P. Vienslāņa perceptrons (Single-Layer Perceptron)

Vienslāņa perceptrons (*Single-Layer Perceptron, SLP*) vai, vienkārši, perceptrons ir viens no atpazīstamākajiem neironu tīklu modeļiem, kuru jau 20.g. 60. gadu sākumā piedāvāja Frenks Rozenblats (*Frank Rosenblat*t). Savā klasiskajā formā tas praktiski netiek izmantots, jo nespēj risināt nelineāras klasifikācijas problēmas, tomēr ļoti daudzi neironu tīklu modeļi ir veidojušies tieši uz perceptrona bāzes, pazīstamākais no kuriem ir daudzslāņu perceptrons ar kļūdu atgriezeniskās izplatīšanās apmācības metodi (*Multilayer Perceptron; Error Backpropagation*). Perceptrons realizē paraugu klasifikāciju (atpazīšanu) vai, no matemātiskās puses skatoties, funkcijas modelēšanu (aproksimāciju).

P1. PERCEPTRONA UZBŪVE UN DARBĪBAS PRINCIPI.

P1.1. PARAUGU KLASIFIKĀCIJA, FUNKCIJAS APROKSIMĀCIJA.

Daudzas reālās dzīves problēmas var tikt interpretētas kā klasifikācijas problēmas. Piemēram, bankas darbinieks kredītu daļā klasificē klientus vismaz divās grupās – tajos, kuriem var un tajos, kuriem nevar izsniegt kredītu. Bankas klientu klasifikācijas pamatā ir zināmais katra klienta riska līmenis, kuru var noteikt ienākumu līmenis, iepriekšējo kredītsaistību izpilde utt. Lūk, piemērs triviālai funkcijai, kas varētu noteikt, dot vai nedot kredītu klientam x:

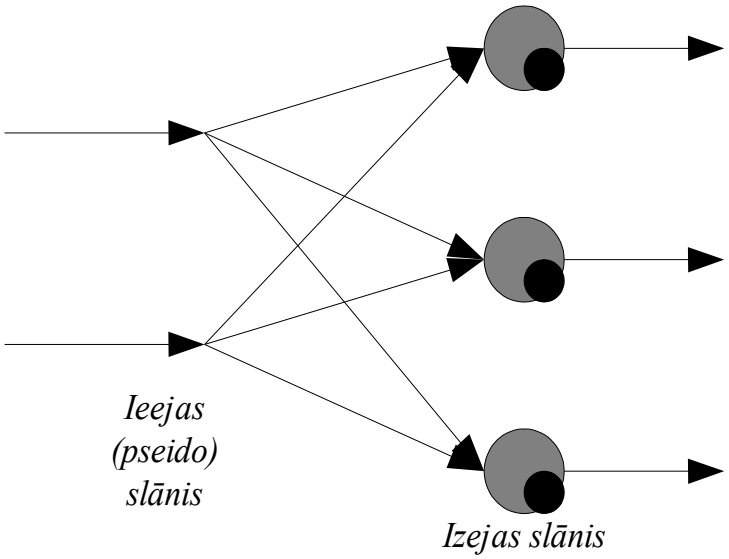
$$\text{dot\_kredītu(x)} = \begin{cases} jā\ ;\ alga(x) \geq 100 \\ nē\ ;\ alga(x) < 100 \end{cases}$$

Balstoties uz vēsturiskajiem datiem par klientiem un par viņiem zināmo informāciju, var apmācīt neironu tīklu, kas spētu jauniem klientiem izrēķināt viņu uzticamības līmeni (modelētu uzticamības funkciju).

P1.2. PERCEPTRONA GALVENĀS KONSTRUKTĪVĀS ĪPAŠĪBAS.

1. Divi slāņi – ieejas slānis (*input layer*) nosacīti sastāv tikai no neironu izejām un paredzēts tikai ievadvērtību uzstādīšanai) un izejas slānis (*output layer*). Ieejas slānis uzskatāms par neīstu jeb pseido-slāni, un vienīgais īstais slānis ir izejas slānis, tāpēc arī nosaukums – vienslāņa perceptrons. Ieejas slāni tikai tehnisku iemeslu dēļ vispār sauc par slāni (daudzslāņu perceptrona gadījumā ērtāk uzprogrammēt).

2. Slāņi izvietoti viens aiz otra. Katrs ieejas slāņa neirons ir savienots ar katru izejas slāņa neironu. Ja neironu tīklā ir m ieejas un n izejas, tad ieejas slānī ir m neironi un attiecīgi katram izejas slāņa neironam ir m svari (neskaitot t.s. nobīdi), kā arī izejas slānī ir n neironi.
3. Svari ir robežās *[-q; +q]*, kur q ir jebkurš pozitīvs skaitlis, bet parasti ievietojas intervālā *[-1; +1]*.
4. Neironu izejošās vērtības ir intervālā *[0; 1]*. Tas attiecas arī uz ieejas slāni, tādējādi arī visa neironu tīkla ieejas vērtības ir padodamas šajā intervālā.
5. Katrā neironā ir viens papildus svars (bez atbilstošās ieejas no ieejas slāņa), saukts par nobīdi (*bias*) vai līdzsvarojošo elementu. Ērtības labad šis papildus svars bieži tiek uzskatīts par parastu svaru un numurēts ar indeksu 0, un līdz ar to klāt nāk viena papildus ieeja, kas konstanti aizpildīta ar 1.
6. Aktivizācijas funkcija var būt gan lineāra, gan sliekšņveida, gan sigmoidāla.



Attēls P1. Perceptrona grafs ar 2 ieejām un 3 izejām. Katrā izejas slāņa neironā ir papildus svars – nobīde.

P1.3. PERCEPTRONA DARBĪBAS PRINCIPI.

1. Apmācība notiek iteratīvi. Katrā tīkla apmācības solī uz tīkla ieejām tiek uzstādītas ieejas vērtības un katrs izejas slāņa neirons veic vienu apmācības soli.

