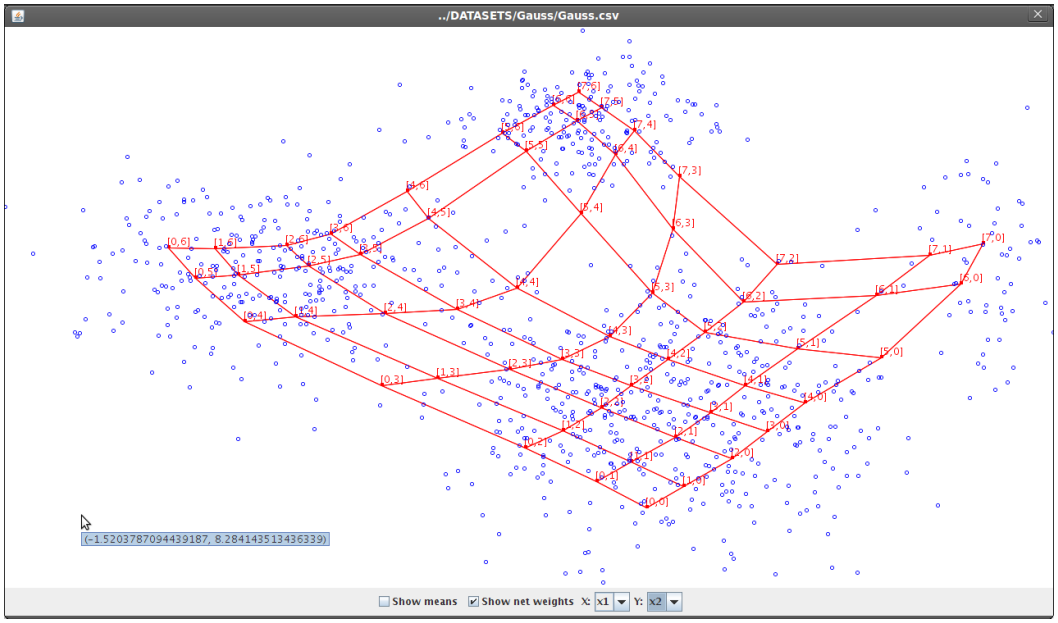
Naučena Kohonenova nevronska mreža je shranjena v izhodno datoteko *ISOM.dat* v primeru iterativnega algoritma ali v datoteko *BSOM.dat* v primeru paketnega algoritma.

## 5.2 Grafična ponazoritev

Naredili smo nekaj osnovnih grafičnih prikazov. Pri manjdimenzionalnih vzorcih je verjetno zanimivo pogledati dvodimenzionalni razpršeni graf (*scatter plot*), ki kaže vhodne vzorce skupaj z naučeno Kohonenovo nevronsko mrežo, če je podana (slika 5.4):

```
java Graf2D datoteka_vzorcev [datoteka_mreže]
```

Prvi argument progama je datoteka vhodnih podatkov, drugi argument je datoteka z naučeno nevronsko mrežo.

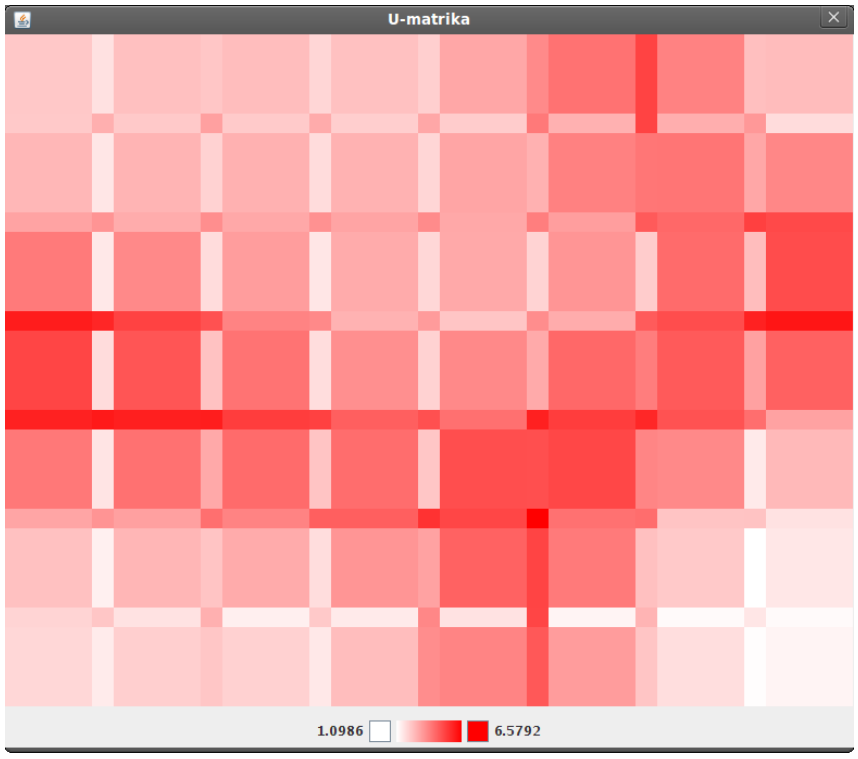



**Slika 5.4** *Razpršeni graf dvodimenzionalnih vhodnih podatkov (modro) skupaj z naučeno Kohonenovo nevronsko mrežo (rdeče). Možna je izbira katerihkoli dveh dimenzij, če jih je več.*

Kot je prej razloženo (podpoglavje 3.8), pri večdimenzionalnih podatkih taka predstavitev ne pomeni veliko. Zato raje uporabimo prikaz distančne matrike (slika 5.5), ki jo dobimo iz podane naučene Kohonenove mreže.

```
java UMatrika datoteka_mreže
```

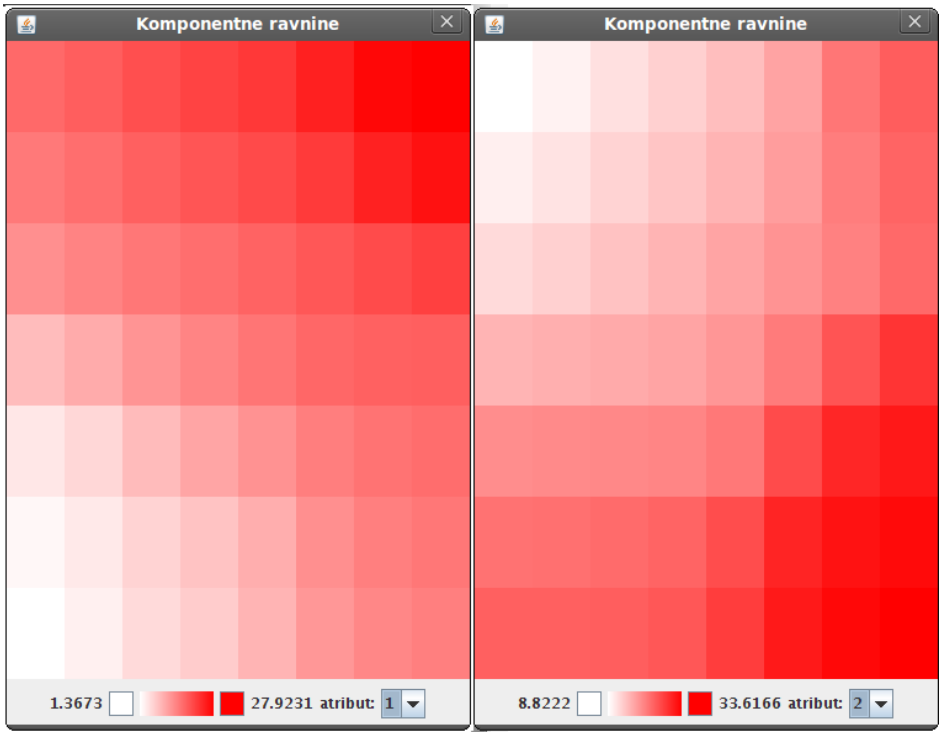
Ta program izračuna in prikaže distančno matriko (slika 5.5) podane dvodimenzionalne (lahko tudi enodimenzionalne) Kohonenove nevronske mreže. Ker je implementirana pravokotna topologija mreže, uporabljamo za prikaz pravokotnike namesto heksagonov. Pri računanju se iz dimenzij mreže ustvari nova večja matrika zaradi vmesnih polj. Prvo se izračunajo horizontalne in vertikalne razdalje, potem diagonalne in na koncu povprečne razdalje vsakega nevrona.




