1 UMETNE NEVRONSKE MREŽE

1.1 Kaj je umetna nevronska mreža?

Umetna nevronska mreža (*artificial neural network*)**,** pogosto imenovana kar nevronska mreža, je matematični oz. računski model, ki temelji na delovanju bioloških nevronskih mrež. Sestavljena je iz množice medsebojno povezanih preprostih procesnih elementov, **umetnih nevronov** (*artificial neuron*)**,** katerih utežene povezave določajo obnašanje mreže.
Princip delovanja je enak, kot naj bi ga uporabljali človeški možgani, ki imajo sposobnost organiziranja nevronov, tako da izvršujejo določene naloge (npr. razpoznavanje vzorcev, zaznavanja, nadzor gibanja). Možgani tako delo opravijo v desetinki sekunde, medtem ko bi računalnik potreboval še za dosti lažje probleme več dni.
Značilnost nevronskega procesiranja je v tem, da upošteva nakopičeno znanje, pridobljeno v fazi učenja, ter da odgovarja na vhodne dogodke na način, ki je najbližji glede na izkušnje v učni dobi. To pomeni, da je za procesiranje mreže značilno posploševanje in tolerantnost napak na vhodu, kar je vsekakor lastnost, zaradi katere jih uvrščamo med inteligentne procesorske sisteme.
Teoretični začetki nevronskih mrež sodijo v leto 1943. Takrat sta McCulloch in Pitts postavila matematični model živčne celice oziroma nevrona. Njun model, ki se uporablja še danes, predstavlja element, v katerega vodijo uteženi vhodi, izhod elementa pa je odvisen od vrednosti vsote produktov vhodnih spremenljivk z ustreznimi utežmi. Če je vrednost vsote nad neko posebno vrednostjo oz. **pragom** (*threshold*), potem ima izhod eno vrednost, v nasprotnem primeru pa drugo.

1.2 Umetni nevron

Umetni nevron je model, ki je zasnovan kot približek biološkemu nevronu. Dendrite pri biološkem nevronu nadomestijo vhodi (x_i), ki imajo vsak svojo utež (w_i). Če imamo v obravnavi več nevronov, potem se uteži povezav nevrona k zapišejo kot w_{ki}. Izhod nevrona k, ki predstavlja vsoto vhodov, pomnoženih z ustreznimi utežmi, imenujemo tudi **aktivacija** (u_k). Aktivacija nevrona je vhod v **aktivacijsko funkcijo** (*activation function*)**,** včasih imenovano tudi **funkcija stiskanja**, ki omeji amplitudo izhoda nevrona. Vrednosti izhoda tipično pripadajo intervalu [0, 1]ali intervalu [−1, 1]. Izhod nevrona k označimo z y_k.
Model nevrona k vsebuje tudi zunanji parameter, **prag** (θ_k), ki predstavlja nivo signala, pri katerem se sproži nevron. Nevron opišemo z naslednjim parom enačb.