2. Apmācības procesu beidz, ja kopējā kļūda uz visiem apmācības piemēriem sasniedz pietiekoši zemu līmeni vai arī, ja sasniegts pārāk liels iterāciju skaits. Beidzamais gadījums var nozīmēt arī to, ka uz dotajiem piemēriem tīklu vispār nevar apmācīt.
3. Katrs apmācības piemērs ir divnieks, kuru varētu interpretēt kā {paraugs, piešķiramā kategorija} vai {jautājums, pareizā atbilde}, vai {argumentu vērtība, funkcijas vērtība}.
4. Paraugu atpazīšana notiek vienā solī un salīdzinoši ātri.

P2. APMĀCĪTA PERCEPTRONA DARBINĀŠANA.

Tīkla inicializācija ar ievadvērtībām.

$$O_1 \leftarrow X$$

kur $X[m]$ *– paraugs,*

$O_1[m]$ *– ieejas slāņa neironu izejas.*

Parauga atpazīšana notiek vienā solī.

Katram izejas slāņa neironam notiek aprēķini – formulas P1 un P2.

Summēšanas funkcija.

$$NET_j = b_j + \sum_{i=1}^{n} w_{ji} o_i \qquad \text{(P1)}$$

kur NET_j *– j-tā neirona summēšanas vērtība,*

w_{ji} *– j-tā neirona i-tais svars,*

b_j *– j-tā neirona papildus svars (nobīde),*

o_i *– ieejas slāņa i-tā neirona izejas vērtība.*

Summēšanas funkcijas variants, ja papildus svars b_j tiek identificēts kā 0-tais svars w_{j0}.

$$NET_j = \sum_{i=0}^{n} w_{ji} o_i \qquad \text{(P1a)}$$

kur NET_j *– j-tā neirona summēšanas vērtība,*

w_{ji} *– j-tā neirona i-tais svars (*w_{j0} *– papildus svars),*

$o_i = 1$ *(i=0),*

o_i *– ieejas slāņa i-tā neirona izejas vērtība (i=[1..m]).*

Aktivizācijas funkcija.

Parasti tiek lietota sigmoidāla funkcija (P2, P2a), bet iespējama arī sliekšņveida (P2d) vai lineāra (P2b, P2c). Svarīgi atcerēties, ka atkarībā no izmantotās aktivizācijas funkcijas atšķiras pielietojamais apmācības algoritms.

Formulā P2 parādīta sigmoidāla aktivizācijas funkcija, izmantojot t.s. loģistisko funkciju (**$\mathbf{f}_{log}$**). Cits sigmoidālas funkcijas variants būtu hiperboliskais tangenss (*tanh*).

$$o_j = f_{\log}(NET_j) = \frac{1}{1+\mathrm{e}^{-\frac{1}{g}NET_j}} \qquad \text{(P2)}$$

kur NET_j *– j-tā neirona summēšanas funkcija,*
o_j *– j-tā neirona izeja,*
g *– līknes slīpuma koeficients (gain).*

Ērtības nolūkos lietosim līknes slīpuma koeficientu $h = \frac{1}{g}$, tādējādi, nedaudz vienkāršojot formulu P2, iegūstot P2a.

$$o_j = f_{log2}(NET_j) = \frac{1}{1+\mathrm{e}^{-h\,NET_j}} \qquad \text{(P2a)}$$

kur NET_j *– j-tā neirona summēšanas funkcija,*
o_j *– j-tā neirona izeja,*
h *– līknes slīpuma koeficients.*

Formulā P2b parādīts lineāras aktivizācijas funkcijas triviālais variants.

$$f_{lin}(NET_j) = o_j = NET_j \qquad \text{(P2b)}$$

kur NET_j *– j-tā neirona summēšanas funkcija,*
o_j *– j-tā neirona izeja.*

Formulā P2c parādīts lineāras aktivizācijas funkcijas variants, ierobežojot vērtības intervālā [0..1].

$$o_j = f_{lin2}(NET_j) = \begin{Bmatrix} 0\,;\, NET_j < 0 \\ 1\,;\, NET_j > 1 \\ NET_j\,;\, NET_j \in [0..1] \end{Bmatrix} \qquad \text{(P2c)}$$

kur NET_j *– j-tā neirona summēšanas funkcija,*
o_j *– j-tā neirona izeja.*

Formulā P2d parādīta sliekšņveida aktivizācijas funkcija.

$$o_j = f_\Theta(NET_j) = \begin{cases} 0\,;\, NET_j \leqslant \Theta \\ +1\,;\, NET_j > \Theta \end{cases} \qquad \text{(P2d)}$$

kur NET_j *– j-tā neirona summēšanas funkcija,*

o_j *– j-tā neirona izeja,*

Θ *- sliekšņa vērtība.*

Rezultāts.

Pēc tīkla darbināšanas (uz paraugu X) tīkla dotā atbilde atrodas uz izejas slāņa neironu izejām O_2.

P3. PERCEPTRONA APMĀCĪBA AR ITERĀCIJU METODI.

P3.1. APMĀCĪBAS PROCESS.

Dots neironu tīkls {O_1[m], W_2[n][m+1], O_2[n]}, kurā ir m ieejas un n izejas. Izejas slānī ir n neironi, un katrā neironā ir m svari, kā arī papildus svars – nobīde. (Piemērs, kad m=2 un n=3, parādīts attēlā P1.) Tiek izmantota sigmoidālā vai lineārā aktivizācijas funkcija.

Dots testa piemēru kopums P, kurā ir piemēri skaitā r. Katrs testa piemērs tiek identificēts kā {P_s, D_s}, kur s=1..r. Katra ieejas parauga P_s garums ir m, bet izejas parauga (vēlamās atbildes) D_s garums ir n.

Svaru korekcijas metodes pamatā ir **kļūdas kvadrātu minimizācijas** princips. Pie neironu tīkliem to sauc arī par **Delta likumu** (*Delta rule*) vai **Vidrova-Hofa** (*Widrow-Hoff*) likumu.

Apmācības uzdevums. Iegūt spēju atpazīt paraugus, klasificējot tos kādā no kategorijām.

Apmācības process notiek šādi:

1. Visu svaru inicializācija ar gadījuma vērtībām intervālā [-x, +x], kur 0<x<1. Parasti x vērtība ir tuvāka nullei nekā 1, piemēram, 0.3.
2. Apmācības procesa veikšana pēc kārtas uz visiem apmācības piemēriem, atkārtojot to tik ilgi, kamēr tiek sasniegts konverģences nosacījums – vai nu tīkla kopējā kļūda kļūst pietiekoši maza vai arī iterāciju skaits kļūst pārāk liels (formulas P3, P6, P7).

