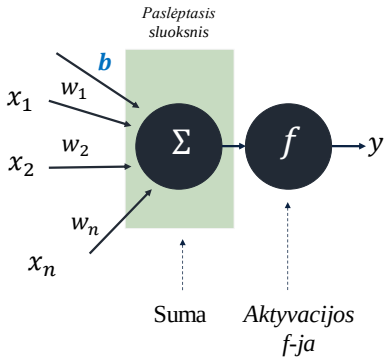
Dirbtiniai neuroniniai tinklai

Turinys

1 Kas yra dirbtiniai neuroniniai tinklai ir kaip jie veikia

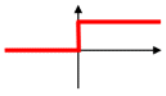
Neuronas – tai nervinė ląstelė, turinti kūną su branduoliu ir ataugas, kurios išsišakojusios kaip medžiai ir liečiasi su tokiomis pačiomis šalimais esančių ląstelių ataugomis sudarydamos milžinišką neuroninį tinklą, kuris ir yra mūsų smegenys. Kompiuterijoje dirbtinis neuronas – tai biologinio neurono abstrakcija. Tarpusavyje sujungtus dirbtinius neuronus vadiname **dirbtiniu neuroniniu tinklu** (**DNT**). Nesigilinant į neuromokslą ir biologiją, dirbtiniai neuroniniai tinklai gali būti apibūdinami kaip matematinė funkcija *f*, kuri atvaizduoja duotą įėjimą į pageidaujamą išėjimą. **DNT** sudaro tokie komponentai (1 pav.)

1. įvesčių aibė $\boldsymbol{X = \{x_1, x_2, \ldots, x_n\}}$;
2. tam tikras kiekis paslėptųjų neuronų sluoksnių;
3. neurono išvestis ***y***;
4. aibės svorių ***W*** ir poslinkio reikšmių ***b*** tarp kiekvieno sluoksnio;
5. aktyvinimo funkcijos ***f*** kiekviename paslėptajame sluoksnyje.



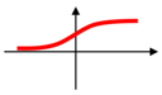
1 pav. *Dirbtinio neurono struktūra*

Poslinkio reikšmė (angl. *bias*) – tai yra papildomas neuroninio tinklo parametras, naudojamas išėjimui reguliuoti. Poslinkio reikšmė yra fiksuota ir paprastai b = 1. Šios reikšmės įtraukimas yra svarbus norint išgauti lankstesnius rezultatus. Be to, ji leidžia pastumti aktyvinimo funkciją į kairę arba dešinę. Pagal sužadinimo signalą naudojant aktyvinimo funkciją skaičiuojama neurono išėjimo reikšmė. Dirbtinių neuronų tinkluose gali būti naudojamos įvairios aktyvinimo funkcijos. Kokio sudėtingumo aktyvinimo funkcija taikoma, priklauso nuo sprendžiamo uždavinio. Dažniausiai yra taikomas loginis sigmoidas. Taip pat ganėtinai dažnai naudojamos ir tiesinė, slenksčio, hiberbolinio tangento funkcijos (2 pav.).



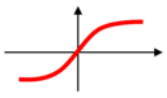
$$f(x) = \begin{cases} 0, & jei\ x < 0 \\ 1, & jei\ x \geq 0 \end{cases}$$

(a) Slenkstinė (šuolinė)