$$u_k = \sum_{j=1}^{s} w_{kj} x_j$$

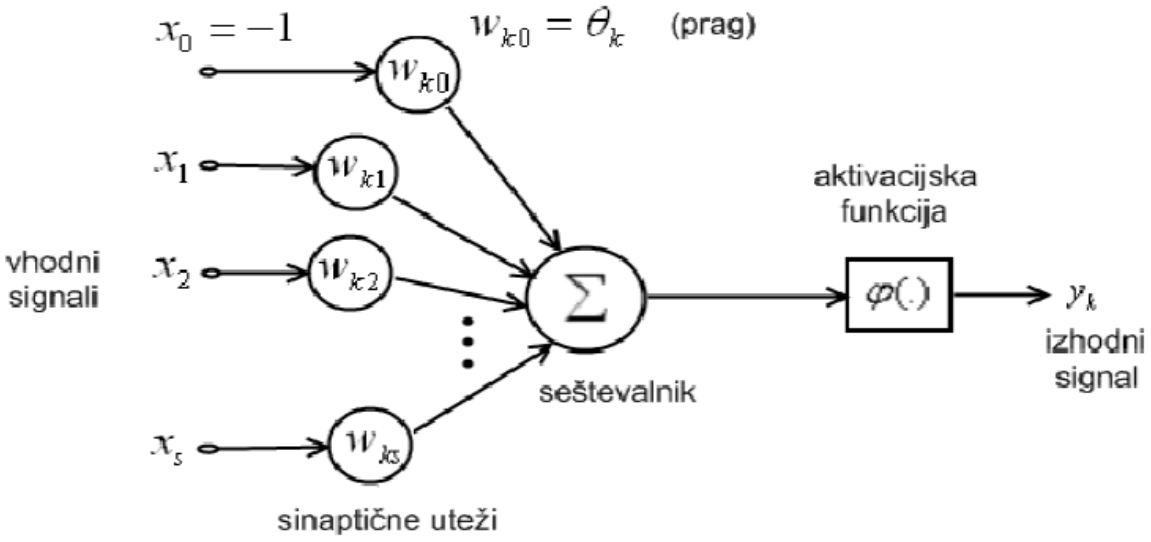
$$y_k = \varphi(u_k - \theta_k),$$

pri cemer so $x_1, x_2, \ldots, x_s$ vhodni signali, $w_{k1}, w_{k2}, \ldots, w_{ks}$ sinaptične uteži nevrona, u_k je aktivacija, θ_k je prag, φ je aktivacijska funkcija in y_k je izhodni signal nevrona. Če vpeljemo $w_{k0} = \theta_k$ in $x_0 = -1$ ter zapišemo $v_k = u_k - \theta_k$, lahko preoblikujemo zgornji enačbi kot:

$$v_k = \sum_{j=0}^{s} w_{kj} x_j$$

$$y_k = \varphi(v_k).$$

Model umetnega nevrona vidimo na sliki 1.1.



Slika 1.1: Model umetnega nevrona

Tipi aktivacijskih funkcij

Obstajajo naslednji tipi aktivacijske funkcije:
1. **Pragovna (stopničasta) funkcija** (*threshold function*):

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases}$$

Tedaj je izhod iz nevrona (izpustimo indeks *k*):

$$y = \varphi(v) = \varphi(u - \theta) = \begin{cases} 1, & u \geq \theta \quad oziroma \quad v \geq 0 \\ 0, & u < \theta \quad oziroma \quad v < 0 \end{cases}$$

Nevronu s pragovno aktivacijsko funkcijo rečemo **McCulloch-Pittsov model**, mnogi avtorji pa tej obliki umetnega nevrona s to aktivacijsko funkcijo pravijo **pragovna logična enota** (*threshold logic unit – TLU*). Uporablja se za klasifikacijo linearno ločljivih vzorcev.

2. **Odsekoma linearna funkcija:**

$$\varphi(v) = \begin{cases} 1, & v \geq 1/2 \\ v + 1/2, & -1/2 < v < 1/2 \\ 0, & v \leq -1/2 \end{cases}$$

3. **Sigmoidna funkcija** je daleč najbolj običajna oblika aktivacijske funkcije pri nevronskih mrežah. Omenimo tri tipe sigmoidne funkcije:

a) **Logistična funkcija** (*logistic function*):

$$\varphi(v) = \sigma(v) = \frac{1}{1 + e^{-av}},$$

kjer je *a* **parameter naklona** (*slope parameter*). S spreminjanjem tega parametra dobimo sigmoidne funkcije z različnimi nakloni.