Katrā apmācības procesa solī tiek veiktas šādas darbības:

1. Tīkla inicializācija ar parauga P_s vērtībām.
2. Tīkla darbināšana ar doto paraugu (formulas P1, P2 (P2a) vai P2b (P2c)).
3. Svaru korekcija, balstoties uz vēlamā rezultāta D_s atšķirības no tīkla dotā rezultāta O (formulas P3, P4 vai P4b, P5).

Apmācības procesa daļu, kurā tieši vienu reizi notiek apmācība uz visiem apmācības piemēriem, sauc par **epohu** *(epoch)*.

Katra neirona kļūdas izrēķināšana fiksētam apmācības paraugam.

$$e_j = d_j - o_j \tag{P3}$$

kur e_j *– j-tā neirona kļūda,*

d_j *– j-tā neirona vēlamā izeja,*

o_j *– j-tā neirona faktiskā izeja.*

Svara korekcijas aprēķināšana neironam ar sigmoidālu (loģistisko) aktivizācijas funkciju:

$$\Delta w_{ji}(t+1) = \eta\, h\, o_i\, e_j\, o_j (1 - o_j) + \alpha\, \Delta w_{ji}(t) \tag{P4}$$

kur $\Delta w_{ji}(t+1)$ *– j-tā neirona i-tā svara izmaiņa,*

$\Delta w_{ji}(t)$ *– j-tā neirona i-tā svara izmaiņa iepriekšējā solī,*

η *(eta) – apmācības koeficients [0..1],*

$h = \frac{1}{g}$ *– līknes slīpuma koeficients,*

o_i *– i-tais ienākošais signāls (ieejas slāņa i-tā neirona izeja),*

e_j *– j-tā neirona izrēķinātā kļūda,*

o_j *– j-tā neirona izrēķinātā izejas vērtība,*

α *(alfa) – inerces faktors (momentum factor), [0..1] – nosaka, cik lielā mērā iepriekšējā iterācijas solī izrēķinātā svara korekcija ietekmē šī soļa svara korekciju.*

Svara korekcijas aprēķināšana neironam ar lineāru aktivizācijas funkciju:

$$\Delta w_{ji}(t+1) = \eta\, o_i\, e_j + \alpha\, \Delta w_{ji}(t) \tag{P4b}$$

kur visi apzīmējumi atbilst formulas P4 apzīmējumiem.

Formāli šo metodi nevar pielietot neironiem ar sliekšņveida aktivizācijas funkciju, jo šāda funkcija nav diferencējama, tomēr praktiski arī sliekšņveida aktivizācijas funkcijas gadījumā dažreiz ir iespējams lietot formulu P4b.

Svara korekcija.

$$w_{ji}(t+1)=w_{ji}(t)+\Delta w_{ji} \tag{P5}$$

kur $w_{ji}(t+1)$ *– jaunā svara vērtība,*

$w_{ji}(t)$ *– vecā svara vērtība,*

Δw_{ji} *– svara izmaiņa,*

t *– soļa numurs (1..N).*

Tīkla kvadrātiskās kļūdas izrēķināšana visiem paraugiem (epohai kopumā).

$$E=\frac{1}{2}\sum_{s=1}^{r}\sum_{j=1}^{n} e_{sj}^2 \tag{P6}$$

kur E *– kopējā kvadrātiskā kļūda ,*

e_{sj} *– j-tā neirona kļūda, darbinot uz s-tā parauga,*

r *– testa piemēru skaits,*

n *– neironu skaits izejas slānī,*

0.5 – koeficients ar tehnisku nozīmi – aprēķinu vienkāršošanai.

Kvadrātiska neirona kļūdas funkcija (formulas P6 speciālgadījums vienam neironam un fiksētam apmācības piemēram).

$$E_j=\frac{1}{2}e_j^2 \tag{P6a}$$

kur E_j *– j-tā neirona kvadrātiskā kļūda,*

e_j *– j-tā neirona kļūda.*

Apmācības procesa konverģences nosacījums

$$E\le\epsilon\vee u>UMAX \tag{P7}$$

kur E *– kopējā kvadrātiskā kļūda,*

ϵ *(epsilon) – maksimālā pieļaujamā kļūda,*

u *– kārtējās epohas numurs,*

$UMAX$ *– maksimālais pieļaujamais epohu skaits.*

P3.2. APMĀCĪBAS PROCESA SPECIFISKAS LIETAS.

Apmācības koeficients.

Apmācības koeficients η (eta) formulās P4 un P4b uzrādīts kā konstante, tomēr tas apmācības procesa gaitā var mainīties – laika gaitā pakāpeniski samazinoties. Piedāvātie izmaiņu grafiki ņemti no [Haykin, 1999].

Viena no vienkāršākajām apmācības koeficienta izmaiņas funkcijām ir t.s. stohastiskās aproksimācijas grafiks (S*tochastic Approximation Schedule*):