$$f(x) = \frac{1}{1 + e^{-x}}$$

(b) Loginio sigmoido

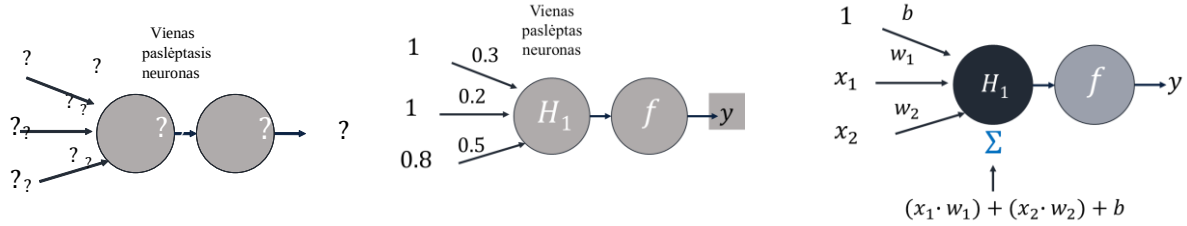


$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

(c) Hiperbolinio tangento

2 pav. *Aktyvinimo funkcijos*

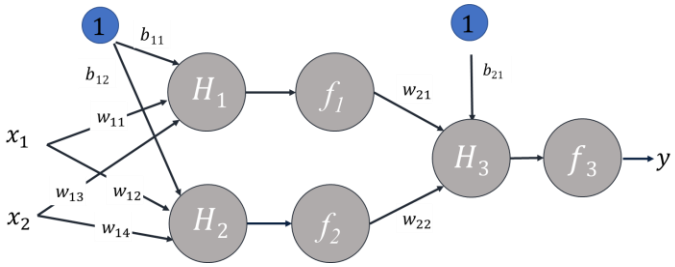
Paimkime patį paprasčiausią pavyzdį su vienu paslėptuoju neuronu H_1, dviem įvesties kintamaisiais x_1, x_2, poslinkio reikšme b ir vienu išvesties kintamuoju y (3 pav. (a)) Neuroninis tinklas paprasčiausiu atveju veikia kaip funkcija, atvaizduojanti įvesties reikšmes į išvesties reikšmę ar reikšmes, nes įvesčių taip pat gali būti keletas. Norėdami suprasti, kaip apskaičiuojama išvesties reikšmė y, parinkime tinklo parametrams konkrečias reikšmes. Tarkime, $x_1 = 1$; x_2= 0,8 ir poslinkis =1. Kaip matyti iš 3 paveikslėlio, kiekviena iš šių įvesčių turi svorius b, w_1, w_2. Tarkime, svorių reikšmės yra w_1,= 0,2; w_2,= 0,5; b = 0,3 (3 pav. (b)). Tokiu atveju neurono H_1 reikšmė apskaičiuojama, kaip į jį ateinančių svorių ir įvesčių reikšmių (įskaitant ir poslinkio reikšmę, jeigu tokia yra) sandaugų suma $(x_1 \cdot w_1) + (x_2 \cdot w_2) + b = (1 \cdot 0{,}2) + (0{,}8 \cdot 0{,}5) + (0{,}3 \cdot 1)$ (3 pav. (c)). Susumavus gauname $H_1 = 0{,}9$. Ši reikšmė paduodama aktyvinimo funkcijai f (0,9). Išvesties reikšmė y priklauso nuo to, kokia bus pasirinkta aktyvinimo funkcija. Tarkime, mes panaudosime sigmoido funkciją $f(x) = \frac{1}{1+e^{-x}} =$ $f(x) = \frac{1}{1+e^{-0{,}9}} = 0{,}71$, todėl $y =$ 0,71.



(a) (b) (c)

3 pav. *Neuroninio tinklo su vienu paslėptuoju neuronu struktūra (a), įvesčių ir svorių reikšmės (b) ir paslėptojo neurono skaičiavimo formulė (c)*

Paimkime šiek tiek sudėtingesnį pavyzdį, kur neuroninį tinklą sudaro trys neuronai H_1, H_2, H_3 (4 pav.). Kaip matyti iš paveikslėlio, čia atsiranda du sluoksniai neuronų, todėl turime ir daugiau tinklo parametrų: w_{11}= 0.2; w_{12}= 0,5; $w_{13} = 0{,}3$, $w_{14} = 0{,}8$ $w_{21} = 0{,}4$, $w_{22} = 0{,}5$, $b_{11} = 0{,}3$, $b_{12} = 0{,}3$, $b_{21} = 0{,}5$.



4 pav. *Kelių sluoksnių neuroninis tinklas*

Turint daugiau nei vieną sluoksnį, skaičiavimus reikia pradėti nuo pirmojo sluoksnio ir gautas reikšmes paduoti tolimesniems, ir taip tęsti, kol bus pasiektas paskutinis sluoksnis ir gauta y reikšmė. Todėl pateiktam pavyzdžiui pirmiausia paskaičiuojamos H_1 ir H_2 reikšmės:

$$H_1 = (x_1 \cdot w_{11}) + (x_2 \cdot w_{13}) + b_{11} = (1 \cdot 0{,}2) + (0{,}8 \cdot 0{,}3) + 0{,}3 = 0{,}74;$$

$$H_2 = (x_1 \cdot w_{12}) + (x_2 \cdot w_{14}) + b_{12} = (1 \cdot 0{,}5) + (0{,}8 \cdot 0{,}8) + 0{,}3 = 1{,}44.$$

Šios apskaičiuotos neuronų reikšmės paduodamos į atitinkamas aktyvinimo funkcijas f_1 ir f_2. Tarkime, tai bus slenkstinė funkcija, kur išėjimas gali įgyti reikšmę 0 arba 1 (2 pav. (a)). Atlikus veiksmus gauname tokias reikšmes:

$f_1\ (0{,}74) = 1;$

$f_2\ (1{,}44) = 1.$

Kitu etapu yra skaičiuojama neurono H_3 reikšmė. Į H_3 neuroną kaip įvesčių reikšmės yra paduodamos apskaičiuotos reikšmės iš funkcijų $f_1\ ir\ f_2$ bei svoriai w_{21} ir w_{22}:

$H_3 = (w_{21} \cdot 1) + (w_{22} \cdot 1) + b_{21} = (0{,}4 \cdot 1) + (0{,}5 \cdot 1) + 0{,}5 = 1{,}4.$

Tarkime, f_3 aktyvinimo funkcija yra loginis sigmoidas:

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{1}{1 + e^{-1.4}} = 0{,}8.$$

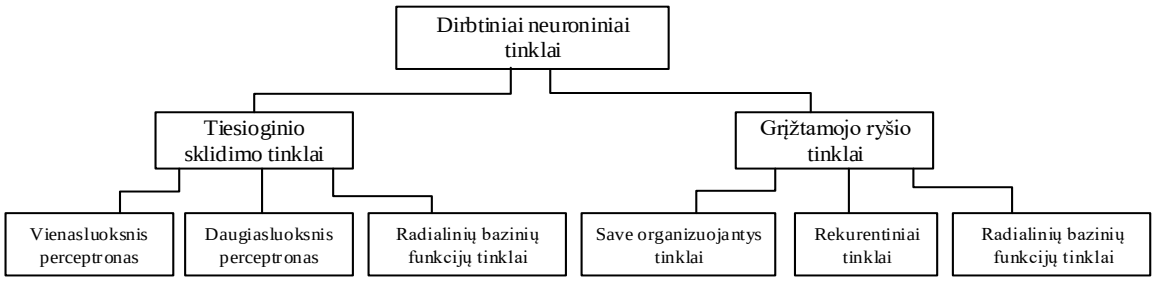
Gauta 0,8 reikšmė yra lygi išvesties *y* reikšmei, todėl $y = 0{,}8.$

