Kapitola 4

Perceptrón a viacvrstvový perceptrón

TÉMY

- ***Perceptrón (jednovrstvový)***
  - *algoritmus učenia*
  - *lineárna a nelineárna separovateľnosť*

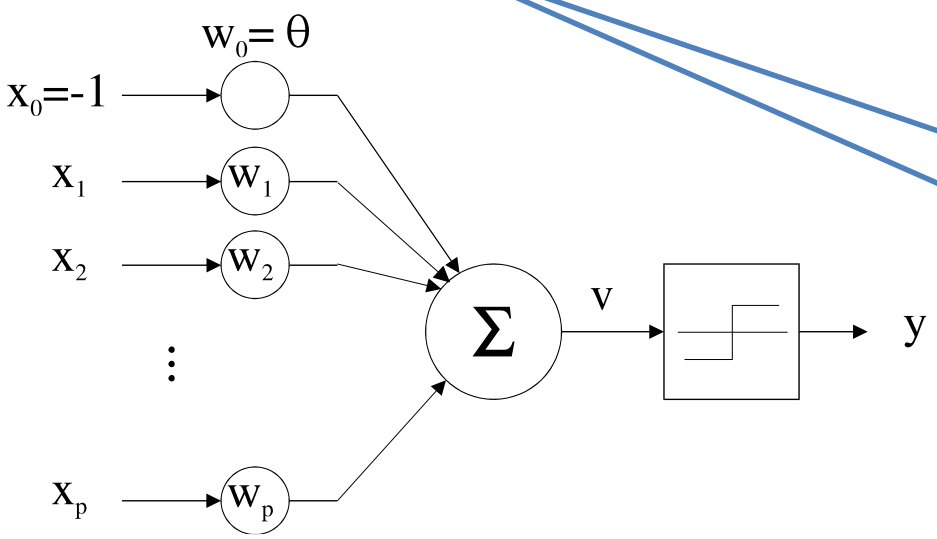
- ***Viacvrstvový perceptrón (MLP, multilayer perceptron)***
  - *algoritmus spätného šírenia chýb (backpropagation algorithm, error-backpropagation algorithm, BP alg.)*
    - *odvodenie algoritmu pre výstupný neurón*
    - *odvodenie algoritmu pre skrytý neurón*
  - *vysvetlenie spätného šírenia chýb*
  - *sigmoidálna nelinearita*
  - *rýchlosť učenia*
  - *momentová technika*
  - *vzorkový a dávkový mód*
  - *sumarizácia algoritmu spätného šírenia*

- *tipy pre lepšiu činnosť BP algoritmu*
- *generalizácia*
- *aproximácia funkcií*
- *univerzálna aproximačná teoréma*
- *univerzálna aproximácia*
- *MLP s viacerými skrytými vrstvami*
- *teorémy o aproximačných schopnostiach MLP*
- *návrh MLP*
    - *napr. veľkosť trénovacej množiny pre binárny klasifikátor*
- *ďalšie spôsoby trénovania MLP*
    - *premenlivá rýchlosť učenia, konjugovaný gradient, Levenberg–Marquardt*

- ***zjednodušovanie sietí - network pruning:***
    - *používa sa aj pre iné siete, nielen pre MLP*

1. *rast siete (network growing)*
    - *kaskádová korelácia (cascade correlation)*
2. *zmenšovanie siete (network pruning)*
    - *útlm váh (weight decay)*
    - *eliminácia váh (weight elimination)*
    - *optimal brain damage*
3. *iné*
    - *evolučné algoritmy (evolutionary computing)*
        - *genetické algoritmy*

Perceptrón (jednovrstvový), (“single-layer perceptron”)



jednoduchá neurónová sieť pre klasifikáciu špeciálneho typu vzoriek, ktoré sú *lineárne separovateľné*

$$y(n) = \mathrm{sgn}\left[\mathbf{w}^T(n)\mathbf{x}(n)\right]$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\left[d(n) - y(n)\right]\mathbf{x}(n)$$

$$d(n) = \begin{cases} +1 & \text{ak } \mathbf{x}(n) \in C_1 \\ -1 & \text{ak } \mathbf{x}(n) \in C_2 \end{cases}$$