$$\eta(u)=\frac{c}{u} \tag{P10}$$

kur c *– lietotāja izvēlēta konstante,*

u *– iterācijas (epohas) numurs 0, 1, ...*

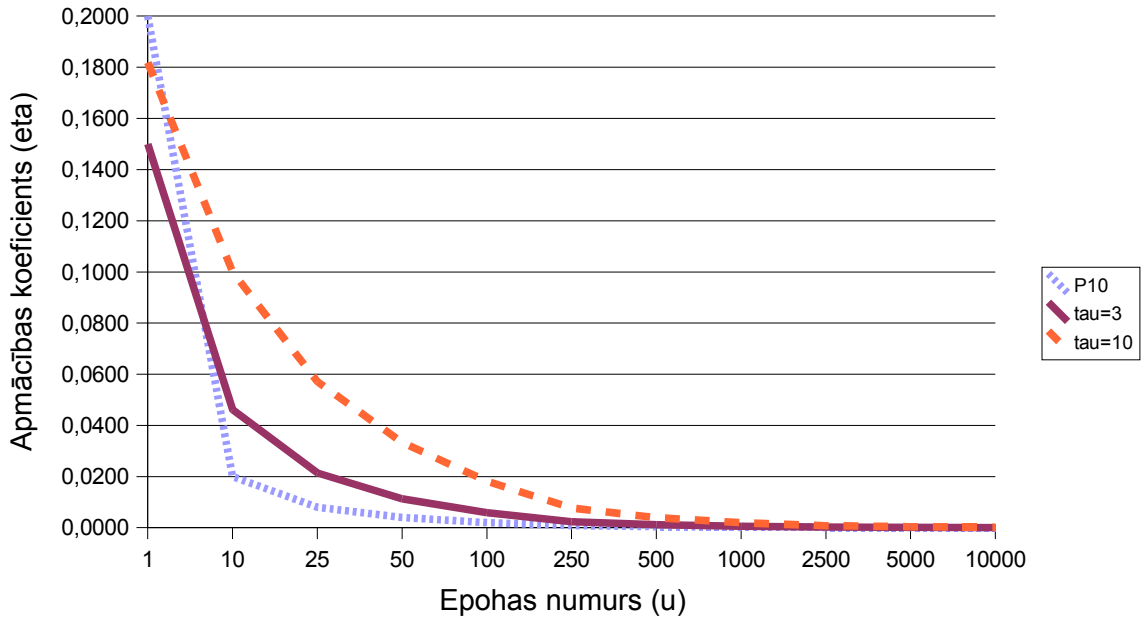
Otrs ir "Vispirms atrast, tad precizēt" grafiks (*search-then-converge*):

$$\eta(u)=\frac{\eta_0}{1+\frac{u}{\tau}} \tag{P11}$$

kur η_0 *– apmācības koeficienta sākuma vērtība,*

τ *– apmācības koeficienta izmaiņas ātrums,*

u *– iterācijas (epohas) numurs 0, 1, ...*



Attēls P3. Apmācības koeficientu izmaiņas grafiks pie sākuma konstantēm $\eta_0 = c = 0.1$. *Formulai P10 dots viens piemērs, bet formulai P11 – divi, attiecīgi mainot koeficientu* τ .

Apmācības procesa 2 režīmi.

Vienas epohas ietvaros apmācības algoritms var tikt darbināts kādā no diviem režīmiem:

1. Secīgais režīms (*sequential mode*);
2. Pakešu režīms (*batch mode*).

Secīgajā režīmā svaru izmaiņa notiek pēc katra apmācības parauga darbināšana, bet pakešu režīmā – tikai epohas beigās. Teorētiski pakešu režīma izmantošana nodrošina precīzāku (ātrāku) rezultātu, tomēr secīgais režīms tiek praktiski lietots biežāk.

P4. PERCEPTRONA APMĀCĪBA AR PSEIDO-APGRIEZTĀS MATRICAS (PSEUDO-INVERSE) METODI.

Dots neironu tīkls {$O_1[m]$, $W_2[n][m+1]$, $O_2[n]$}, kurā ir m ieejas un n izejas. Izejas slānī ir n neironi, un katrā neironā ir m svari, kā arī papildus svars – nobīde. (Piemērs, kad m=2 un n=3, parādīts attēlā P1.) Tiek izmantota lineārā aktivizācijas funkcija ar ierobežojumu intervālā [0, 1].

Dots testa piemēru kopums P, kurā ir piemēri skaitā m. Katrs testa piemērs tiek identificēts kā $\{P_s, D_s\}$, kur s=1..m. Katra ieejas parauga P_s garums ir m, bet izejas parauga (vēlamās atbildes) D_s garums ir n.

Apmācības piemēru kopums ir matricas P un D, kas ir šādā formā:

$$P = \begin{pmatrix} p_{11} & p_{12} & \dots & p_{1m} \\ p_{21} & p_{22} & \dots & p_{2m} \\ \dots & \dots & \dots & \dots \\ p_{s1} & p_{s2} & \dots & p_{sm} \end{pmatrix}$$

kur p_{ri} *– r-tā apmācības parauga i-tais elements.*

$$D = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \dots & \dots & \dots & \dots \\ d_{s1} & d_{s2} & \dots & d_{sn} \end{pmatrix}$$

kur d_{rk} *– r-tā apmācības parauga vēlamā rezultāta k-tais elements.*

Ņemot vērā perceptrona neirona specifiku, ka ir viens papildus svars, matricai P sākumā pievienosim vienu papildus kolonnu, kas aizpildīta ar 1 (salīdzināt formulas P1 un P1a).

Izmantojot *Pseudo-inverse* metodi, svaru matrica W tiek iegūta (respektīvi, tiek veikta tīkla apmācība), izmantojot formulu P8:

$$W = P^{+} D \qquad \text{(P8)}$$

kur W *– neironu tīkla svaru matrica,*
P *– apmācības paraugu matrica,*
D *– apmācības paraugu vēlamo rezultātu matrica,*
$(\cdot)^{+}$ *– pseudo-inverse operators (kurš definēts formulā P9).*

Pseido-apgrieztās matricas (*pseudo-inverse*) operators:

$$X^{+} = (X^{T} X)^{-1} X^{T} \qquad \text{(P9)}$$

kur $(\cdot)^{+}$ *– pseudo-inverse operators,*
$(\cdot)^{-1}$ *– matricas apgriešanas operators,*
$(\cdot)^{T}$ *– matricas transponēšanas operators.*

Ar formulas P8 palīdzību iegūta svaru matrica W reprezentē apmācītu neironu tīklu un ir pierakstāma formā:

$$W = \begin{pmatrix} w_{10} & w_{20} & \dots & w_{n0} \\ w_{11} & w_{21} & \dots & w_{n1} \\ \dots & \dots & \dots & \dots \\ w_{1m} & w_{2m} & \dots & w_{nm} \end{pmatrix}$$

kur w_{ik} *– i-tā neirona k-tais svars (i=1..n, k=0..m). Ja k=0, tad* w_{ik} *– papildus svars (bias).*

Var teikt, ka no matemātiskā viedokļa tīkla apmācība ar pseido-apgrieztās matricas metodi notiek vienā solī, tomēr no reālo aprēķinu viedokļa tas tā nebūt nav. Vēl vairāk – ar šādu metodi apmācīt netriviālu neironu tīklu ir praktiski neiespējami, jo matricas apgriešana, kas ir sarežģītākā šī algoritma komponente, pat salīdzinoši zemas kārtas matricām ir ļoti dārga no izpildes viedokļa.