Akivaizdu, kad abiejuose pavyzdžiuose visi skaičiavimai atliekami siekiant apskaičiuoti išvesties y reikšmę. Tačiau kyla klausimas, koks tokių skaičiavimų tikslas, ką daryti su šia reikšme, ar ji gerai apskaičiuota ir pan. Tai pirmiausia reikėtų apsibrėžti, kokia yra pageidaujama *y* reikšmė ir kada ji laikoma teisinga ir neteisinga. DNT tikslas surasti tokius svorius, kurie leistų išgauti norimą išvesties reikšmę. Ir todėl esminis DNT mokymosi procesas pagrįstas tinkamiausių svorių paieška.

1.1 Dirbtinių neuroninių tinklų klasifikacija

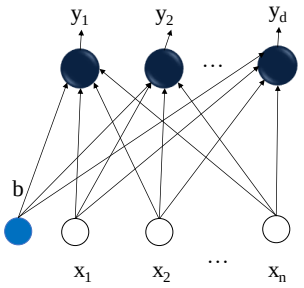
Jungiant vienus dirbtinius neuronus su kitais yra konstruojamas dirbtinis neuroninis tinklas. Kiekvienas neuroninis tinklas turi įėjimų, išėjimų ir paslėptųjų neuronų sluoksnius. Įėjimo, paslėptieji ir išėjimo neuronai yra jungiami tarpusavyje. Vienų neuronų išėjimo reikšmės atitinka kitų neuronų įėjimo reikšmes. Pagal neuronų jungimo konstrukciją dirbtiniai neuroniniai tinklai gali būti skirstomi į dvi grupes (5 pav.):

1. tiesioginio sklidimo tinklai;
2. grįžtamojo ryšio (arba rekurentiniai) tinklai.

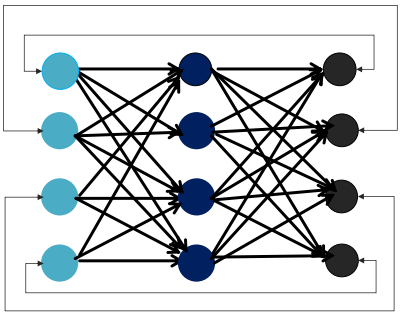


5 pav. *Kelių sluoksnių neuroninis tinklas*

Tiesioginio sklidimo neuroniniai tinklai – tai tinklai, kuriuose galimos tik vienkryptės jungtys. Tiesioginio sklidimo neuroniniuose tinkluose egzistuoja tik tiesioginiai sujungimai tarp tinklo viršūnių (neuronų, įėjimų ir išėjimų) ir visa informacija tinkle juda viena kryptimi į priekį nuo įvesties sluoksnio per paslėptuosius sluoksnius link išvesties sluoksnio. Paprasčiausias tokio tipo neuroninis tinklas – **perceptronas**, susidedantis iš vieno sluoksnio *d* neuronų, sujungtų su *n* įėjimais. Neuronų skaičius *d* lygus išėjimų skaičiui (6 pav).



6 pav. *Tiesioginio sklidimo neuroninis tinkas – perceptronas*



7 pav. *Jordano grįžtamojo ryšio neuroninis tinklas*

Grįžtamojo ryšio neuroniniai tinklai − tai sudėtingesni tinklai, kurie skirtingai nuo tiesioginio sklidimo tinklų gali turėti tiek vienkrypčius, tiek dvikrypčius sujungimus, ir būtent pastarųjų dėka signalai gali sklisti visomis kryptimis. Tokio tinklo pavyzdys gali būti − uždarojo ciklo *Jordano* neuroninis tinklas, kuriame išėjimas grįžta atgal į įėjimą kaip grįžtamasis ryšys (7 pav.).

Rekurentiniai tinklai − tai grįžtamojo ryšio tinklai su uždaraisiais ciklais. **Pilnai rekurentiniai tinklai** − tai viena paprasčiausių neuroninio tinklo struktūrų, todėl kad visos tinkle viršūnės yra sujungtos tarpusavyje ir kiekviena viršūnė veikia ir kaip įvestis, ir kaip išvestis.

Nuo tinklo architektūros priklauso ir tinklo mokymo algoritmo parinkimas. Galima išskirti dvi pagrindines neuronų mokymo paradigmas:

- prižiūrimojo mokymosi algoritmai → prižiūrimojo mokymosi DNT;
- neprižiūrimojo mokymosi algoritmai → neprižiūrimojo mokymosi DNT.

1.2 Prižiūrimojo mokymosi neuroniniai tinklai

Prižiūrimojo mokymosi DNT − tai neuroniniai tinklai, kurie mokymuisi naudoja duomenis su iš anksto žinomomis išvesčių reikšmėmis (etiketėmis), todėl mokymosi metu siekiama surasti svorius, kurie leistų minimizuoti skirtumą tarp gautų rezultatų ir „tikrųjų“ išvesčių. Kaip ir visuose prižiūrimojo mokymo algoritmuose bendra duomenų imtis yra padalijama į mokymo ir testavimo poaibius (8 pav. (a)). Kartais yra išskiriamas ir validavimo poaibis. Standartiškai mokymo poaibis sudaro nuo 60 iki 80 % visų duomenų, testavimo poaibis nuo 20 iki 40 %, o jeigu reikia išskirti validavimo poaibį, tai dažniausiai šis sudaro nuo 10 iki 20 % visų duomenų (8 pav. (b)).