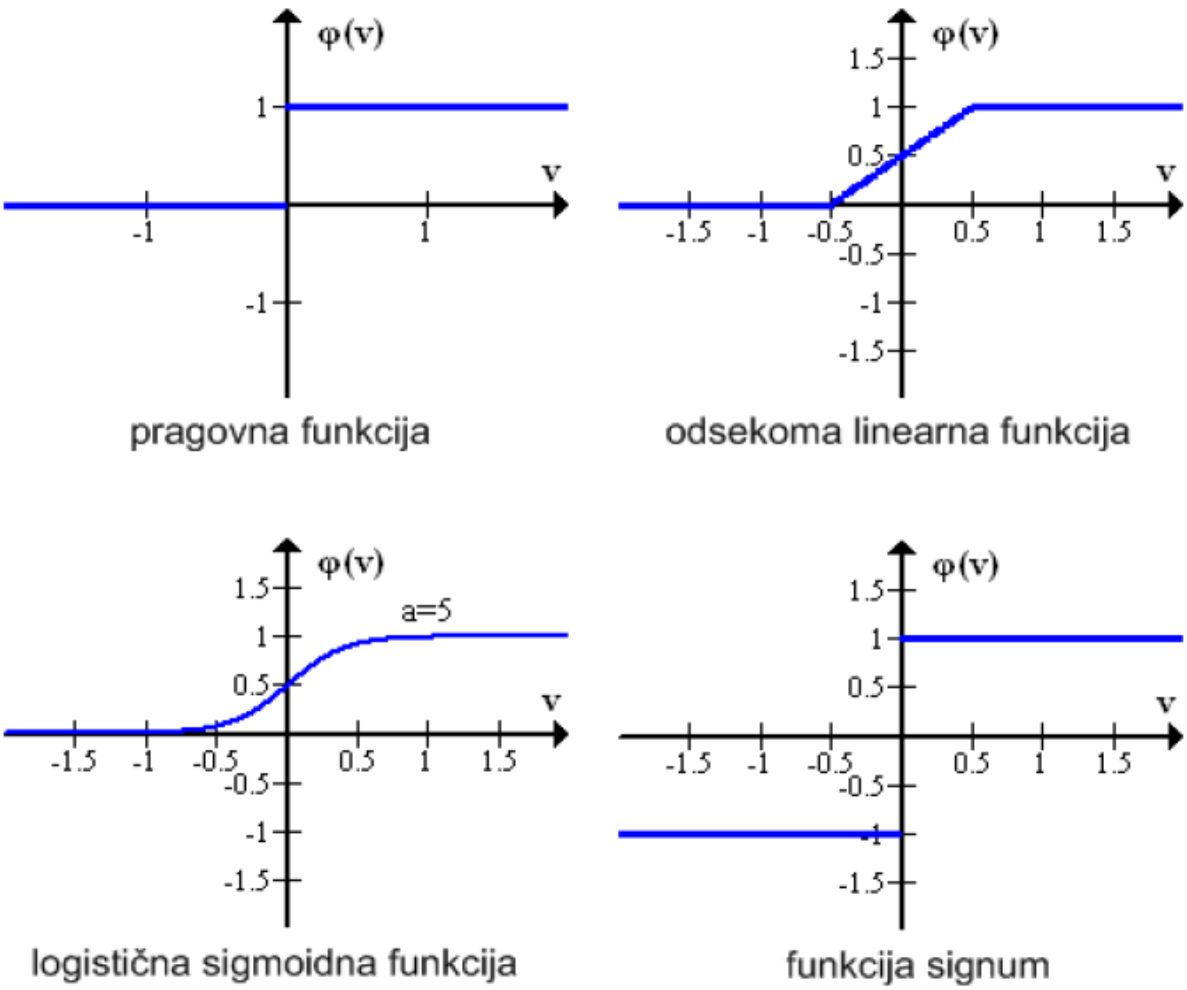
b) **Funkcija signum:**

$$\varphi(v) = \operatorname{sgn}(v) = \begin{cases} 1, & v > 0 \\ 0, & v = 0 \\ -1, & v < 0 \end{cases}$$

c) **Funkcija hiperbolicni tangens:**

$$\varphi(v) = \tanh\frac{v}{2} = \frac{1-e^{-v}}{1+e^{-v}}$$

Oblike funkcij vidimo na sliki 1.2.



Slika 1.2: Oblike aktivacijskih funkcij

1.3 Proces učenja umetne nevronske mreže

Ena najpomembnejših lastnosti nevronske mreže je zmožnost učenja in izboljševanja svoje učinkovitosti skozi proces učenja. V grobem bi lahko rekli, da ta proces nastavi nevronsko mrežo tako, da na določeno množico vhodov reagira z določeno množico izhodov.
Učenje poteka kot iterativni proces prilagajanja, ki se izvaja na utežeh in pragovih glede na določeno učno pravilo. V povezavi z nevronskimi mrežami definiramo učenje na naslednji način:

Učenje je proces, pri katerem se prosti parametri nevronske mreže prilagodijo skozi nenehen proces vzpodbude iz okolja, v katerega je mreža vložena. Tip učenja določa način, po katerem se spreminjajo parametri.

Ta definicija vključuje naslednje zaporedje dogodkov:
1. Nevronsko mrežo vzpodbuja okolica.
2. Nevronska mreža se spremeni zaradi te vzpodbude.
3. Nevronska mreža odgovori na nek način okolju zaradi sprememb, ki se zgodijo v njeni notranji zgradbi.

Naj bo $w_{kj}(n)$ vrednost sinaptične uteži w_{kj} v casu n. Prilagodimo sinaptično utež $w_{kj}(n)$ z $\Delta w_{kj}(n)$ tako, da dobimo osveženo vrednost $w_{kj}(n+1)$:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

Prilagoditev $\Delta w_{kj}(n)$ se izračuna kot rezultat vzpodbude okolice (dogodek 1). Osvežena vrednost $w_{kj}(n+1)$ predstavlja spremembo v mreži kot rezultat te vzpodbude (dogodek 2). Dogodek 3 se izvrši, ko se izračuna odgovor nove mreže, ki deluje z osveženo vrednostjo parametra $w_{kj}(n+1)$.
Opisano množico pravil imenujemo **algoritem učenja** (*learning algorithm*). Obstaja več algoritmov za učenje nevronskih mrež, ki se v osnovi razlikujejo po načinu prilagoditve sinaptične uteži w_{kj}.
Ločimo naslednje vrste algoritmov učenja:

- učenje s popravljanjem napake (*error-correction* learning),
- Hebbovo učenje (*Hebbian learning*),
- tekmovalno učenje (*competitive learning*) in
- Boltzmannovo učenje (*Boltzmann learning*).