**Slika 5.5** *Distančna matrika (U-matrix) iste Kohonenove nevronske mreže, ki je ponazorjena na grafu v sliki 5.4. Odtenki barve ponazarjajo velikost razdalje. Večja polja predstavljajo povprečno razdaljo tega nevrona do ostalih sosednjih nevronov. Manjša kvadratna polja so povprečne razdalje med sosednjimi diagonalnimi pari nevronov. Manjši pravokotniki so razdalje med levimi in desnimi, oziroma med zgornjimi in spodnjimi nevroni. Tukaj so razvidne štiri gruče razdvojene z temnejšimi barvami.*

Podobno smo naredili prikaz komponentnih ravnin (podpoglavje 3.8). Pri prikazu komponentne ravnine za podano nevronsko mrežo se z odtenkom izbrane barve določi vrednost izbranega atributa v vsakem nevronu (slika 5.6).

```
java KompRavnina datoteka_mreže
```




**Slika 5.6** *Komponentne ravnine za isto mrežo kot na sliki 5.4. Odtenek barve, kot je prikazano v spodnji skali, ponazarja vrednost izbranega atributa v vsakem nevronu. V levem oknu je izbran prvi atribut, v desnem drugi.*

## 5.3 Razdelitev v gruče

Razdelitev v gruče je razložen v podpoglavju 4.1.2. Implementirali smo nenadzorovan način in nadzorovan način s klasificirano učno množico. Pri nadzorovanem načinu uporabljamo učno množico (običajno enako kot pri učenju mreže) v kateri vsaki vzorec ima določeni razred. Nevron razdelimo v gručo (razred) v katero sodi večina vzorcev, ki se v njega preslika:

```
java RazdeliSOM datoteka_mreže datoteka_klasificiranih_vzorcev –s
```

Pri nenadzorovanem načinu program podano Kohonenovo nevronsko mrežo razdeli v gruče in kot izhod da nevrone, ki imajo določene pripadnosti gručam:

```
java RazdeliSOM datoteka_mreže datoteka_vzorcev si|ce|co gran  gama
```

Program na vhodu dobi datoteko Kohonenove nevronske mreže, datoteko vhodnih vzorcev, parametre ki določajo granularnost razdelitve v gruče in način merjenja razdalj med in znotraj gruč *(single linkage, centroid linkage, complete linkage)*. Pri razdelitvi v gruče obravnavamo dendrogram, ki je binarno drevo v kateremu elementi ponazarjajo gruče (spajanja) z manjšimi podgručami kot otroki. Pri izgradnji dendrograma (slika 5.7) prvo iz nevronov, ki zadoščajo neenačbi (24) ustvarimo gruče, ki vsebujejo samo en nevron. Matrika razdalj je spodnje trikotna matrika, ki v elementu $D_{i,j;i<j}$ vsebuje razdaljo med *i*-tom in *j*-tom gručom. Ta matrika zmanjša potrebo po nepotrebnem računanju vseh razdalj v vsakem koraku. Posodobi se samo ustrezna vrstica in stolpec novonastale gruče v matriki. Gruče držimo v polju in pri vsakem koraku gremo skozi matriko distanc in iščemo najmanjšo razdaljo med obstoječimi gručami. Nova gruča v polju nadomesti svojega levega otroka (manjši indeks v polju). Desnega otroka (večji indeks v polju) označimo kot neobstoječega. Po ustvaritvi nove gruče, posodobimo ustrezni del matrike razdalj. Spajanje se konča, ko v polju ostane samo ena gruča, ki je koren dendrograma (slika 5.7). Kot je opisano v podpoglavju 4.1.2., po izgradnji dendrograma sledijo še tri faze. V drugi fazi poiščemo zanimive gruče (slika 5.8). Potem te gruče v tretji fazi (slika 5.9) združujemo, če ustrezajo pogoju. V zadnji fazi izvajamo končno združevanje (slika 5.10). V končnem združevanju gremo skozi seznam vseh gruč narejenih v prejšnjih fazah in prvo združujemo gruče, ki vsebujejo samo en nevron, če ustrezajo pogojem. Potem se spet sprehajamo skozi seznam gruč in združujemo vse gruče, ki ustrezajo pogojem ne glede na velikost. Končamo ko noben par gruč ni več možno združiti.

```
izbere nevrone v katere se preslika zadostno število vzorcev
inicializira polj gruč – vsaka gruča vsebuje samo en nevron
inicializira matriko razdalj D
while število gruč > 1
      (najde najmanjšo razdaljo v spodnje trikotni matriki razdalj:)
      for r od 1 do števila vrstic v D
            for c od 0 do r–1
                  if razdalja v D[c][r] najmanjša do sedaj
                        (zapomni gruče z najmanjšo razdaljo)
                        mr <– r
                        mc <– c
                  end if
            end for
      end for
      ustvari novo gručo z levim otrokom polje[mc] in desnim polje[mr]
      polje[mc] <– nova gruča
      polje[mr] <– ne obstaja več
      (posodobi matriko razdalj)
      for c od 0 do mc–1
            if gruča v polje[c] obstaja
                  D[c][mc]<–razdalja med gručami polje[c] in
                  polje[mc]
            end if
      end for
      for r od mc+1 do števila vrstic v matriki razdalj
            if gruča v polje[c] obstaja
                  matrika[mc][r]<–razdalja med gručami polje[r] in
                  polje[mc]
            end if
      end for
end while
koren dendrograma <– polje[mc]
```

**Slika 5.7** *Izgradnja dendrograma*

```
procedure najdiZanimiveZdružitve(koren)
      if koren ni list
            označiZanimiveGruče(koren.levo, 1)
            označiZanimiveGruče(koren.desno, 1)
      end if
end procedure

procedure označiZanimiveGruče(gruča, nivo)
      if gruča ni list
            prag <- (nivo+1) / (število izbranih nevronov - 1)
            maksNotranja <- maksRazdaljaZnotrajGruč(gruča.levo,
                              gruča.desno)
            koeficient <- maksNotranja /
                  razdaljaMedGručami(gruča.levo, gruča.desno)
            if koeficient > prag
                  označi gručo kot zanimivo
            else
                  označi gručo kot nezanimivo
            end if
            označiZanimiveGruče(gruča.levo, nivo+1)
            označiZanimiveGruče(gruča.desno, nivo+1)
      end if
end procedure
```