(a) (b)

8 pav. *Bendro duomenų rinkinio išskaidymas į (a) dvi dalis: mokymo ir testavimo ir (b) tris dalis: mokymo, testavimo ir validavimo*

Mokymo poaibis naudojamas modeliui mokyti, o testavimo poaibis tai nauji ir išmokytam modeliui nematyti duomenys, su kuriais tikrinama, kaip gerai modelis išmoko. Validavimo poaibis leidžia patikrinti modelio tinkamumą, ir jeigu validavimo paklaida kyla, tai galima traktuoti, kad modelis persimoko.

1.2.1 Mokymasis ir testavimas

Pagrindiniai DNT mokymosi žingsniai su mokymo duomenų imtimi:

1. inicijuoti svorius (dažniausiai atsitiktinai);
2. dauginti svorius *w* su įvesčių reikšmėmis *x* ir juos sumuoti;
3. paskaičiuoti išvesties y reikšmes;
4. lyginti gautus DNT rezultatus *y* su „tikraisiais“ rezultatais *t*;
5. atnaujinti svorius *w*;
6. kartoti, kol bus pasiektas norimas tikslumas.

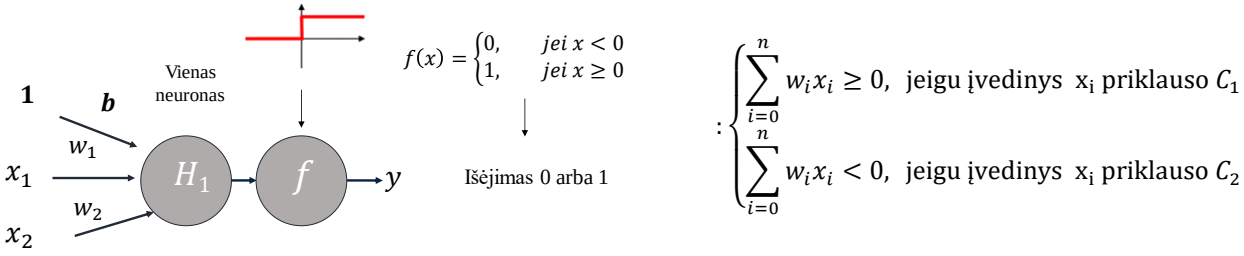
Išmokytas tinklas yra testuojamas su naujais duomenimis. Pagrindiniai DNT testavimo žingsniai:

1. išmokytą tinklą panaudoti testavimo duomenims;
2. įvertinti gautus rezultatus;
3. spręsti persimokymo / nepakankamo mokymosi problemas (*jeigu tokios egzistuoja*).

Modelio rezultatams įvertinti dažnai naudojama **kryžminė validacija**. Kryžminė validacija – tai statistinis metodas, kuris dalija duomenų rinkinį į tam tikrą skaičių mažesnių rinkinių (poaibių). Šis metodas padeda geriau suprasti, ar modelis tinkamai veikia su skirtingais duomenų rinkiniais ar vis tik jautrus duomenims. Tai pat leidžia išvengti persimokymo. Kryžminės validacijos metodų yra įvairių, tačiau vienas populiariausių tai ***k*-kartų** kryžminė validacija, kuri visą duomenų rinkinį padalija į *k* vienodo dydžio rinkinių. Vienas rinkinys naudojamas testuoti, visi kiti mokytis. Tolimesniu etapu keičiamas testavimo rinkinys (mokymo taip pat), ir taip tęsiamas procesas, kol testavimas bus atliktas su visais galimais rinkiniais.

1.2.2 Sprendimo pavyzdys – vienasluoksnis perceptronas

Vienasluoksnis perceptronas (angl. *single layer perceptron*) – tai yra vieno neurono tinklas. Vienasluoksnio perceptrono uždavinys – suklasifikuoti duomenis į dvi klases C_1 ir C_2 (9 pav.).



9 pav. *Vienasluoksnio perceptrono uždavinys*

Perceptrono mokymosi žingsniai:

1. inicijuoti svorius *w* (atsitiktinai);
2. dauginti svorius *w* su įėjimo reikšmėmis *x* ir juos sumuoti (pridėti *b* reikšmę);
3. panaudoti šuolinę funkciją y reikšmei (0 arba 1) paskaičiuoti;
4. lyginti gautus *y* rezultatus su „tikraisiais“ t;
5. atnaujinti svorius;
6. kartoti 2-5 žingsnius, kol perceptronas bus išmokytas.

Kaip mokymosi duomenų pavyzdį paimkime vaikinų atrankos į krepšinio komandą 3 duomenų pavyzdžius (1 lent.). Pagal krepšininko *ūgį* ir *svorį* yra sprendžiama, ar priimti krepšininką į komandą. Dėl patogumo įvesties

kintamuosius pervardinkite: *ūgį* į *x1 ir svorį* į *x2*. Išvesties kintamasis *atsakymas* prilyginamas *t* (tai „tikrieji“ arba tikslo kintamasis). Visas reikšmes normalizuokime intervale [-1;1] (2 lent.). Normalizacija yra dažnai naudojama sprendžiant DNT uždavinius dėl įvesčių kintamųjų skirtingo reikšmių intervalo.