Poleg algoritmov učenja poznamo še **paradigme učenja** (*learning paradigms*), ki se nanašajo na modele okolja, v katerem delujejo nevronske mreže. Ločimo tri osnovne paradigme učenja:

- nadzorovano ali vodeno ucenje (*supervised learning*),
- okrepitveno ucenje (*reinforcement learning*) in
- samoorganizirajoce (*self-organizing*) ali nenadzorovano ucenje (*unsupervised learning*).

2 HOPFIELDOVA NEVRONSKA MREŽA

Na Hopfieldovo nevronsko mrežo lahko gledamo tudi kot na nelinearni **asociativni pomnilnik** (*associative memory, content addresable memory*), katerega primarna naloga je priklicati vzorec, ki je bil shranjen v pomnilniku, na podlagi nepopolne oziroma delno popačene predstavitve tega vzorca.
Hopfieldova nevronska mreža je tudi najpopularnejša oblika nevronske mreže, ki je sposobna izvajati asociativno povezovanje vzorcev. Bistvo Hopfieldovih mrež je, da si lahko zapomnijo (hranijo) množico učnih vzorcev kot množico **stabilnih stanj**, imenovanih tudi **atraktorji** (*attractors*). Če damo na vhode mreže testni vzorec, zna mreža konvergirati k enemu od množice učnih vzorcev oziroma enemu od stabilnih stanj, ki je najbližje testnemu vzorcu.
Želimo si, da mreža poveže pokvarjeno sliko z njeno originalno, neokrnjeno različico. Podobno deluje tudi človeški spomin, ki lahko popači videno ali slišano, vendar je z nekaj truda sposoben priklicati nazaj manjkajoče podrobnosti.
Gradniki Hopfieldove mreže so nevroni tipa TLU (McCulloch-Pitts), ki so medsebojno povezani tako, da je izhod vsakega nevrona povezan na vhode vseh ostalih nevronov v mreži, kar pomeni, da ima vsak posamezni nevron vpliv na aktivacijo vseh ostalih nevronov. Delež tega vpliva je omejen s pomočjo uteži, ki so dodeljene vsakemu vhodu v nevron, izračunane pa so v fazi učenja.
Nevron i ima izhod y_i. Če je nevron aktiven, ima izhod +1, če ni aktiven, ima stanje -1. Za mrežo z N nevroni je stanje mreže definirano z vektorjem $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$
Velja:

$$y_j = \begin{cases} +1, & \check{c}e \quad v_j > 0 \\ -1, & \check{c}e \quad v_j < 0 \end{cases}$$

Če velja = 0 v_j, potem y_j ohrani prejšnje stanje, ne glede na to, ali je bil nevron j aktiven ali ne. Ta funkcija je funkcija signum in jo lahko zapišemo tudi kot:

$$y_j = \mathrm{sgn}[v_j]$$

Oznaka v_j pomeni aktivacijo nevrona j, ki je dana z enačbo:

$$v_j = \sum_{i=1}^{N} w_{ji} y_i - \theta_j$$

θ_j je konstantni prag nevrona j, določen od zunaj.

2.1 Algoritem Hopfieldove mreže

Algoritem deluje v dveh fazah, in sicer v fazi učenja ter fazi iskanja.