**Slika 5.8** *Iskanje zanimivih gruč (združevanj)*

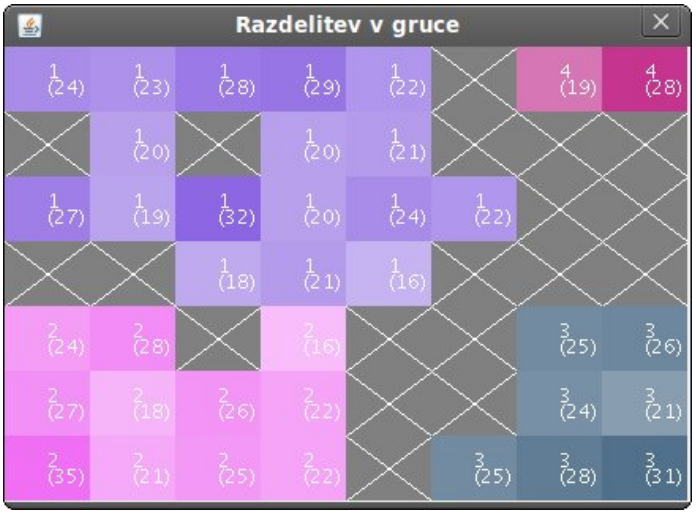
```
procedure iščiPotencialneGruče(gruča)
      if gruča ni list
            if gruča je označena kot zanimiva
                  številoVzorcevVNegručah <- preštejSvobodneVzorce(gruča)
                  številoVzorcevVGručah <- preštejVzorceVGručah(gruča)
                  if številoVzorcevVNegručah < številoVzorcevVGručah
                        dodaj gručo v seznam potencialnih gruč
                  else
                        iščiPotencialneGruče(gruča.levo)
                        iščiPotencialneGruče(gruča.desno)
                  end if
            end if

      else
            dodaj gručo v seznam potencialnih gruč
      end if
end procedure
```

**Slika 5.9** *Iskanje potencialnih gruč iz zanimivih spajanj (gruč)*

```
procedure končnoSpajanje()
      spajajSamoSingletone()
      while zgodiloSeJeSpajanje
            spajajKarkoli()
      end while
end procedure
procedure spajajSamoSingletone()
      for i od prve do predzadnje potencialne gruče
            for j od i+1 do zadnje potencialne gruče
                  zapomni i in j če so nevroni teh dveh 'singleton'
                  gruč najmočnejše povezani
            end for
            spoji gruče i in j v novo
      end for
end procedure
procedure spajajKarkoli()
      for i od prve do predzadnje potencialne gruče
            for j od i+1 do zadnje potencialne gruče
                  if obe gruče i in j nista singletoni
                        if notranjaRazdalja / razdaljaMedGručami > 1
                              spajaj gruče i in j
                        else
                              if dovlj močno povezane gruče
                                    spajaj gruče i in j
                              end if
                        end if
                  end if
                  zapomni i in j če so nevroni teh dveh 'singleton'
                  gruč najmočnejše povezani
            end for
            spoji gruče i in j v novo
      end for
end procedure
```

**Slika 5.10** *Končno spajanje potencialnih gruč*

Grafična ponazoritev razdelitve Kohonenove nevronske mreže v gruče obarva vsako gručo v drugačno barvo. Posebej označeni so nevroni mreže, ki niso upoštevani v razdelitvi v gruče.




**Slika 5.11** *Ponazoritev razdelitve v gruče Kohonenove nevronske mreže in podatkovne množice iz slike 5.4. Za preverjanje smiselnosti določenih gruč pomaga U-matrika v sliki 5.5. Razvidne so 4 gruče. Med njimi so prazni (interpolacijski) nevroni, ki sploh niso uporabljeni v razdelitvi v gruče. Zgornja številka v nevronu označuje gručo, spodnja številka v oklepajih označuje število vzorcev ki se preslikajo v ta nevron.*

## 5.4 Klasifikacija

Nevrone razdeljene v gruče uporabimo kot klasifikator (diskriminantne funkcije) (podpoglavje 4.2.5). Kot vhod v klasifikacijo, podamo naučeno Kohonenovo nevronsko mrežo, razdelitev v gruče in množico vzorcev katere želimo klasificirati.

```
java Klasificiraj datoteka_mreže datoteka_razdelitve datoteka_vzorcev
```

Uporabljamo algoritem 1-NN (1 najbližjih sosedov). Za vsaki vzorec iz podane datoteke se poišče njemu najbližji nevron in se mu dodeli razred (gruča) tega nevrona. Skozi klasifikacijo se naredi matrika zmedenosti s katero se, kot je opisano v podpoglavju 4.2.6, zlahka ocenjuje klasifikacija. Bolj podrobne meritve so prikazane v naslednjih primerih.

Matrika zmedenosti

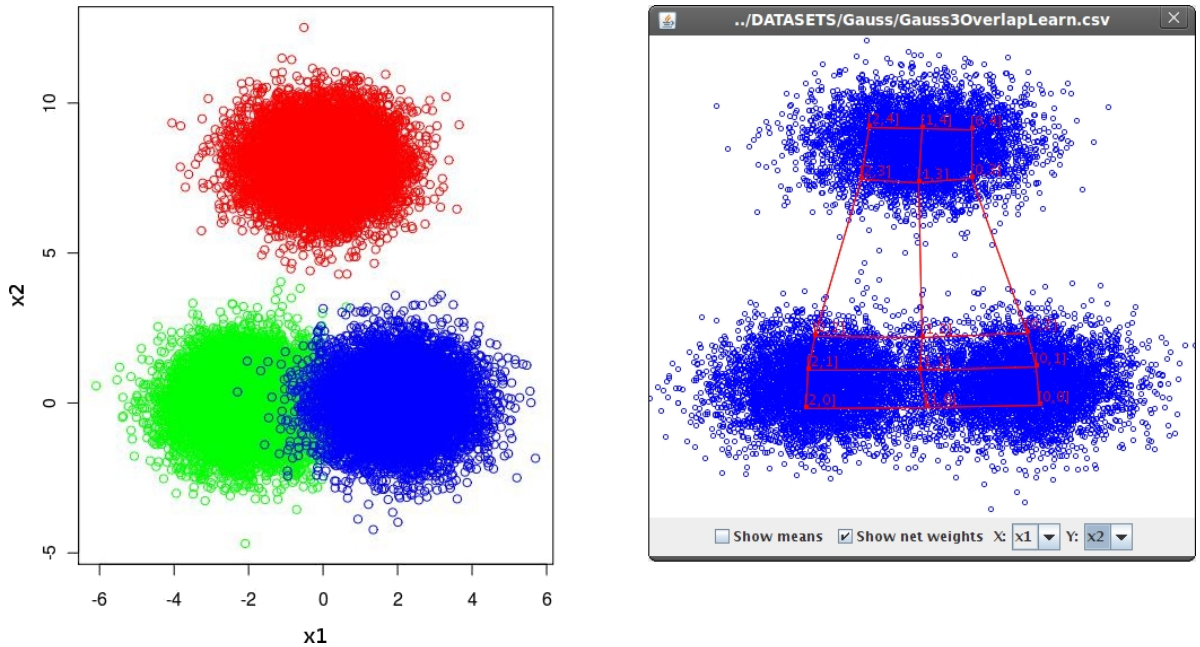
| Pravilno \ Napovedovano: | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 1 | 56 | 3 | 0 | |
| 2 | 4 | 62 | 5 | |
| 3 | 0 | 0 | 47 | |

napovedovalna natančnost p = 93.22%

**Slika 5.12** *Matrika zmedenosti (confussion matrix) za tri razreda. Stolpci ponazarjajo napovedane (klasificirane) razrede. Vrstice ponazarjajo pravilne razrede. 3 vzorca iz razreda 1 so napačno klasificirani v razred 2. Iz razreda 2 so 4 vzorca napačno klasificirana v razred 1 in 5 v razred 3. Vsi vzorci razreda 3 so pravilno klasificirani.*

## 5.5 Primer 1

Kot prvi primer smo generirali 30000 dvodimenzionalnih vzorcev, razdeljenih v tri normalno porazdeljene gruče (slika 5.13). Dve spodnji gruči se delno prekrivata. Za učenje Kohonenove nevronske mreže smo kot primer uporabili oba načina nenadzorovanega učenja




**Slika 5.13** *Levo je prikaz vhodnih vzorcev razdeljenih v tri gruče (dve spodnji gruči se delno prekrivata). Desno je na gručah ponazorjena naučena Kohonenova mreža s 15 nevronov (rdeče).*

(iterativni in paketni).