1 lentelė. *Pradinis mokymosi duomenų rinkinys*

Ūgis		Svoris	Atsakymas
195		91	Priimtas
211		114	Priimtas
188		101	Nepriimtas

2 lentelė. *Pradinis mokymosi duomenų rinkinys*

x1	x2	***t***
–0,39	–1	1
1	1	1
–1	–0,13	0

1. Pirmasis žingsnis – atsitiktinai inicijuojami svoriai:

$$\begin{bmatrix} w_o \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0{,}9 \\ 0{,}7 \\ 0{,}4 \end{bmatrix}$$

2. Dauginti svorius *w* su įėjimo reikšmėmis *x* ir juos sumuoti (pridėti *b* reikšmę):

$$0{,}9 \cdot 1 + (-0{,}39 \cdot 0{,}7) + (-1 \cdot 0{,}4) = \ 0{,}227$$

$$0{,}9 \cdot 1 + (1 \cdot 0{,}7) + (1 \cdot 0{,}4) = \ 2$$

$$0{,}9 \cdot 1 + (-1 \cdot 0{,}7) + (-0{,}13 \cdot 0{,}4) = \ 0{,}148.$$

3. Panaudoti šuolinę funkciją f išvesties y reikšmei paskaičiuoti (0 arba 1):

$$f(x) = \begin{cases} 0, & jei\ x < 0 \\ 1, & jei\ x \geq 0 \end{cases}; \quad y_1 = 1; \quad y_2 = 1; y_3 = 1;$$

4. Lyginti gautus y rezultatus su „tikraisiais“ t:

y	***t***	
1	1	☑
1	1	☑
1	0	☒

tikrasis → t ; *gautasis* → y

$$i = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

5. Atnaujinti perceptrono svorius pagal formulę:

$$w_j^{n+1} = w_j^n + \eta(y_i - t_i)x_j^{(i)},$$

čia w_j nauja svorio reikšmė, *n+1* sekančios iteracijos numeris, w_j^n dabartinė svorio reikšmė, η mokymosi žingsnis.

Svorių atnaujinimas. 1 epocha, 1 iteracija

Atnaujinti perceptrono svorius w_j^{n+1} pirmajam duomenų rinkiniui (*i*-tajai eilutei). Kadangi turime tik tris pavyzdžius, tai pirmoji epocha bus vykdoma tris kartus. Pradiniu laiko momentu n = 1 svorių reikšmės yra lygios:

$$\begin{bmatrix} w_o \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0{,}9 \\ 0{,}7 \\ 0{,}4 \end{bmatrix}$$

Tikriname, ar pirmojo duomenų rinkinio *t* reikšmė yra lygi apskaičiuotai *y* reikšmei:

$$\begin{matrix} x_0 & x_1 & x_2 \end{matrix} \quad \begin{matrix} i= & t & y \end{matrix}$$
$$\begin{bmatrix} 1 & -0.39 & -1 \\ 1 & 1 & 1 \\ 1 & -1 & -0.13 \end{bmatrix} \quad \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Kaip matyti, kai $i = 1$, tai $t_i = y_i$. Todėl perskaičiuojant svorius šie nepakinta:

$$w_j^{n+1} = w_j^n + \eta(y_i - t_i)x_j^{(i)};$$
$$w_0^2 = 0{,}9 + 0{,}05(1-1)1 = 0{,}9;$$
$$w_1^2 = 0{,}7 + 0{,}05(1-1)(-0{,}39) = 0{,}7;$$
$$w_2^2 = 0{,}4 + 0{,}05(1-1)(-1) = 0{,}4.$$

Svorių atnaujinimas. 1 epocha, 2 iteracija

Tikriname, ar antrojo duomenų rinkinio *t* reikšmė yra lygi *y* reikšmei:

$$\begin{matrix} x_0 & x_1 & x_2 \end{matrix} \quad \begin{matrix} i= & t & y \end{matrix}$$
$$\begin{bmatrix} 1 & -0.39 & -1 \\ 1 & 1 & 1 \\ 1 & -1 & -0.13 \end{bmatrix} \quad \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Kai $i = 2$, tai $t_i = y_i$, ir perskaičiuojant svorius šie taip pat nepakinta lyginant su pradiniais:

$$w_0^2 = 0{,}9 + 0{,}05(1-1)1 = 0{,}9$$
$$w_1^2 = 0{,}7 + 0{,}05(1-1)1 = 0{,}7$$
$$w_2^2 = 0{,}4 + 0{,}05(1-1)1 = 0{,}4.$$

Svorių atnaujinimas. 1 epocha, 3 iteracija

Tikriname, ar trečioji duomenų rinkinio *t* reikšmė yra lygi *y* reikšmei:

$$\begin{matrix} x_0 & x_1 & x_2 \end{matrix} \quad \begin{matrix} i= & t & y \end{matrix}$$
$$\begin{bmatrix} 1 & -0.39 & -1 \\ 1 & 1 & 1 \\ 1 & -1 & -0.13 \end{bmatrix} \quad \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Kai $i = 3$, tai $t_i \neq y_{i,}$ ir perskaičiuojant svorius šie nežymiai pasikeičia:

$w_0^2 = 0{,}9 + 0{,}05(0-1)1 = 0{,}85$

$w_1^2 = 0{,}7 + 0{,}05(0-1)(-1) = 0{,}75$

$w_2^2 = 0{,}4 + 0{,}05(0-1)(-0{,}13) = 0{,}4.$

Po šios epochos matome, kad svoriai šiek tiek pakito ir su šiais pakitusiai svoriais yra perskaičiuojamos išvesties reikšmės *y:*

n=1 n=2

$$\begin{bmatrix} w_o \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.8 \\ 0.4 \end{bmatrix} \Rightarrow \begin{bmatrix} w_o \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.85 \\ 0.75 \\ 0.4 \end{bmatrix}$$

$0{,}85 \cdot 1 + (-0{,}39 \cdot 0{,}75) + (-1 \cdot 0{,}4) = 0{,}157,$ todėl $y_1 = 1, y_1 = t_1$

$0{,}85 \cdot 1 + (1 \cdot 0{,}75) + (1 \cdot 0{,}4) = 2,$ todėl $y_2 = 1, y_2 = t_2$

$0{,}85 \cdot 1 + (-1 \cdot 0{,}75) + (-0{,}13 \cdot 0{,}4) = 0{,}048,$ todėl $y_3 = 1, y_3 \neq t_3.$

Kadangi y_3 reikšmė nelygi tikrajai reikšmei t_3, todėl mokymasis yra tęsiamas.

Svorių atnaujinimas. 2 epocha, 1 iteracija

Tiek pirmoje, tiek antroje iteracijoje svoriai nekinta, nes paskaičiuotų išvesčių reikšmės yra lygios tikrosioms reikšmėms.

$w_0^2 = 0{,}85 + 0{,}05(1-1)1 = 0{,}85;$

$w_1^2 = 0{,}75 + 0{,}05(1-1)(-0{,}39) = 0{,}75;$

$w_2^2 = 0{,}4 + 0{,}05(1-1)(-1) = 0{,}4.$

Svorių atnaujinimas. 2 epocha, 2 iteracija

$w_0^2 = 0{,}85 + 0{,}05(1-1)1 = 0{,}85;$

$w_1^2 = 0{,}75 + 0{,}05(1-1)1 = 0{,}75;$

$w_2^2 = 0{,}4 + 0{,}05(1-1)1 = 0{,}4.$

Trečioje iteracijoje svoriai atnaujinami, nes $y_3 \neq t_3$

$w_0^2 = 0{,}85 + 0{,}05(0-1)1 = 0{,}8;$

$w_1^2 = 0{,}75 + 0{,}05(0-1)(-1) = 0{,}8;$

$w_2^2 = 0{,}4 + 0{,}05(0-1)(-0{,}13) = 0{,}406.$

Po šios epochos matome, kad svoriai vėl nesmarkiai pakito, ir su naujomis svorių reikšmėmis yra perskaičiuojamos išvesties reikšmės *y:*

n=2 n=3

$$\begin{bmatrix} w_o \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.85 \\ 0.75 \\ 0.4 \end{bmatrix} \Rightarrow \begin{bmatrix} w_o \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.8 \\ 0.4 \end{bmatrix}$$

$0{,}8 \cdot 1 + (-0{,}39 \cdot 0{,}8) + (-1 \cdot 0{,}4) = 0.08$, todėl $y_1 = 1, y_1 = t_1$;

$0{,}8 \cdot 1 + (1 \cdot 0{,}8) + (1 \cdot 0{,}4) = 2$, todėl $y_2 = 1, y_2 = t_2$;

$0{,}8 \cdot 1 + (-1 \cdot 0{,}8) + (-0{,}13 \cdot 0{,}4) = -0{,}05$, todėl $y_3 = 0, y_3 = t_3$.

Gauti rezultatai rodo, kad visos išvesčių reikšmės y_i yra lygios tikrosioms reikšmės t_i, todėl mokymo procesą galima baigti ir pradėti testavimą su naujausiomis tinklo svorių reikšmėmis. Testavimo metu neuroniniam tinklui paduodami nauji ir nematyti duomenų pavyzdžiai:

$$\begin{matrix} x_0 & x_1 & x_2 \end{matrix}$$
$$\begin{bmatrix} 1 & -0.7 & -0.9 \\ 1 & 0.8 & 0.8 \\ 1 & 0.5 & -0.1 \end{bmatrix}$$

t	y
0	?
1	?
1	?

Testavimo rezultatai:

$0{,}8 \cdot 1 + (-0{,}7 \cdot 0{,}8) + (-1 \cdot 0{,}4) = -0{,}12$, todėl $y_1 = 0, y_1 = t_1$;

$0{,}8 \cdot 1 + (1 \cdot 0{,}8) + (1 \cdot 0{,}4) = 1{,}76$, todėl $y_2 = 1, y_2 = t_2$;

$0{,}8 \cdot 1 + (-1 \cdot 0{,}8) + (-0{,}13 \cdot 0{,}4) = 1.16$, todėl $y_1 = 1, y_3 = t_3$.