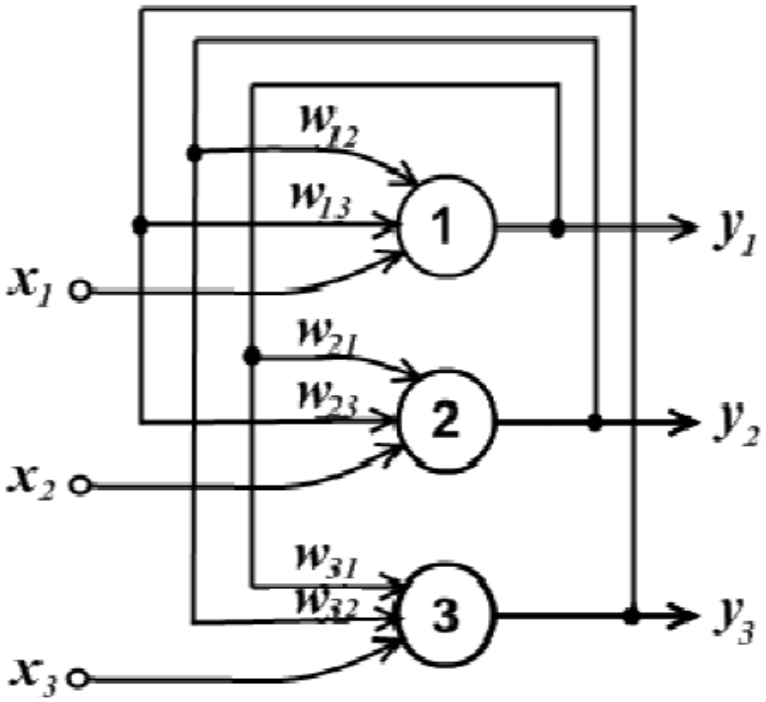
1. Faza ucenja

Recimo, da želimo shraniti množico N-dimenzionalnih vektorjev (binarnih besed), ki jih označimo s ξ_μ , pri čemer je $\mu = 1,\ldots, p$. Elementi vektorja ξ_μ so ±1. Ti vektorji predstavljajo učne vzorce in s tem osnovni spomin mreže.

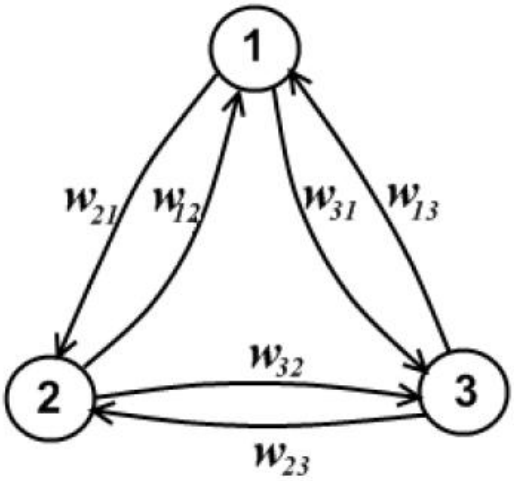
Hopfieldovo mrežo učimo z nastavljanjem uteži povezavam med nevroni. Za razliko od algoritma vzvratnega razširjanja tu vrednosti utežem nastavimo samo enkrat. Uporabimo posplošeno Hebbovo učenje, s katerim so sinaptične uteži iz nevrona *i* k nevronu *j* definirane z naslednjo enačbo:

$$w_{ji} = \begin{cases} \dfrac{1}{N}\sum_{\mu=1}^{p} \xi_{\mu,j}\xi_{\mu,i}, & j \neq i \\ 0, & j = i, \quad i, j = 1, 2, \ldots, N \end{cases}$$

Razlog za uporabo $1/N$ kot konstante proporcionalnosti je v poenostavitvi matematičnega opisa pridobivanja informacij.
Naj **W** označuje matriko sinaptičnih uteži dimenzije N x N , kjer je element matrike v *j*-ti vrstici ter *i*-tem stolpcu w_{ji}. Matrika uteži je simetricna ($w_{ij} = w_{ji}$) . Sklepamo, da je izhod vsakega nevrona povezan na vhode vseh ostalih nevronov. Ker velja $w_{ii} = 0$, povratne povezave na isti nevron ni.
Na sliki 2.1 je prikazana Hopfieldova mreža za N=3. Vektor **x** simbolizira od zunaj vsiljeno stanje **y.** Pogosto pa rišemo Hopfieldovo mrežo v poenostavljeni obliki, ki je prikazana na sliki 2.2.



Slika 2.1: Hopfieldova mreža za N=3



Slika 2.2: Poenostavljena predstavitev Hopfieldove mreže za N=3

2. Faza iskanja

V tej fazi postavimo na vhod mreže, ki je v bistvu identičen izhodu, poljuben N-dimenzionalni poskusni vektor **x**, ki ima elemente ±1. Tipično predstavlja nepopolno ali šumno verzijo osnovnega spomina mreže. Vektorju **x** poskušamo poiskati podoben učni vzorec, ki je že shranjen v Hopfieldovi mreži kot stabilno stanje.

Najprej postavimo **y**(0) = **x**, nadaljnja stanja pa računamo po naslednji formuli

$$y_j(n+1) = \operatorname{sgn}\left[\sum_{i=1}^{N} w_{ji} y_i(n) - \theta_j\right], \quad j = 1,\ldots,N$$

$$\mathbf{y}(n+1) = \operatorname{sgn}\left[\mathbf{W}\mathbf{y}(n) - \Theta\right]$$

Če je y_j večji kot 0, nevron j preklopi v stanje +1 ali ostane v tem stanju, če je že tam. Če je y_j manjši kot 0, nevron j preklopi v stanje −1 ali ostane v tem stanju, če je že tam. Če je y_j enak 0, nevron j ostane v prejšnjem stanju, ne glede na to, kakšno je to stanje bilo.