Pri obeh učenjih bomo ustvarili Kohonenovo nevronsko mrežo enakih dimenzij: 3 stolpcev in 5 vrstic. Iterativno učenje mreže dimenzij 3x5 s začetno velikostjo soseske 3 in začetno hitrostjo učenja 0.9 na podatkovni množici Gauss3.csv (Slika 5.13) smo zagnali kot:

```
java IterativniSOM Gauss3OverlapLearn.csv 3 5 3 0.9 500000
```

Spreminjanje parametrov skozi iterativno učenje je ponazorjeno v sliki 5.14. Učenje smo zagnali znova s podanim pragom maksimalne kvantizacijske napake (<=0.8) in minimalne topološke urejenosti (>=1.0):

```
java IterativniSOM Gauss3.csv 3 5 3 0.9 500000 0.8 1.0
```

Paketno učenje mreže enakih velikosti zaženemo z:

```
java PaketniSOM Gauss3.csv 3 5 3 500
```

Spreminjanje parametrov skozi paketno učenje je ponazorjeno v sliki 5.15. Učenje smo zagnali znova z namenom ustavljanja učenja v koraku ko imamo dovolj kvalitetno mrežo:

```
java PaketniSOM Gauss3.csv 3 5 3 500 0.8 1.0
```

Obe nastale (iterativno in paketno učenje) Kohonenove nevronske mreže so istih kvalitet.

Distančna matrika (*U-matrix*) dobljena z:

```
java UMatrika GaussSOM.dat
```

je ponazorjena v sliki 5.16.

Nastalo mrežo razdelimo na nadzorovan način *s:*

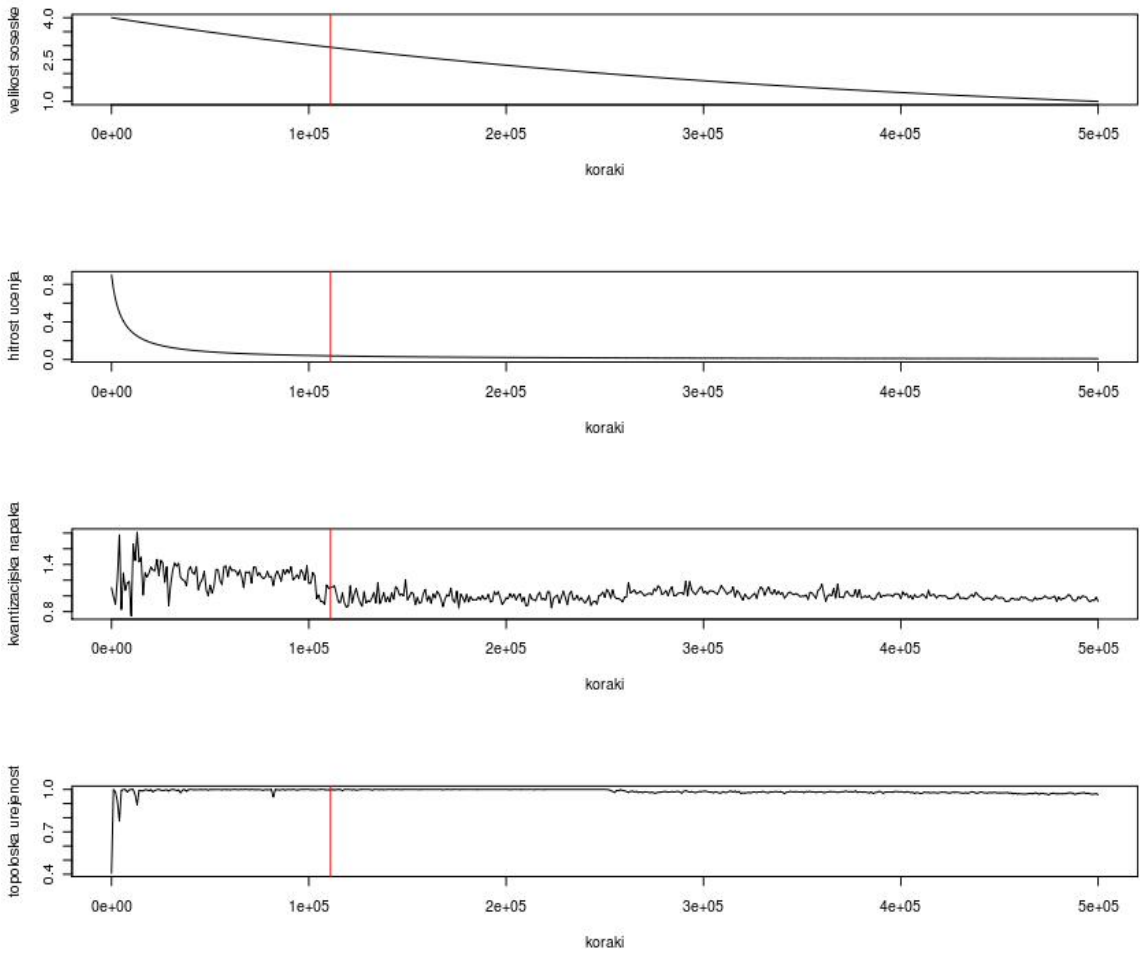
```
java RazdeliSOM GaussSOM.dat Gauss3OverlapLearnClass.csv –s
```

Razdelitev nevronov v tri gruče je ponazorjena v sliki 5.17.

S razdeljevanjem nevronov v gruče (slika 5.17) smo dobili klasifikator. Množica podatkov (slika 5.13) uporabljenih za učenje mreže je generirana umetno in zato zlahka z istimi parametri generiramo novo naključno testno množico s 30000 vzorci za ocenjevanje klasifikatorja.

Ocenjevanje klasifikacije (slika 5.18) zaženemo s podano naučeno Kohonenovo mrežo, razdelitvijo v gruče in testno množico klasificiranih podatkov:

```
java Klasificiraj Gauss3SOM.dat Gruce.dat Gauss3OverlapTestClas.csv
```