Matricas transponēšana ir pārveidojums, kurā kādas matricas rindas aizvieto ar tās kolonnām.

Par matricas X **apgriezto matricu** sauc tādu matricu A^{-1}, kam spēkā sakarība $AA^{-1}=E$, kur E – vienības matrica.

Vienības matrica ir tāda kvadrātiska matrica, kam diagonāle, kas iziet no kreisā augšējā stūra, ir aizpildīta ar 1, bet pārējā daļa – ar 0.

Matricas X apgrieztās matricas X^{-1} iegūšanas viens no variantiem.

1. Atrod det X; ja det X = 0, tad apgrieztā matrica neeksistē.
2. Iegūst X^* – adjunktu matricu.
3. Iegūst matricas X^* transponēto matricu, kuru sauc par matricas X piesaistīto matricu $P = X^{*T}$.
4. Katru piesaistītas matricas P elementu dala ar matricas X determinantu un iegūst apgriezto matricu: $X^{-1} = P / \det X$.

Determinants. Otrās kārtas determinantu matricai A definē kā $\det A = a_{11}a_{22}-a_{21}a_{21}$.

Minors. Par elementam a_{ij} atbilstošo minoru M_{ij} sauc tādu n-1 kārtas determinantu, ko iegūst, dotajā determinantā svītrojot i-to rindu un j-to kolonnu.

Adjunkts. Par elementam a_{ij} atbilstošo algebrisko papildinājumu jeb adjunktu sauc skaitli $A_{ij} = (-1)^{i+j}*M_{ij}$.

Augstākas kārtas determinants. Determinantu var iegūt kā summu no reizinājumiem, kurus iegūst kādas rindas (kolonnas) visus elementus reizinot ar to atbilstošajiem adjunktiem.

P5. TRIVIĀLA PERCEPTRONA PIEMĒRS.

P5.1. UZDEVUMA NOSTĀDNE.

Uzdevums. Izveidot perceptronu ar divām ieejām un vienu izeju un apmācīt uz funkciju OR ("loģiskais VAI"), un nodemonstrēt apmācītā tīkla darbību.

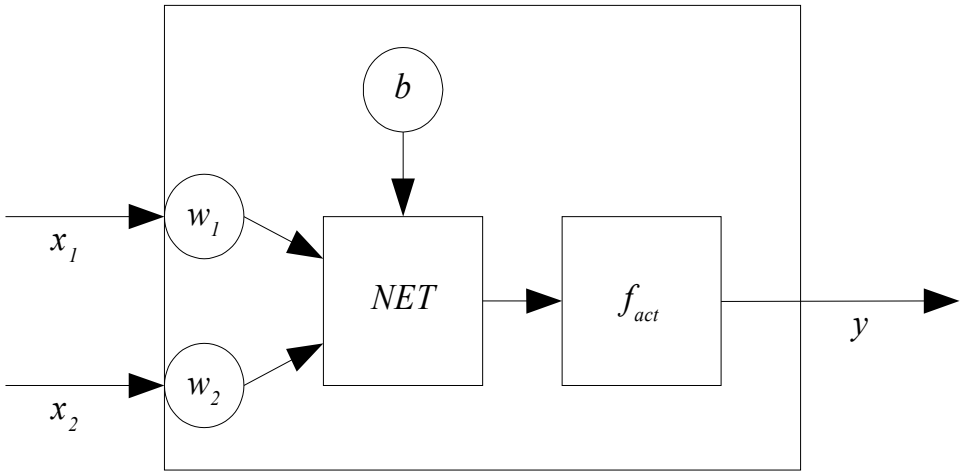
Tabula P1. Loģiskā VAI definēšana.

x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1

P5.2. TĪKLA KONSTRUĒŠANA UN APMĀCĪBAS SAGATAVOŠANA.

Veidojamā perceptrona uzbūve.

Perceptronam (sk. attēlu P2) ir divas ieejas un viena izeja (m=2, n=1), tātad tas sastāv no viena neirona izejas slānī. Piemērā uzskatāmības dēļ ieejas vērtības apzīmēsim ar x_1 un x_2, bet izejas vērtību ar y. Neirona kļūdu apzīmēsim ar e_{err}, lai nosaukums nejuktu ar konstanti e.



Attēls P2. Triviāla perceptrona piemērs (viens neirons ar divām ieejām un vienu izeju).

Apmācības piemēri.

Atbilstoši loģiskā VAI definīcijai, ir 4 apmācības piemēri, kur, katra piemēra pirmais paraugs ir garumā 2, bet otrais – garumā 1.

$P_1 = \{0, 0\}, D_1 = \{0\};$
$P_2 = \{0, 1\}, D_2 = \{1\};$
$P_3 = \{1, 0\}, D_3 = \{1\};$
$P_4 = \{1, 1\}, D_4 = \{1\}.$

P5.3. APMĀCĪBAS PROCESS AR ITERĀCIJU METODI (SIGMOIDĀLA (LOĢISTISKĀ) AKTIVIZĀCIJAS FUNKCIJA).

Pieņemsim, ka tīkls $\{w_1, w_2, b\}$ inicializēts ar sākuma vērtībām $\{0.3, 0.2, -0.3\}$.

Apmācības koeficients $\eta = 1.0$.

Inerces faktors $\alpha = 0$. Līdz ar to otrais saskaitāmais formulā P4 tiek izlaists.

Maksimālā pieļaujamā kļūda $\epsilon = 0.05$.

Sigmoidālā aktivizācijas funkcija, līknes slīpuma koeficients h = 5.

Maksimālais pieļaujamais epohu skaits UMAX = 100.

Algoritms (sigmoidālai funkcijai).

Neironu tīkls tiek inicializēts ar sākuma vērtībām (1. rinda, kolonnas w_1, w_2, b).