Ločimo dva različna modela: **Hopfieldov model** in **Littleov model.** Hopfieldov model uporablja **asinhrono** (serijsko) dinamiko, kjer se nevroni prožijo en za drugim, vrstni red proženja pa je naključen. Littleov model uporablja **sinhrono** (vzporedno) dinamiko, kjer se vsi nevroni prožijo hkrati. Hopfieldova mreža vedno konvergira k stabilnemu stanju, medtem ko Littleova mreža konvergira k stabilnemu stanju ali k ciklu z dolžino dva.

Za oceno napredovanja uporabimo **energijsko funkcijo,** ki je opisana v naslednjem razdelku. Proženje nevronov izvajamo tako dolgo, dokler se stanje mreže ne spreminja več. Tedaj najdemo stabilno stanje **y** oziroma rešitev. Za stabilno stanje velja formula:

$$\mathbf{y} = \operatorname{sgn}\left[\mathbf{W}\mathbf{y} - \Theta\right]$$

kjer je Θ vektor praga, ki ga nastavljamo od zunaj.

Če Hopfieldovo mrežo uporabimo za shranjevanje K vzorcev s pomočjo Hebbovega predpisa za uteži, ima mreža običajno več **lažnih atraktorjev** (*spurious attractors*), ki jim pravimo tudi **lažna stanja** (*spurious states*). Lažna stanja so stabilna stanja mreže, ki se razlikujejo od osnovnih pomnilnih vrednosti mreže.

Ob združitvi faz učenja in iskanja lahko delovanje Hopfieldove mreže opišemo tudi s pomočjo naslednjega psevodokoda:

Psevodokod Hopfieldovega algoritma

Vhodi: p $\xi_1, \xi_2, \ldots, \xi$ - shranjeni podatki, **x** – vhodni vzorci
Izhod: **y** – stabilna stanja
1: **function** HOPFIELDOVA MREŽA(_ , **x**, **y**)
2: izracunaj uteži glede na želeno množico podatkov
(p $\xi_1, \xi_2, \ldots, \xi$), ki jih želimo shraniti (faza 1)
3: **foreach** vhodni vzorec **x** (faza 2)
4: inicializiraj stanje mreže: **y**(0) = **x**
5: **while** stanje mreže se vec ne spreminja **do**
6: izracunaj nadaljnja stanja mreže
7: **end while**
8: **end foreach**
9: **end function**

Zgled 1: Hopfieldova mreža z dvema ucnima vzorcema (dimenzija N = 3) ter tremi nevroni

Učna vzorca $\xi_1 = [\xi_{11}, \xi_{12}, \xi_{13}]^T = [1, -1, 1]^T$ in $\xi_2 = [\xi_{21}, \xi_{22}, \xi_{23}]^T = [-1, 1, -1]^T$ želimo shraniti. Naj bodo pragovi θ_i pri vseh nevronih enaki nič. Uteži mreže so določene z algoritmom Hebbovega učenja :

$$w_{11} = w_{22} = w_{33} = 0$$

$$w_{12} = \frac{1}{3}\sum_{\mu=1}^{2}\xi_{\mu 1}\xi_{\mu 2} = \frac{1}{3}(\xi_{11}\xi_{12} + \xi_{21}\xi_{22}) = \frac{1}{3}(1\cdot(-1) + (-1)\cdot 1) = -\frac{2}{3}$$

$$w_{13} = \frac{1}{3}\sum_{\mu=1}^{2}\xi_{\mu 1}\xi_{\mu 3} = \frac{1}{3}(\xi_{11}\xi_{13} + \xi_{21}\xi_{23}) = \frac{1}{3}(1\cdot 1 + (-1)\cdot(-1)) = \frac{2}{3}$$

$$w_{23} = \frac{1}{3}\sum_{\mu=1}^{2}\xi_{\mu 2}\xi_{\mu 3} = \frac{1}{3}(\xi_{12}\xi_{13} + \xi_{22}\xi_{23}) = \frac{1}{3}((-1)\cdot 1 + 1\cdot(-1)) = -\frac{2}{3}$$