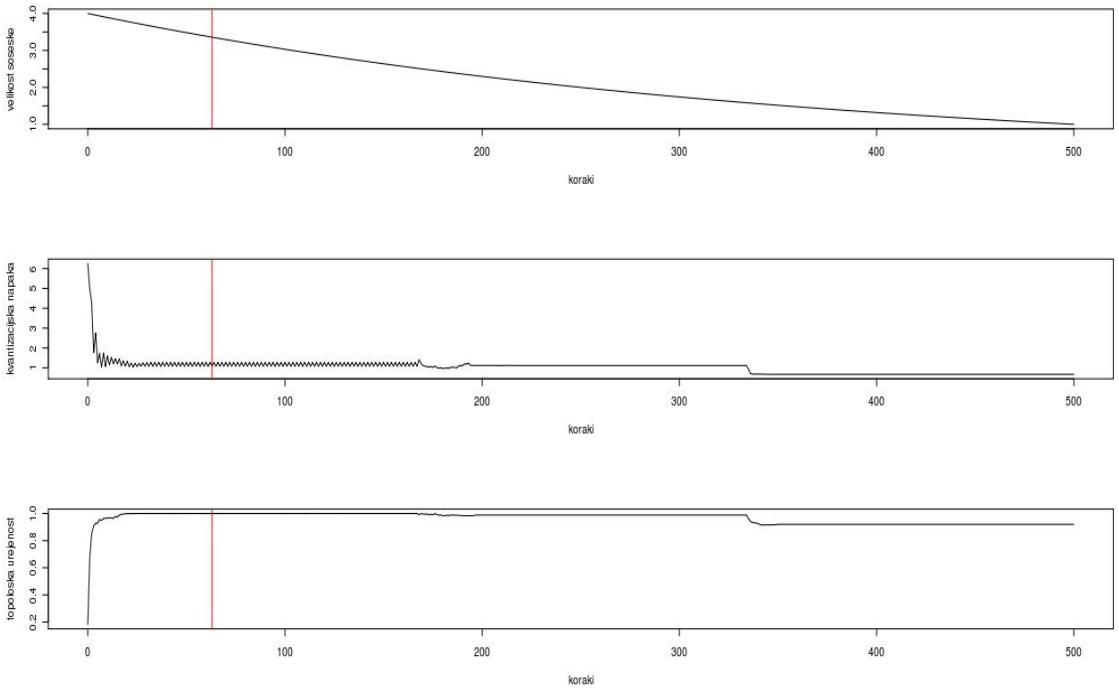



**Slika 5.14** *Potek spreminjanja parametrov učenja (hitrost učenja, velikost soseske) in kvalitete mreže (kvantizacijska napaka in topološka urejenost) pri iterativnem učenju za vhodne podatke na slki 5.13. Rdeča vertikalna črta ponazarja korak pri katerem smo končali učenje, ker je kvaliteta mreže zadoščala.*

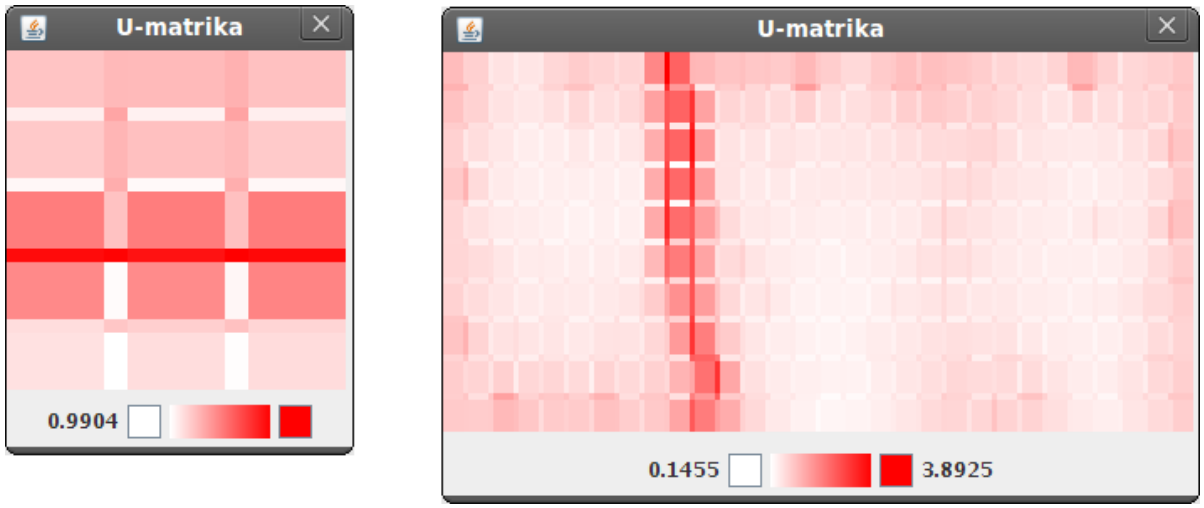
Dobljena klasifikacija manjše mreže s 15 nevronov ima natančnost 94.21%. S uporabo metode LVQ smo jo uspeli izboljšati na 94.53% (slika 5.18). Ker bi radi dobili bolj natančne meje med prekrivajočimi gručami, smo ustvarili večjo mrežo s 300 nevronov (30 x 10):

```
java PaketniSOM Gauss3.csv 30 10 17 500
```

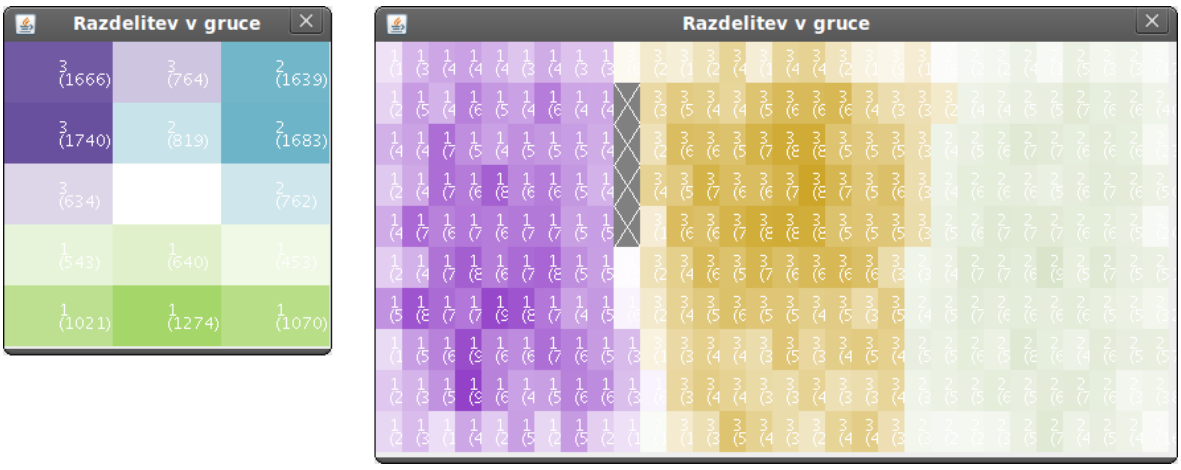
Dobljena klasifikacija je natančnejša s 98.41% (slika 5.18).



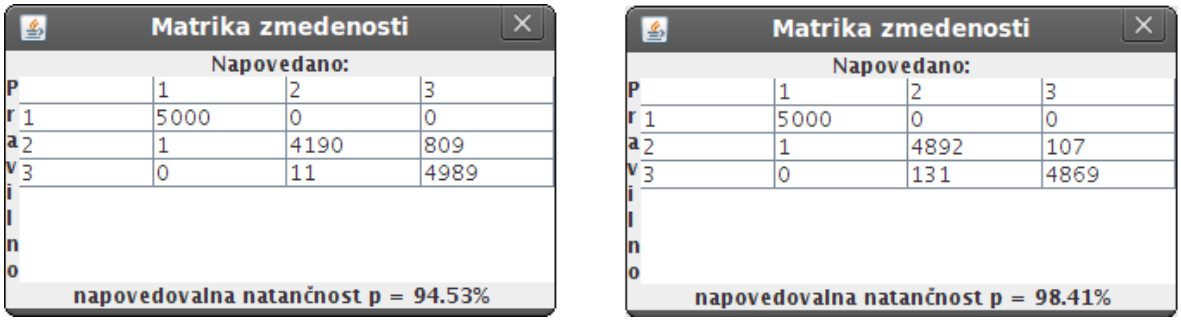

***Slika 5.15*** *Potek spreminjanja parametrov učenja (velikost soseske) in kvalitete mreže (kvantizacijska napaka in topološka urejenost) pri paketnem učenju za vhodne podatke na sliki 5.13.*




***Slika 5.16*** *Distanča matrika vhodnih podatkov iz slike 5.13.Levo je manjša mreža s 15 nevronov, desno je večja s 300 nevronov. Ker med dvemi prekrivajoči gručami ni meje, bi zelo težko bilo mrežo razdeliti na nenadzorovan način.*




***Slika 5.17*** *Nadzorovana razdelitev naučene Kohonenove nevronske mreže (vhodni podatki v sliki 5.13, U-matrika v sliki 5.16) v tri gruče. Levo je prikaz manjše mreže s 15 nevronov, desno je prikazana razdelitev v tri gruče večje mreže s 300 nevronov.*