Katrā solī notiek šādas darbības (1-6):

1. Parauga vērtību uzstādīšana uz tīkla ieejām.
2. NET vērtības aprēķināšana (formula P1): $NET = x_1w_1+x_2w_2+b$.
3. Neirona izejas vērtības aprēķināšana (formula P2a): $y = f_{log2}(NET)$.
4. Kļūdas izrēķināšana (formula 3): $e_{err} = d - y$.
5. Svaru korekcijas aprēķināšana nākošajam solim t+1 no izrēķinātajām vērtībām solī t un svaru korekcija (formulas P4, P5): $w_i(t+1) = w_i(t) + 5\, x_i e_{err}\, y(1-y)$.
6. Ja epohas beigas, tad aprēķināt epohas kopējo kļūdu E (formula P6) un ja kļūda ir kļuvusi mazāka par maksimāli pieļaujamo ($\epsilon = 0.05$) vai arī epohas numurs u pārsniedz maksimāli pieļaujamo (formula P7), tad beigt, citādi turpināt ar nākošo soli, atgriežoties punktā 1.

Tabula P2. Algoritma demonstrācija uz testa piemēriem.

Epoha, u	Solis, t	Par. nr.	x_1	x_2	w_1	w_2	b	NET	y	d	e_{err}	E
1	1	P_1	**0**	**0**	0,3	0,2	-0,3	-0,3	**0,18**	**0**	-0,18	
	2	P_2	**0**	**1**	0,3	0,2	-0,44	-0,24	**0,24**	**1**	0,76	
	3	P_3	**1**	**0**	0,3	0,89	0,25	0,55	**0,94**	**1**	0,06	
	4	P_4	**1**	**1**	0,32	0,89	0,27	1,47	**1**	**1**	0	0,62
2	5	P_1	**0**	**0**	0,32	0,89	0,27	0,27	**0,79**	**0**	-0,79	
	6	P_2	**0**	**1**	0,32	0,89	-0,38	0,5	**0,93**	**1**	0,07	
	7	P_3	**1**	**0**	0,32	0,91	-0,36	-0,04	**0,45**	**1**	0,55	
	8	P_4	**1**	**1**	1	0,91	0,32	2,23	**1**	**1**	0	0,94
3	9	P_1	**0**	**0**	1	0,91	0,32	0,32	**0,83**	**0**	-0,83	
	10	P_2	**0**	**1**	1	0,91	-0,25	0,66	**0,96**	**1**	0,04	
	11	P_3	**1**	**0**	1	0,92	-0,25	0,75	**0,98**	**1**	0,02	
	12	P_4	**1**	**1**	1	0,92	-0,24	1,68	**1**	**1**	0	0,7
4	13	P_1	**0**	**0**	1	0,92	-0,24	-0,24	**0,23**	**0**	-0,23	
	14	P_2	**0**	**1**	1	0,92	-0,44	0,47	**0,91**	**1**	0,09	
	15	P_3	**1**	**0**	1	0,95	-0,41	0,59	**0,95**	**1**	0,05	
	16	P_4	**1**	**1**	1,01	0,95	-0,4	1,57	**1**	**1**	0	0,06
5	17	P_1	**0**	**0**	1,01	0,95	-0,4	-0,4	**0,12**	**0**	-0,12	
	18	P_2	**0**	**1**	1,01	0,95	-0,46	0,49	**0,92**	**1**	0,08	
	19	P_3	**1**	**0**	1,01	0,98	-0,43	0,58	**0,95**	**1**	0,05	
	20	P_4	**1**	**1**	1,03	0,98	-0,42	1,59	**1**	**1**	0	0,02

Pēc kolonnas "y" pēdējām 4 vērtībām redzams, ka tajā esošās vērtības ir diezgan tuvu vēlamajām, tātad tīkls ir apmācījies.

P5.4. APMĀCĪBAS PROCESS AR ITERĀCIJU METODI (LINEĀRA AKTIVIZĀCIJAS FUNKCIJA).

Pieņemsim, ka tīkls $\{w_1, w_2, b\}$ inicializēts ar sākuma vērtībām {-0.1, -0.2, 0.3}.

Apmācības koeficients $\eta = 1.0$.

Inerces faktors $\alpha = 0$. Līdz ar to otrais saskaitāmais formulā P4 tiek izlaists.

Maksimālā pieļaujamā kļūda $\epsilon = 0.1$.

Lineārā aktivizācijas funkcija ar ierobežotu vērtību intervālā [0, 1].

Maksimālais pieļaujamais soļu skaits UMAX = 100.

Algoritms (lineārai funkcijai).

Algoritms ir tāds pats kā, veicot apmācību tīklam ar sigmoidālo aktivizācijas funkciju, atšķiras tikai formulu pielietojums.

Neironu tīkls tiek inicializēts ar sākuma vērtībām (1. rinda, kolonnas w_1, w_2, b).

Katrā solī notiek šādas darbības (1-6):

1. Parauga vērtību uzstādīšana uz tīkla ieejām.
2. NET vērtības aprēķināšana (formula P1): NET = $x_1w_1+x_2w_2+b$.
3. Neirona izejas vērtības aprēķināšana (formula P2c): y = f_{lin2} (NET).
4. Kļūdas izrēķināšana (formula 3): e_{err} = d – y.
5. Svaru korekcijas aprēķināšana nākošajam solim t+1 no izrēķinātajām vērtībām solī t un svaru korekcija (formulas P4b, P5): $w_i(t+1)=w_i(t)+x_i e_{err}$.
6. Ja epohas beigas, tad aprēķināt epohas kopējo kļūdu E (formula P6) un ja kļūda ir kļuvusi mazāka par maksimāli pieļaujamo ($\epsilon=0.1$) vai arī epohas numurs u pārsniedz maksimāli pieļaujamo (formula P7), tad beigt, citādi turpināt ar nākošo soli, atgriežoties punktā 1.

Tabula P3. Algoritma demonstrācija uz testa piemēriem.