Ostale uteži dobimo s simetrijo:

$$w_{21} = w_{12} = -\frac{2}{3}$$

$$w_{31} = w_{13} = \frac{2}{3}$$

$$w_{32} = w_{23} = -\frac{2}{3}$$

Vse uteži zapišemo v matriki uteži:

$$\mathbf{W} = \frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}$$

Glede na to, da imamo tri nevrone v mreži, obstaja 2^3 = 8 možnih stanj mreže. Od teh osmih stanj sta samo stanji [1, –1, 1] in [–1, 1, –1] stabilni, ostalih šest stanj je nestabilnih.
Uporabimo asinhrono dinamiko proženja. Stanje mreže določamo tako, da izberemo enega izmed nevronov in izračunamo lastno aktivacijo glede na trenutno veljavne vhode. Temu pravimo tudi, da se nevron sproži. Mreža se potem znajde v stanju, ki je bodisi enako prejšnjemu bodisi se od njega razlikuje v enem bitu.
Izberimo poskusni vektor $\mathbf{x_1} = [1, 1, 1]^T$. Predpostavimo, da je prvi nevron izbran za proženje. Aktivacijo izračunamo tako, da množimo prvo vrstico matrike **W** s stanjem $\mathbf{x}_1$:

$$\frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 0$$

Aktivacija 0 pomeni, da se ohrani obstoječe stanje prvega nevrona, tako da se stanje mreže ne spremeni.
Predpostavimo, da je drugi nevron izbran za proženje. Množimo drugo vrstico matrike **W** s stanjem $\mathbf{x}_1$:

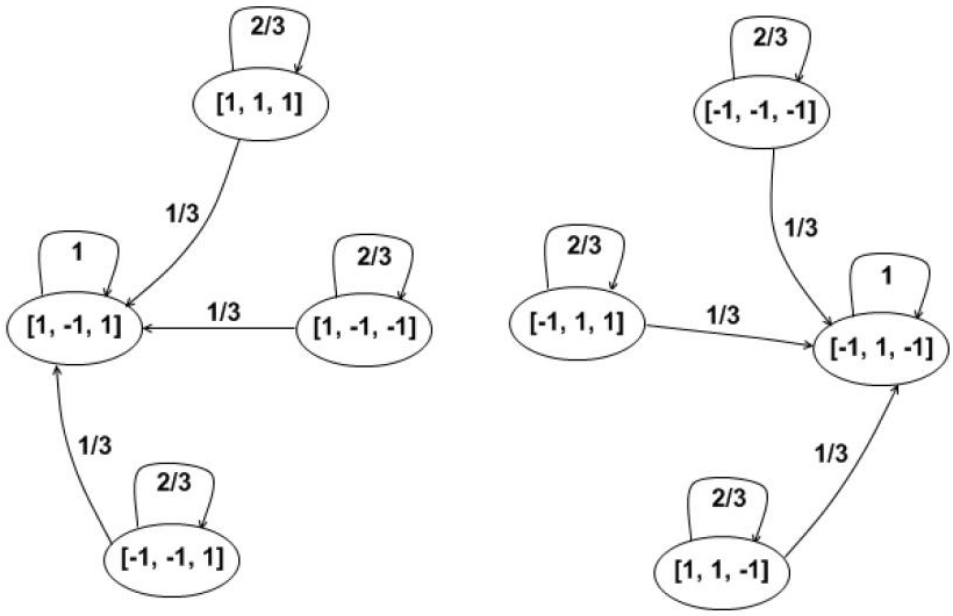
$$\frac{1}{3}\begin{bmatrix} -2 & 0 & -2 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = -\frac{4}{3}$$

Aktivacija – 4/3 pomeni, da je novo stanje drugega nevrona enako –1, kar spremeni celotno stanje mreže iz stanja $\mathbf{x}_1$ v stanje [1 ,–1, 1], kar pa je že stabilno stanje mreže oziroma naučeni vzorec.
Poglejmo še primer, če bi pri začetnem stanju $\mathbf{x}_1$ izbrali za proženje tretji nevron. Množimo tretjo vrstico matrike **W** s stanjem $\mathbf{x}_1$:

$$\frac{1}{3}\begin{bmatrix} 2 & -2 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 0$$

Aktivacija 0 pomeni, da se pri tretjem nevronu ohrani obstoječe stanje, tako da se tudi stanje mreže ne spremeni in je enako stanju $\mathbf{x}_1$.
Lahko rečemo, da se v dveh tretjinah primerov stanje ne spremeni, v eni tretjini primerov pa stanje $\mathbf{x}_1$ vodi v stabilno stanje, ki je tudi eno izmed dveh naučenih stanj mreže. Stanja, iz

katerih v nobenem primeru ne moremo priti v kako drugo stanje, so stabilna stanja. Prehode stanj lahko predstavimo tudi v obliki diagrama prehajanja stanj (slika 2.3). Vrednosti ob povezavah ponazarjajo verjetnost prehoda iz enega stanja v drugo ob naključni izbiri nevrona za proženje.