***Slika 5.18*** *Matrike zmedenosti klasifikacije. Levo je matrika za mrežo s 15 nevronov, desno je za mrežo s 300 nevronov. Vsi vzorci prvega razreda, ki je krepko razdeljen od ostalih (slika 5.15) so pravilno klasificirani v obe mreži. V bistvu je samo en nevron dovolj za določanje meje te gruče. Pri drugih dveh gručah, ki se delno prekrivata je napaka očitna. V levi matriki je napaka očitno večja ker so v mreži s manj nevronov prekrivajoče meje med gručami nenatančno določene.*

## 5.6 Primer 2

V drugem primeru uporabljamo podatkovno množico *Wine* [UCI], ki vsebuje podatke kemične analize 13 lastnosti treh vrst vina. Dimenzij (atributov) je 13, razredov je 3 od katerih vsaki vsebuje približno enako članov (vin). Skupno število vzorcev je 177 (59 v 1. razredu, 71 v

2. razredu in 48 v 3. razredu). Ta podatkovna množica je primerna za prvo testiranje novega klasifikatorja. Ker določeni atributi imajo večji rang od ostalih, je zaželeno to podatkovno množico pred uporabo standardizirati. Na sliki 5.19 je možno primerjati razliko v poteku učenja med nestandardizirano in standardizirano (skalirano) množico.

Za učenje dvodimenzionalne mreže s 10 stolpcev in 8 vrstic uporabimo iterativno učenje s začetno velikostjo soseske in začetno hitrostjo učenja 6 in 0.9:

```
java IterativniSOM wine_skalirano.csv 10 8 6 0.9 500000
```

Čas izvajanja 500 000 korakov je 7.1 s. Dobimo topološko urejeno mrežo s kvantizacijsko napako 1.392.

Ko mrežo razdelimo s:

```
java RazdeliSOM WineSOM.dat wine_skalirano.csv co 6 1
```

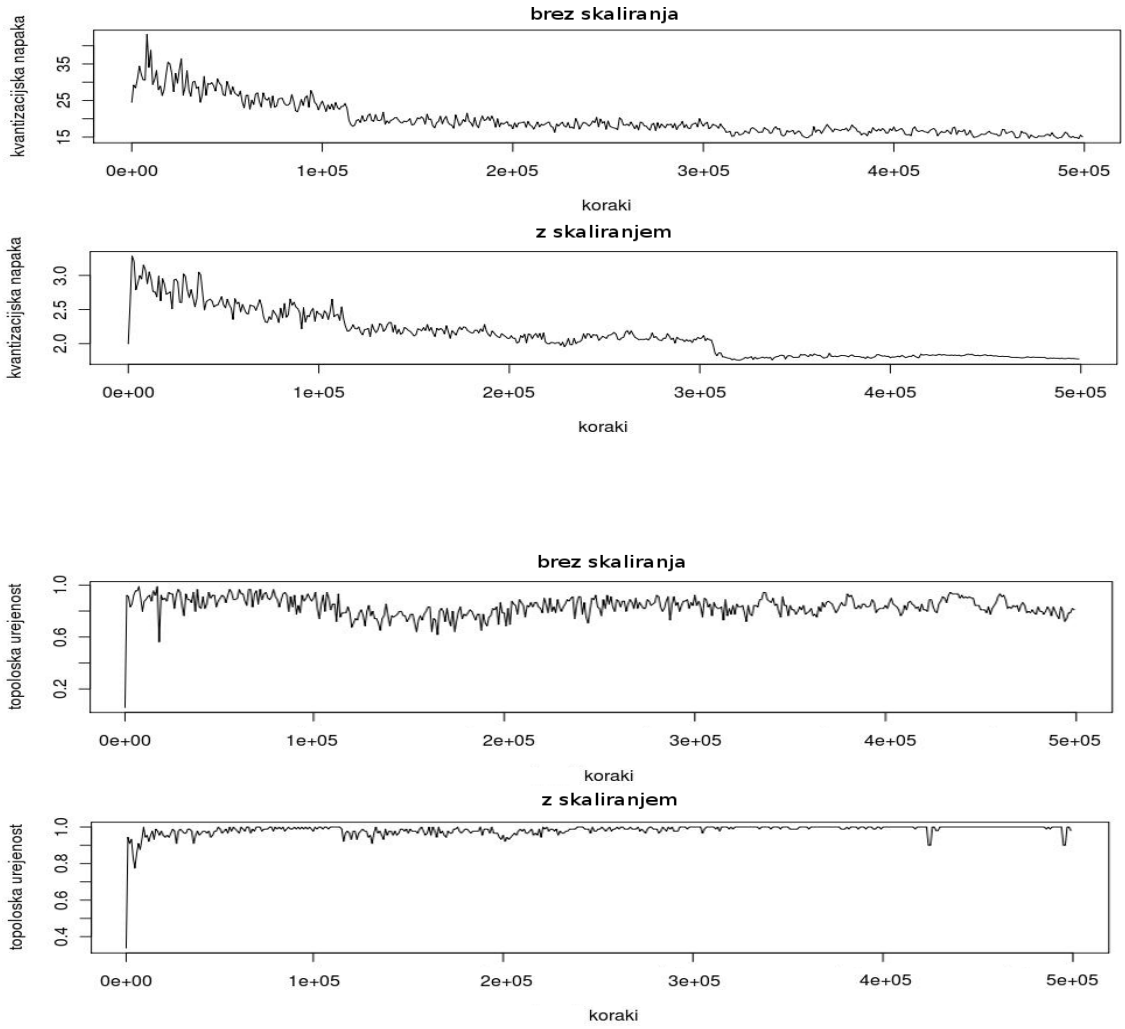
dobimo štiri gruče. Radi bi imeli tri gruče in zato en nevron ki je napačno razdeljen ručno dodelimo v najbližjo gručo in spremenimo številke gruč v številke ustreznih razredov (slika 5.21). Pomagamo si s distančno matriko (slika 5.20).

S 50% (sodi vzorci) skalirane podatkovne množice *Wine* smo naučili Kohonenovo nevronsko mrežo, drugih 50% (lihi vzorci) uporabimo za izvajanje (ocenitev) klasifikacije. Klasifikacija ni popolna, kot je razvidno iz matrike zmedenosti (slika 5.22). Na natančnost klasifikatorja vpliva več faktorjev. Kohonenova nevronska mreža, ki ima nevrone bolj v središčih Voronoievih celic (centrirana Voronieva teselacija) tvori natančnejši klasifikator. Pri izboljševanju nastale Voronoieve teselacije pomaga metoda LVQ. Razdelitev nevronov v gruče ima še večji vpliv na natančnost. Nevroni razdeljeni v napačno gručo sigurno narobe klasificirajo. Če algoritem razdeljevanja v gruče ne da zadostne rezultate, je možno *ručno* dodatno popraviti razdeljevanje s pomočjo distančne matrike ali uporabiti klasificirano učno množico podatkov za nadzorovano razdeljevanje nevronov v gruče.