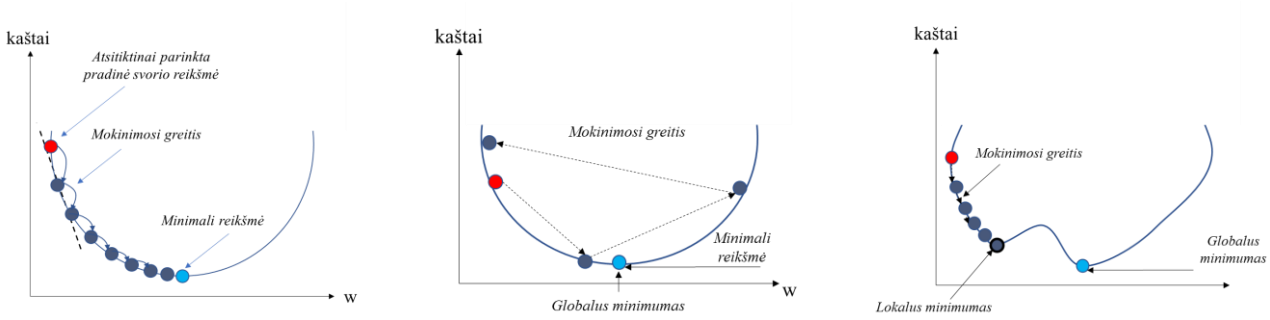
Šiuo atveju testavimo rezultatai yra tokie, kokių buvo siekiama, nes visos reikšmės priskirtos teisingai. Tačiau taip būna ne visada. Pats mokymas (svorių atnaujinimas) gali būti stabdomas remiantis skirtingais kriterijais. Pavyzdžiui, kai tinklo svoriai nebekinta, kai pasiektas nustatytas maksimalus iteracijų ar epochų skaičius, kai bus pasiekta minimali paklaida, kai pradeda augti validacijos paklaida ir pan.

Minimaliai paklaidai tarp gautųjų įvesčių reikšmių y_i ir tikrųjų reikšmių t_i paskaičiuoti dažnai naudojama suminė kvadratinė paklaida:

$$E(W) = \frac{1}{2}\sum_{i=1}^{n}(y_i - t_i)^2.$$

Kuo mažesnė paklaida, tuo tinklas geriau išmokytas.

Perceptrono mokymo algoritmas vykdomas iteracijomis, todėl paklaidą gali paskaičiuoti po kiekvienos užduoties atlikimo. Žinant iteracijos paklaidą, perceptrono svorius galima keisti pažingsniui. Su kiekviena kita iteracija perceptrono svoriai yra keičiami į reikšmę, mažinančią paklaidą. Jeigu paklaidos funkcija $E(W)$ yra diferencijuojama pagal svorius, jos minimumas gali būti randamas gradientiniais optimizavimo metodais.



(a) (b) (c)

10 pav. *Perceptrono mokymas gradientinio nusileidimo metodu: gradientinio nusileidimo principas; (b) per didelio mokymo greičio atvaizdavimas; (c) per mažo mokymo greičio atvaizdavimas*

Gradientinio nusileidimo principas – leidimas „kalnu žemyn“ iki tol, kol bus rastas lokalus arba globalus minimumas (10 pav. (a)). Žingsnio dydį nusako mokymosi greitis (angl. *rate*), taip pat, kaip ir gradiento nuolydis. Tokio mokymosi tikslas – keičiant tinklo svorius įvertinti bendrą tinklo paklaidą ir jos kitimą. Kiekvienas atnaujinimas modifikuoja reikšmes priešinga gradiento kryptimi:

$$w_j^{n+1} = -\eta \nabla E(w).$$

Todėl reikia paskaičiuoti kiekvieno svorio nuostolių funkcijos dalinę išvestinę:

$$w_j^{n+1} = -\eta \frac{\partial E}{\partial w_j};$$

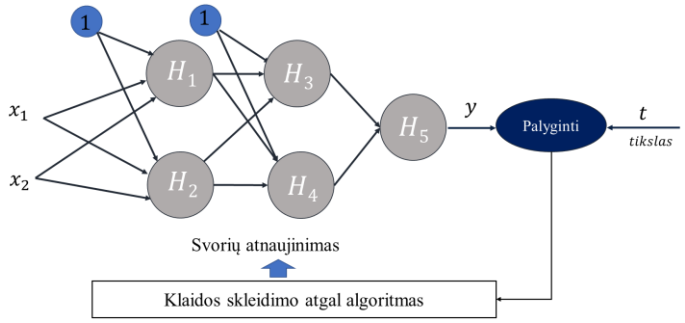
$$w_j^{n+1} = -\eta \frac{\partial E}{\partial w_j} = \eta \sum_i (t_i - y_i)(x_{ij}).$$

Mokymosi parinkimo greitis yra atsakingas žingsnis, nes jeigu greitis yra **per didelis**, tai gradientinis nusileidimas prašoks minimumą ir diverguos (10 pav. (b)). Jeigu mokymo greitis **per mažas**, tai sprendimas bus ilgas ir reikalaus daug epochų, ir tikėtina, kad sprendimas užstrigs ties lokaliu minimumu (10 pav. (c)).

Nors ir gradientiniu nusileidimu grįstas mokymas atrodo identiškas perceptrono mokymosi taisyklei, vis tik keli skirtumai gali būti išskirti:

1. gradientinio nusileidimo mokymosi algoritme išvesties *y* reikšmė yra tikra reikšmė, o ne klasės etiketė (pvz., 0 arba 1) kaip perceptrono mokymosi taisyklėje.
2. svoriai yra atnaujinami remiantis visais mokymosi rinkinio duomenimis (o ne atnaujinant svorius palaipsniui po kiekvieno pavyzdžio).