Slika 2.3: Diagram prehajanja stanj

V prejšnjem primeru smo imeli asinhrono proženje, kjer smo naključno izbrali nevron za proženje ter osvežili stanje. Poglejmo še, kaj bi dobili, če prožimo sinhrono (Littleov model). Vzemimo spet poskusni vektor $\mathbf{x_1} = [1, 1, 1]^T$. Sledi izračun stanja, če prožimo vse nevrone hkrati glede na stanje $\mathbf{x}_1$:

$$\mathbf{y}(1) = \mathrm{sgn}[\mathbf{W}\mathbf{y}(0)] = \mathrm{sgn}\left[\frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right] = \mathrm{sgn}\left[\frac{1}{3}\begin{bmatrix} 0 \\ -4 \\ 0 \end{bmatrix}\right]$$

Ker je $\mathbf{y}_2(1) = -4/3$, funkcija sgn določi vrednost -1. Vrednosti $\mathbf{y}_1(1)$ in $\mathbf{y}_3(1)$ sta 0, zato sta prvi in tretji nevron v stanju, kot sta bila pred prihodom vektorja $\mathbf{x}_1$. Torej:

$$\mathbf{y}(1) = [1, -1, 1]^T = \boldsymbol{\xi}_1$$

Ker je to že stabilno stanje, bo $\mathbf{y}(2) = \mathbf{y}(1)$. Že v eni iteraciji smo kot rezultat dobili stabilno stanje, ki ustreza prvemu shranjenemu vzorcu. Na podoben nacin lahko rešimo tudi druge poskusne vektorje.

2.2 Energijska funkcija

Hopfieldove mreže imajo vsakemu stanju mreže prirejeno skalarno vrednost, ki je določena z energijsko funkcijo. Lastnost energijske funkcije je, da se ji vrednost zmanjšuje vsakič, ko mreža spremeni svoje stanje. Funkcija doseže lokalni minimum, ko dosežemo stabilno stanje mreže. Zapis energijske funkcije:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ i\neq j}}^{N} w_{ji} y_i y_j + \sum_{i}^{N} \theta_i y_i$$

Če predpostavimo, da so pragovi θ_j enaki 0 za vse *j*, je energijska funkcija diskretne verzije Hopfieldove mreže definirana z naslednjo enačbo:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ i\neq j}}^{N} w_{ji} y_i y_j$$

Energijska funkcija je simetrična, saj njena vrednost ostane nespremenjena, če so stanja nevronov nasprotna. To pomeni, da če osnovni pomnilni vzorec ξ_μ ustreza lokalnemu minimumu energijske funkcije, istemu lokalnemu minimumu ustreza tudi vzorec $-\xi_\mu$.
Posledica algoritma Hopfieldove mreže, ki poskrbi, da vsako nestabilno stanje mreže postopoma konvergira k stabilnemu stanju, je, da je energijska funkcija monotono padajoča. Vrednosti energijske funkcije bodo upadale, dokler funkcija ne doseže lokalnega minimuma na **energijskem površju** (*energy landscape*), ki opisuje odvisnost energijske funkcije od stanja mreže glede na določeno množico sinaptičnih uteži. Recimo, da vhodni vzorec mreže predstavlja začetno točko na energijskem površju. Algoritem Hopfieldove mreže bo napredoval po površju proti lokalnemu minimumu. Ko bo lokalni minimum dosežen, se bo algoritem ustavil, saj bi vsak naslednji premik pomenil »vzpon« po energijskem površju.

Zgled 2: izračun energijske funkcije za primer Hopfieldove mreže s tremi nevroni iz zgleda 1
Če vstavimo vrednosti v energijsko enačbo, dobi energijska funkcija naslednjo obliko:

$$E = \frac{2}{3}(y_1 y_2 + y_2 y_3 - y_1 y_3)$$

Preglednica 3.1 kaže vrednosti *E* za vseh osem možnih stanj mreže. Vrednosti v preglednici potrjujejo, da energijska funkcija doseže svojo minimalno vrednost v stabilnih stanjih mreže.
Preglednica 3.1: Izračun energijske funkcije za Hopfieldovo mrežo iz zgleda 1

y_1	y_2	y_3	E
−1	−1	−1	2/3
+1	−1	−1	2/3
−1	+1	−1	−2
+1	+1	−1	2/3
−1	−1	+1	2/3
+1	−1	+1	−2
−1	+1	+1	2/3
+1	+1	+1	2/3