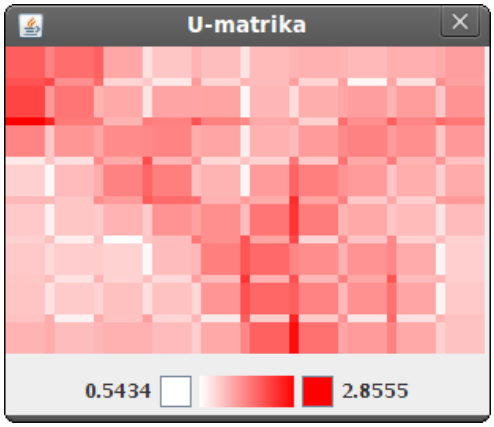
Klasifikacijo s razdeljeno mrežo (slika 5.21) izvajamo s testno množico klasificiranih podatkov:

```
java Klasificiraj WineSOM.dat Gruce.dat wine_test_klas.csv
```

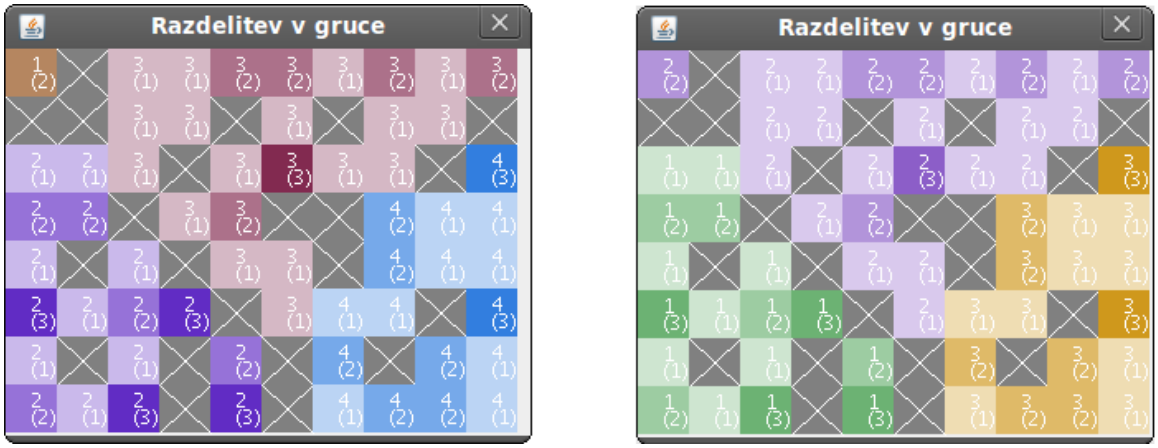
Meritev je ponazorjena v sliki 5.22, kjer je izmerjena napovedovalna natančnost od 91.01%




***Slika 5.19*** *Primerjava spreminjanja kvalitete mreže pri skalirani in neskalirani množici podatkov. Največja razlika je razvidna zgoraj pri kvantizacijski napaki.*




***Slika 5.20*** *Distančna matrika za Kohonenovo nevronsko mrežo podatkovne množice "Wine".*




***Slika 5.21*** *Nenadzorovana razdelitev v gruče. V levem oknu je prikazan rezultat nenadzorovane razdelitve v gruče, kjer je en nevron nezaželeno razdeljen v novo gručo (zgornji levi kot). V desnem oknu je "ručno" izboljšana razdelitev, kjer so gruče tudi preimenovane (pomagali smo si s distančno matriko za ugotavljanje mej in s matriko zmedenosti za določanje imen(številk) gručam)*


***Slika 5.22*** *Matrika zmedenosti za klasifikacijo s nenadzorovano razdeljeno mrežo.*

Klasifikacijo s nadzorovano razdeljeno mrežo izvajamo s testno množico klasificiranih podatkov:

```
java Klasificiraj WineSOM.dat GruceNadzorovane.dat wine_test_klas.csv
```

Meritve so ponazorjene v sliki 5.23, kjer je izmerjena napovedovalna natančnost od 94.38%

Kot je razloženo, natančnost klasifikacije (uteži nevronov) lahko izboljšamo s LVQ metodo, kateri podamo naučeno Kohonenovo mrežo, razdelitev v gruče, klasificirano testno množico

podatkov in jo zaženemo na 1000 korakov:

```
java LVQ WineSOM.dat Gruce.dat wine_test_klas.csv 1000
```

Izboljšava je razvidna na klasifikaciji s nenadzorovano razdeljeno mrežo, kateri je natančnost zrasla na 93.26% (slika 5.24).


***Slika 5.23*** *Matrika zmedenosti za klasifikacijo s mrežo razdeljeno na nadzorovani način.*

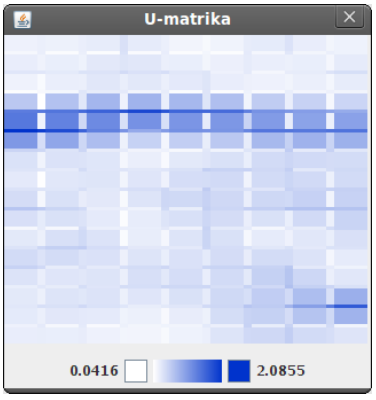

***Slika 5.24.*** *Izboljšava natančnosti klasifikacije po uporabi LVQ metode je razvidna pri nenadzorovano razdeljeni mreži (slika 5.21 in slika 5.22).*

## 5.7 Primer 3

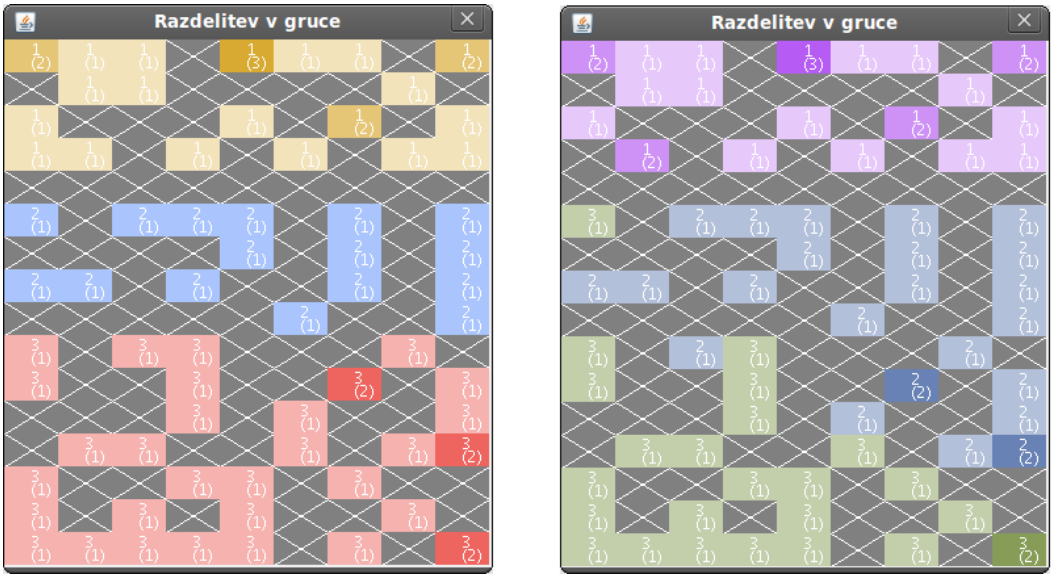
Kot tretji primer adaptivne klasifikacije s Kohonenovo nevronsko mrežo smo izbrali klasično podatkovno množico *Iris* [UCI]. To je verjetno najbolj znana podatkovna množica v področju razpoznavanja vzorcev (klasifikacije). Množica od 150 vzorcev je razdeljena v 3 razreda od katerih vsaki vsebuje 50 vzorcev (33.3%). Vsaki razred predstavlja en tip (*Setosa,*

*Virginica, Versicolor*) rastline *Iris*. En razred je linearno razdeljiv, ostala dva nista. Atributov (dimenzij) je 4 od katerih sta dva visoko korelirana.