Vienasluoksniame neuroniniame tinkle mokymosi algoritmas remiasi tinklo svorių reguliavimu minimizuojant klaidą išėjimo sluoksnyje. Kai neuroninis tinklas turi vieną neuronų sluoksnį, tai mokymosi algoritmas aiškus, nes išėjimo sluoksnio teisingi atsakymai yra žinomi. Tačiau vienasluoksnio tinklo mokymosi būdas nėra tinkamas daugiasluoksnio tinklo mokymuisi, kur nėra žinomos paslėptųjų sluoksnių išėjimo reikšmės. Galimas sprendimo būdas taikyti atgalinio klaidos sklidimo algoritmą, kuris „paskirsto“ klaidą tarp neuroninio tinklo elementų pradedant nuo išėjimo sluoksnio tolyn prie įėjimo sluoksnio (11 pav.).



11 pav. *Atgalinis klaidos sklidimas daugiasluoksnio perceptrono tinkle*

1.3 Neprižiūrimojo mokymosi neuroniniai tinklai

Neprižiūrimojo mokymosi DNT – naudoja duomenis ir sprendžia uždavinius, kuriuose nėra žinoma išvestis, todėl gautų tinklo rezultatų nėra su kuo palyginti. Tokio mokymosi tikslas surasti parametrus, kuriais remiantis pavyktų klasterizuoti duomenis į tam tikrą klasterių (klasių) skaičių.

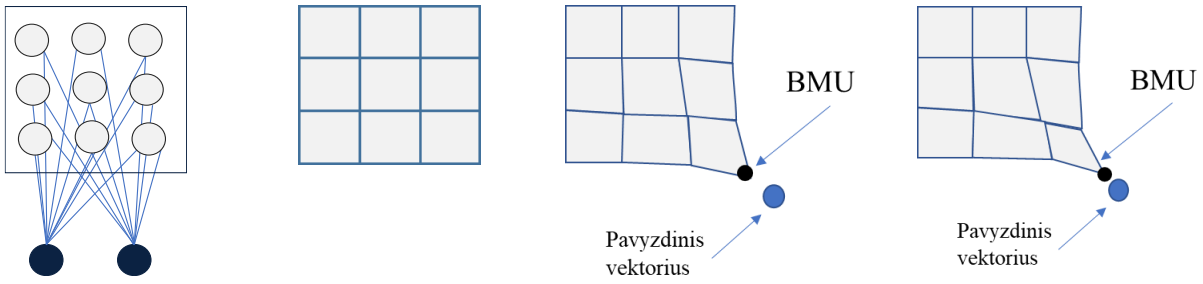
Dažnai toks mokymasis yra pagrįstas konkurenciniu mokymu, kuriame dėl aktyvinimo tarpusavyje konkuruoja išėjimo neuronai. Vienu metu gali būti aktyvinamas tik vienas neuronas. Šis aktyvintas neuronas yra vadinamas laimėtoju (angl. *winning neuron*). Toks konkuravimas gali būti realizuotas tarp neuronų turint šoninius slopinimo ryšius. Dėl to visi neuronai yra priverčiami save organizuoti, todėl ir tinklas vadinamas **save organizuojančiu tinklu** (SOM). Save organizuojantys neuroniniai tinklai **(žemėlapiai)** yra vienas žinomiausių neprižiūrimojo mokymosi neuroninių tinklų.

Save organizuojantys neuroniniai tinklai dar yra vadinami *Kohoneno* neuroniniais tinklais arba *Kohoneno* save organizuojančiais žemėlapiais. Pagrindinis SOM tinklo tikslas – išlaikyti duomenų topologiją. Taškai, esantys arti įėjimo, reikšmių vektoriuje yra atvaizduojami šalia vienas kito ir SOM tinkle. Tokia išdėstymo forma yra žinoma kaip topografinis žemėlapis (12 pav. (a)). Neuronai, dirbantys su labai panašia informacija, yra išdėstomi greta vienas kito, kad galėtų sąveikauti naudodami trumpus sinapsių susijungimus.

SOM tinklai gali būti naudojami kaip vizualizacijos metodas, kuris padeda suprasti daugiamačius duomenis juos atvaizduojant į mažesnio skaičiaus matmenų erdvę – duomenų žemėlapį. SOM taip pat gali būti naudojamas klasterizavimo uždaviniams duomenis sugrupuojant pagal panašumą.

1.3.1 SOM algoritmas

Norint spręsti problemą SOM tinklais, pirmiausia sukuriamas SOM žemėlapis ir sugeneruojamas pirminis svorių vektorius (atsitiktinai). Iš čia yra parenkamas pavyzdinis vektorius (taip pat atsitiktinai) ir tada kiekviena žemėlapio viršūnė – neuronas tikrina, kurio iš jų svoriai yra panašiausi į pavyzdinį svorių vektorių (12 pav. (b)). Reiktų įsiminti, kad visi neuronai tarpusavyje konkuruoja. Laimėjęs neuronas yra vadinamas BMU (angl. *Best Matching Unit*). Kitame etape yra skaičiuojamas BMU kaimynystės spindulys. Iš pradžių spindulys būna didesnis, o paskui su kiekviena iteracija mažėja. Visi neuronai pažymėti kaip kaimynai yra „apdovanojami“, t. y. kiekvienos kaimyninės viršūnės svoriai modifikuojami siekiant padidinti jų panašumą į pavyzdinį vektorių. Kuo viršūnė yra arčiau BMU, tuo labiau keičiami jos svoriai, tuo labiau „apdovanojami“. Šis žingsnis kartojamas *n* kartų, kol pasiekiamas norimas rezultatas. Paprastai tai apima daugiau nei 1000 iteracijų.



(a) (b)

12 pav. *SOM tinklas (a) ir grafinė SOM algoritmo veikimo reprezentacija (b)*