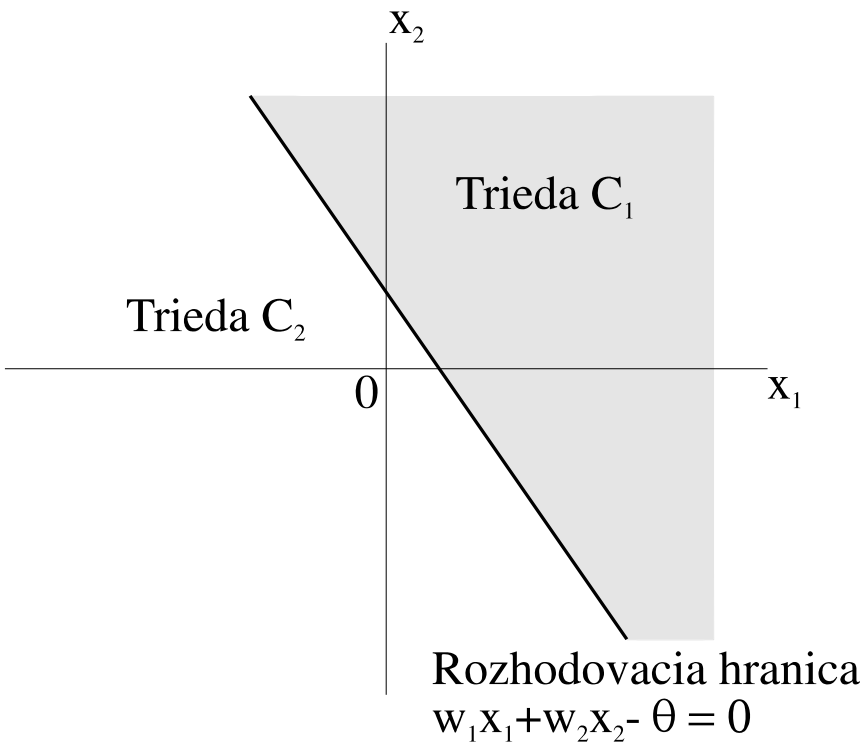
klasické učenie korigujúce chybu, kde η je parameter rýchlosti učenia, a rozdiel $d(n) - y(n)$ reprezentuje chybový signál

- <u>lineárna a nelineárna separovateľnosť</u>

dva rozhodovacie regióny oddelené hyperrovinou definovanou rovnicou

$$\sum_{i=1}^{p} w_i x_i - \theta = 0$$

nadrovina
(hyperplane)



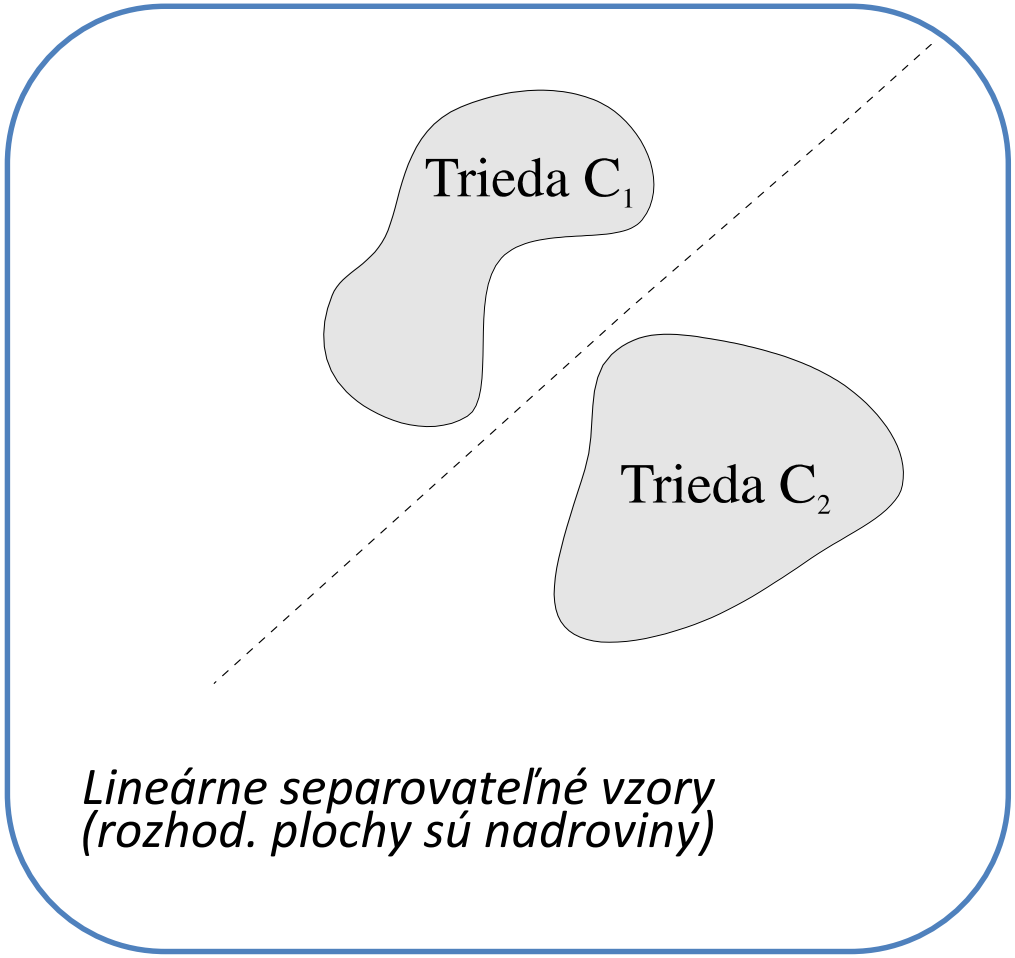
Lineárna separovateľnosť - dvojrozmerné vstupy, dve triedy