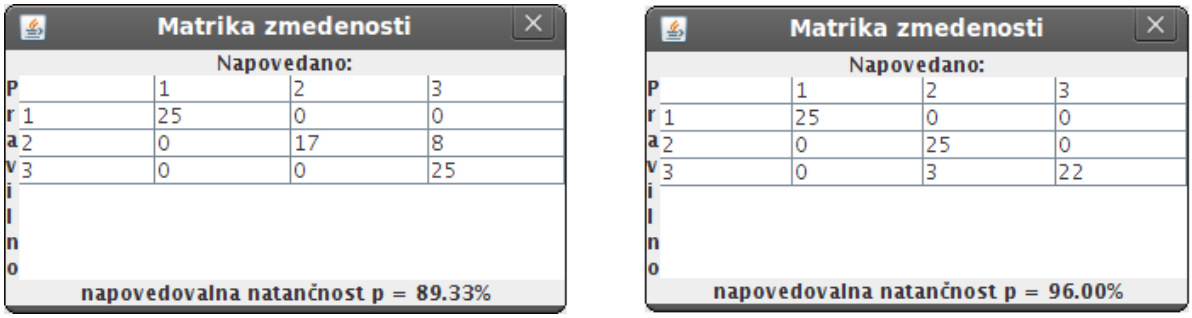
Z 50% podatkovne množice smo naučili Kohonenovo nevronsko mrežo dimenzij 9x16 (slika 5.25). Mrežo smo razdelili v gruče s nadzorovano in manj natančno nenadzorovano metodo (slika 5.26) in izvedli klasifikacijo (slika 5.27) s testno množico podatkov.




***Slika 5.25.*** *V distančni matriki je razviden en razred (zgoraj), ki je linearno razdeljiv od ostalih dveh, med katerimi je meja slabša.*




***Slika 5.26.*** *Dva načina razdeljevanja mreže. Levo je nenadzorovano razdeljevanje, kjer se gruče ne prekrivajo (spodnji dve). Desno je bolj natančno nadzorovano razdeljevanje, kjer se spodnji dve gruči prekrivata.*




***Slika 5.27.*** *Levo je matrika zmedenosti za nenadzorovano razdeljeno mrežo (slika 5.26 levo). Desno je za nadzorovano razdeljeno mrežo (slika 5.26 desno).*

# Poglavje 6

# Sklep

V tem diplomskem delu nam je cilj bilo uporabiti Kohonenovo nevronsko mrežo kot čim natančnejši klasifikator ob čim manjši interakciji uporabnika (nenadzorovano). Prvo smo prikazali idejo na kateri je zasnovana Kohonenova nevronska mreža (SOM), njeno strukturo, učenje in pripadajoče parametre. Potem smo se s Kohonenovo mrežo lotili razdeljevanja podatkov v gruče (razrede) in klasificiranja.

V poglavju 2 smo prikazali osnove in idejo na kateri je zasnovana Kohonenova nevronska mreža. V poglavju 3 je opisan način učenja mreže skupaj s parametri in grafičnimi prikazi. Omenjeni so tudi različni načini na katere gledamo Kohonenovo nevronsko mrežo - projekcija in vektorska kvantizacija. Poglavje 4 je obravnalo problematiko razdeljevanja mreže v gruče, klasifikacije testnih podatkov in merjenje natančnosti klasifikacije. Razdeljevanje mreže v gruče smo realizirali na nadzorovan (s klasificirano učno množico) in nenadzorovan način (aglomerativni hierarhični način s upoštevanjem topološke urejenosti mreže). Prikazali smo proces adaptivne klasifikacije, ki zajema Kohonenovo nevronsko mrežo s dodatkom metode LVQ, ki na nadzorovan način izboljšuje mrežo. Poglavje 5 smo začeli s opisom implementacije, prikazom psevdokode in narejenih grafičnih prikazov. Potem smo izbrali tri primera s katerimi smo ponazorili delovanje narejene demonstracijske programske opreme. V primerih smo primerjali iterativno in paketno učenje. Pokazali smo prednost (manjša kvantizacijska napaka in večja topološka urejenost) skaliranja vhodnih podatkov, ko je to potrebno. V primerjavi nadzorovanega in nenadzorovanega razdeljevanja mreže v gruče smo pokazali da je nadzorovani, kot je pričakovano, veliko bolj natančen ko gruče niso krepko razdeljene. Na koncu vsakega primera v poglavju 5 smo izmerili natančnost klasifikatorja in pokazali možne izboljšave

dobljene s metodo LVQ.

Kohonenova nevronska mreža (ali algoritem SOM) je eden od najpopularnejših nenadzorovanih načinov učenja. Sama Kohonenova nevronska mreža je bolj namenjena grafični ponazoritvi večdimenzionalnih podatkov, kot klasificiranju. Že sam grafični prikaz distančne matrike ponuja kot projekcija veliko pomoč pri razumevanju strukture podatkov (gruče).

Na primerih v poglavju 5 je razvidno da je naučena Kohonenova nevronska mreža uporabna kot klasifikator. Metoda LVQ (uporabljali smo samo algoritem LVQ-1) občasno malo izboljša njegovo natančnost.

V diplomski nalogi smo pokazali nekaj dejstev o Kohonenovi nevronski mreži in njeni uporabi v klasifikaciji:

- Mreža veliko hitreje in zanesljivejše konvergira v urejeno stanje ko se nevroni inicializirajo v že urejeno obliko (linearna inicializacija s PCA metodo).
- Paketni in iterativni način učenja ustvarita enako kvaliteto mreže. Paketni je lažji za nastavitev in večinoma hitrejši.
- Učenje ne deluje za manjkajoče ali nenumerične podatke. Potrebno jih je dopolniti ali nadomestiti s številkami.
- Skaliranje vhodnih podatkov je nujno ko določeni atributi (dimenzije) imajo veliko večje variance od ostalih.
- Nadzorovano razdeljevanje mreže v gruče je natančnejše kot nenadzorvano. Če so gruče krepko razdeljene ni razlike.
- S večjo nevronsko mrežo (manjša kvantizacijska napaka) dobimo natančnejši klasifikator v primerih ko se gruče (razredi) med sabo prekrivajo. Ko so gruče krepko razdeljene število nevronov nima vpliva na natančnost.
- Za večino podatkovnih množic obstajajo natančnejši klasifikatorji kot je Kohonenova nevronska mreža. Prednost ji je nelinearnost, ki je posebej razvidna ko se razredi (gruče) prekrivajo. Kohonenovo mrežo raje uporabljamo kot projekcijo ki grafično ponazori strukturo neznanih podatkov.

# Poglavje 7

# Literatura

**[Kohonen, 2001]** Kohonen, Teuvo. *Self-Organizing Maps*. Tretja izdaja: Springer, 2001.

**[Haykin, 1999]** Simon Haykin. *Neural Networks – A comprehensive foundation.* Druga izdaja: Pearson Prentice Hall, 1999.

**[Rojas, 1996]** Raul Rojas. *Neural Networks. A Systematic Introduction.* Prva izdaja: Springer, 1996.

**[Duda et al., 2004]** Duda, R. O., Hart, P. E., Stork, D. G. *Pattern classification.* Druga izdaja: Wiley, 2004.

**[Fort et al., 2002]** Fort, J.C., Letremy P., Cottrell M., „Advantages and drawbacks of the Batch Kohonen algorithm“. v ESANN'2002, str. 223-230, 2002

**[Kiviluoto, 1996]** Kiviluoto, K. „Topology preservation in Self-Organizing Maps“. *v ICNN,* str. 294-299, 1996

**[Vesanto in Alhoniemi, 2000]** Vesanto J., Alhoniemi E., „Clustering of the Self-Organizing Map“. *v* IEEE Transactions on Neural Networks, May 2000

**[Taşdemir in Merényi, 2005]** Taşdemir K., Merényi E., "Considering Topology in the Clustering of Self-organizing maps", v *WSOM 2005*

**[SOMToolbox, 2000]** Vesanto J., Himberg J., Alhoniemi E., Parhankangas J. *SOM Toolbox for Matlab 5.*Helsinki University Of Technology, April 2000.

**[UCI]** http://archive.ics.uci.edu/ml/

**[NormalizationFAQ]** http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-16.html