Epoha, u	Solis, t	Par. nr.	x_1	x_2	w_1	w_1	b	NET	y	d	e_{err}	E
1	1	P_1	**0**	**0**	-0,1	-0,2	0,3	0,3	**0,3**	**0**	-0,3	
	2	P_2	**0**	**1**	-0,1	-0,2	0	-0,2	**0**	**1**	1	
	3	P_3	**1**	**0**	-0,1	0,8	1	0,9	**0,9**	**1**	0,1	
	4	P_4	**1**	**1**	0	0,8	1,1	1,9	**1**	**1**	0	1,1
2	5	P_1	**0**	**0**	0	0,8	1,1	1,1	**1**	**0**	-1	
	6	P_2	**0**	**1**	0	0,8	0,1	0,9	**0,9**	**1**	0,1	
	7	P_3	**1**	**0**	0	0,9	0,2	0,2	**0,2**	**1**	0,8	
	8	P_4	**1**	**1**	0,8	0,9	1	2,7	**1**	**1**	0	1,65
3	9	P_1	**0**	**0**	0,8	0,9	1	1	**1**	**0**	-1	
	10	P_2	**0**	**1**	0,8	0,9	0	0,9	**0,9**	**1**	0,1	
	11	P_3	**1**	**0**	0,8	1	0,1	0,9	**0,9**	**1**	0,1	
	12	P_4	**1**	**1**	0,9	1	0,2	2,1	**1**	**1**	0	1,02
4	13	P_1	**0**	**0**	0,9	1	0,2	0,2	**0,2**	**0**	-0,2	
	14	P_2	**0**	**1**	0,9	1	0	1	**1**	**1**	0	
	15	P_3	**1**	**0**	0,9	1	0	0,9	**0,9**	**1**	0,1	
	16	P_4	**1**	**1**	1	1	0,1	2,1	**1**	**1**	0	0,05

Pēc kolonnas "y" pēdējām 4 vērtībām redzams, ka tajā esošās vērtības ir diezgan tuvu vēlamajām, tātad tīkls ir apmācījies.

P5.5. APMĀCĪBAS PROCESS AR PSEIDO-APGRIEZTĀS MATRICAS METODI (LINEĀRA AKTIVIZĀCIJAS FUNKCIJA).

Pieņemsim, ka tīkls {w_1, w_2, b} ir formā {w_0, w_1, w_2}.

Apmācības piemēri.

Salīdzinot ar iterāciju metodi, apmācības piemēru matricai ir papildus kolonna, kas aizpildīta ar 1 (sk. arī tabulu P1 un formulu P1a).

$$P=\begin{pmatrix}1 & 0 & 0\\ 1 & 0 & 1\\ 1 & 1 & 0\\ 1 & 1 & 1\end{pmatrix} \quad D=\begin{pmatrix}0\\ 1\\ 1\\ 1\end{pmatrix}$$

Svaru izrēķināšana notiek pēc formulas P8.

Apmācības process.

Matricas Q iegūšana, kurai jārēķina apgrieztā matrica.

$$Q=P^T P=\begin{pmatrix}1 & 1 & 1 & 1\\ 0 & 0 & 1 & 1\\ 0 & 1 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 0 & 0\\ 1 & 0 & 1\\ 1 & 1 & 0\\ 1 & 1 & 1\end{pmatrix}=\begin{pmatrix}4 & 2 & 2\\ 2 & 2 & 1\\ 2 & 1 & 2\end{pmatrix}$$

Matricas Q apgrieztās matricas Q^{-1} izrēķināšana.

$$det\,Q=4$$

$$Q^{*}=\begin{pmatrix}3 & -2 & -2\\ -2 & 4 & 0\\ -2 & 0 & 4\end{pmatrix}=Q^{*\mathrm{T}}$$

$$Q^{-1}=\frac{Q^{*\mathrm{T}}}{det\,Q}=\begin{pmatrix}\frac{3}{4} & -\frac{1}{2} & -\frac{1}{2}\\ -\frac{1}{2} & 1 & 0\\ -\frac{1}{2} & 0 & 1\end{pmatrix}$$

Matricas P pseido-apgrieztās matricas P^{+} izrēķināšana.

$$P^{+}=Q^{-1}P^{T}=\begin{pmatrix}\frac{3}{4} & -\frac{1}{2} & -\frac{1}{2}\\ -\frac{1}{2} & 1 & 0\\ -\frac{1}{2} & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 1 & 1 & 1\\ 0 & 0 & 1 & 1\\ 0 & 1 & 0 & 1\end{pmatrix}=\begin{pmatrix}\frac{3}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4}\\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2}\\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2}\end{pmatrix}$$

Svaru matricas W izrēķināšana.

$$W=P^{+}D=\begin{pmatrix}\frac{3}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4}\\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2}\\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2}\end{pmatrix}\cdot\begin{pmatrix}0\\1\\1\\1\end{pmatrix}=\begin{pmatrix}\frac{1}{4}\\ \frac{1}{2}\\ \frac{1}{2}\end{pmatrix}$$

Tātad $W=\{w_0, w_1, w_2\}=\{0.25, 0.5, 0.5\}$.

Pārbaude.

Pārbaude tiek veikta, izmantojot formulas P1a un P2c.

x_1	x_2	w_0	w_1	w_2	NET	y
0	**0**	0,25	0,5	0,5	0,25	**0,25**
0	**1**	0,25	0,5	0,5	0,75	**0,75**
1	**0**	0,25	0,5	0,5	0,75	**0,75**
1	**1**	0,25	0,5	0,5	1,25	**1**

Neironu tīkls ar pietiekamu precizitāti atpazīst funkciju “OR”. Atpazīšana būtu pilnīgi precīza, ja tiktu lietota sliekšņveida aktivizācijas funkcija (P2d).

P6. PERCEPTRONA MODEĻA NOVĒRTĒJUMS UN ĪPAŠĪBAS.

Šajā nodaļā aprakstītas vienslāņa perceptrona specifiskās īpašības. Perceptrona modeļu kopīgās īpašības aprakstītas nodaļā par daudzslāņu perceptronu (M).

Vienslāņa perceptrons ir efektīvs neironu tīkla modelis ar vienkāršu un pietiekoši ātru apmācību. Galvenais tā trūkums ir nespēja atrisināt nelineāras problēmas, piemēram, pat tik vienkāršu kā XOR problēmu.

Lielākais trieciens (vienslāņa) perceptronam tika dots 1969. gadā, kad iznāca Minska un Peiperta (Minsky, Papert) klasiskā grāmata “Perceptrons”, kur abi zinātnieki pierādīja, ka

Rozenblata perceptrons pēc definīcijas nav spējīgs veikt noteiktus vispārinājumus uz iemācīto piemēru pamata. Bez tam autori izteica pieņēmumu, ka Rozenblata perceptronam atklātie ierobežojumi varētu attiekties arī uz tā variantiem, t.sk. daudzslāņu neironu tīkliem. Tieši šis apgalvojums kļuva par vienu no galvenajiem iemesliem neuzticībai ne tikai perceptroniem, bet neironu tīkliem kopumā, kas tika pilnībā izkliedēta tikai 20. gs. 80. gadu vidū.

Lūk, fragments no grāmatas "Perceptrons" (Minsky, Papert, 1969).

> *The perceptron has shown itself worthy of study despite (and even because of!) its severe limitations. It has many features to attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension of multilayer systems is sterile.*

Tagad ir zināms, ka Minska un Peiperta pieņēmumi izrādījušies nepamatoti, jo ir zināmas vairāki neironu tīklu modeļi, kas ir no skaitļošanas viedokļa daudz spēcīgāki nekā Rozenblata perceptrons, piemēram daudzslāņu perceptrons un radiālo bāzes funkciju tīkls.

P7. TEORĒTISKĀ BĀZE.

Šajā nodaļā aprakstīts perceptrona apmācības metodes teorētiskais pamatojums, kas atbilst formulām P4 un P4b.

Perceptrona apmācības algoritma pamatā gradienta metodes pielietojums kļūdas funkcijai.

Par skalāra lauka u **gradientu** fiksētā punktā M_0 sauc vektoru, kas nosaka virzienu, kādā skalārā lieluma vērtība palielinās visātrāk. Gradienta koordinātas aprēķina, izmantojot funkcijas u parciālos atvasinājumus punktā M_0.

Ja u = u(x;y;z), tad gradients ir $\nabla u = \left[\frac{\partial u}{\partial x}; \frac{\partial u}{\partial y}; \frac{\partial u}{\partial z} \right]^T$

Skalāra lauka gradients jebkurā apgabala punktā ir perpendikulārs līmeņlīnijas/līmeņvirsmas pieskarei (resp., iet pa normāli).

Pielietojot gradientu metodi perceptrona apmācībā:

1. Par skalārā lauka funkciju ņemsim kļūdas funkciju E_j (formula P6a), par argumentiem ņemot neirona svarus $W_j=\{w_{j0}, w_{j1}, ..., w_{jm}\}$.
2. Tā kā mums interesē virziens, kurā skalārā lieluma (kļūdas funkcijas) vērtība samazinās, nevis palielinās, gradientu jāņem ar mīnus zīmi.

Tādējādi svaru izmaiņu (fiksētam apmācības piemēram) mēs varam noformulēt šādi:

$$\Delta W_j \equiv -\nabla E_j(W_j) = -\frac{\partial E_j}{\partial W_j} = -\left[\frac{\partial E_j}{\partial w_{j0}}; \frac{\partial E_j}{\partial w_{j1}}; ...; \frac{\partial E_j}{\partial w_{jm}}\right] \tag{P12}$$

kur E_j *– j-tā neirona kvadrātiskā kļūda,*

W_j *– j-tā neirona svaru vērtības,*

0.5 – koeficients ar tehnisku nozīmi – aprēķinu vienkāršošanai,

w_{ji} *– j-tā neirona i-tais svars.*

Konkretizējot formulu P12, viena svara izmaiņa izskatās šādi:

$$\Delta w_{ji} \equiv -\frac{\partial E_j}{\partial w_{ji}} \tag{P12a}$$

Izteiksim formulu P12 kā vienādību, izmantojot konstanti η (eta):

$$\Delta w_{ji} \equiv -\eta \frac{\partial E_j}{\partial w_{ji}} \tag{P12b}$$

Izteiksim formulas P12a atvasinājumu izvērstā formā – 3 komponenšu reizinājumā:

$$\frac{\partial E_j}{\partial w_{ji}} = \frac{\partial E_j}{\partial o_j} \cdot \frac{\partial o_j}{\partial NET_j} \cdot \frac{\partial NET_j}{\partial w_{ji}} \tag{P13}$$

Izmantojot formulas P3 un P12, aprēķināsim formulas P13 pirmo komponenti.

$$\frac{\partial E_j}{\partial o_j} = \frac{\partial E_j}{\partial e_j} \cdot \frac{\partial e_j}{\partial o_j} = e \cdot \frac{\partial (d_j - o_j)}{\partial o_j} = (d_j - o_j) \cdot (-1) = -(d_j - o_j) \tag{P14}$$

Aprēķināsim formulas P13 otro komponenti.

$$\frac{\partial o_j}{\partial NET_j} = \frac{\partial f_{act}(NET_j)}{\partial NET_j} = f'_{act} \tag{P15}$$

Formula P13 tiek precizēta, atkarībā no pielietotās aktivizācijas funkcijas.

Ja tiek izmantota sigmoidālā aktivizācijas funkcija (P2a), formula P15 tiek precizēta šādi:

$$\frac{\partial o_j}{\partial NET_j} = f'_{log2} = h f_{log2}(1 - f_{log2}) = h o_j (1 - o_j) \qquad \text{(P15a)}$$

Ja tiek izmantota lineārā aktivizācijas funkcija (P2b), formula P15 tiek precizēta šādi:

$$\frac{\partial o_j}{\partial NET_j} = f'_{lin} = 1 \qquad \text{(P15a)}$$

Šī formula der arī aktivizācijas funkcijai P2c.

Tagad, izmantojot formulu P1a, aprēķināsim formulas P13 trešo komponenti.

$$\frac{\partial NET_j}{\partial w_{ji}} = \frac{\partial (w_{j0} o_0 + ... + w_{ji} o_i + ... + w_{jm} o_m)}{\partial w_{ji}} = o_i \qquad \text{(P16)}$$

Precizējot formulu P12b ar formulām P14, P15a un P16, iegūstam svaru korekcijas formulu P4 (sigmoidālai aktivizācijas funkcijai), bet, ja precizējam ar formulām P14, P15b un P16, tad iegūstam svaru korekcijas formulu P4b (lineārai aktivizācijas funkcijai).