Učenie Perceptrónu

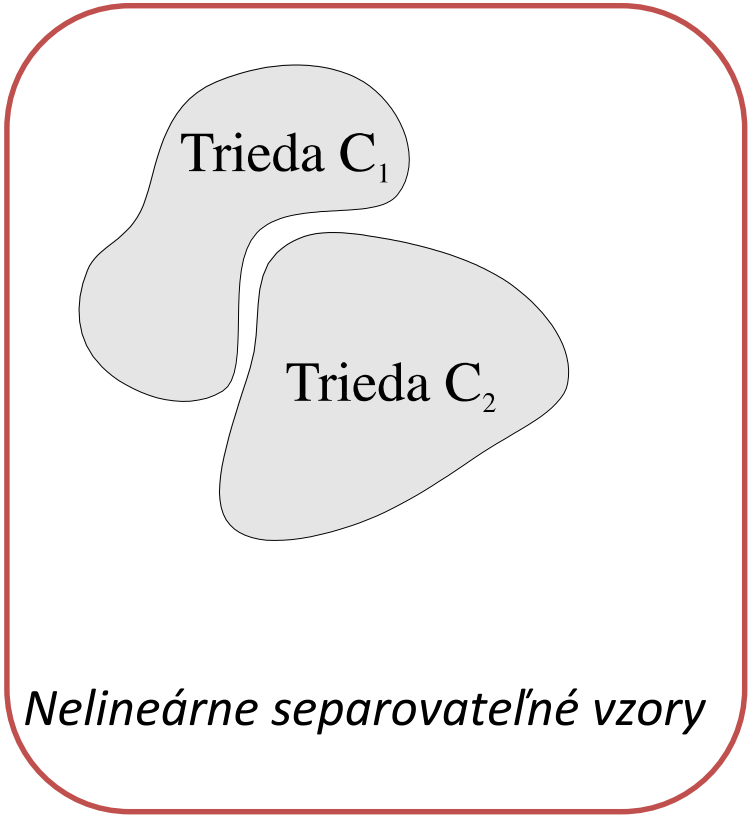
- iterácia *n*: $(p+1)$-rozmerné vektory vstupov $\mathbf{x}(n)$ a váh $\mathbf{w}(n)$
  $\mathbf{x}(n)=[-1, x_1(n), x_2(n),..., x_p(n)]^T$, $\mathbf{w}(n)=[\theta(n), w_1(n), w_2(n),..., w_p(n)]^T$, prah $\theta(n)$, aktuálna odpoveď $y(n)$, žiadaná odpoveď $d(n)$, a parameter rýchlosti učenia η ($0<\eta\leq 1$)
  - *Krok 1 - inicializácia:* Nastav $\mathbf{w}(0)=\mathbf{0}$. Potom pre čas $n=1,2,...$ rob nasledovné kroky.
  - *Krok 2 - aktivácia:* V čase n priveď vstupný vektor $\mathbf{x}(n)$ a žiadanú odpoveď $d(n)$.
  - *Krok 3 - výpočet aktuálnej odpovede:*
    $$y(n)=\text{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)],$$ sgn je funkcia signum.
  - *Krok 4 - adaptácia váhového vektora:*
    $$\mathbf{w}(n+1)=\mathbf{w}(n)+\eta[d(n)-y(n)]\mathbf{x}(n)$$
    kde
    $$d(n)=\begin{cases}+1 & \text{ak } \mathbf{x}(n)\in C_1\\ -1 & \text{ak } \mathbf{x}(n)\in C_2\end{cases}$$
  - *Krok 5:* Inkrementuj n a choď na krok 2.

klasické učenie korigujúce chybu, kde η je parameter rýchlosti učenia, a rozdiel $d(n)-y(n)$ reprezentuje chybový signál

- ešte lineárna a nelineárna separovateľnosť



Lineárne separovateľné vzory (rozhod. plochy sú nadroviny)



Nelineárne separovateľné vzory

- možnosti použitia perceptrónu analyzovali Minsky a Papert 1969
- perceptrón nie je schopný (okrem iných logických funkcií) implementovať ani funkciu XOR (exclusive or)

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

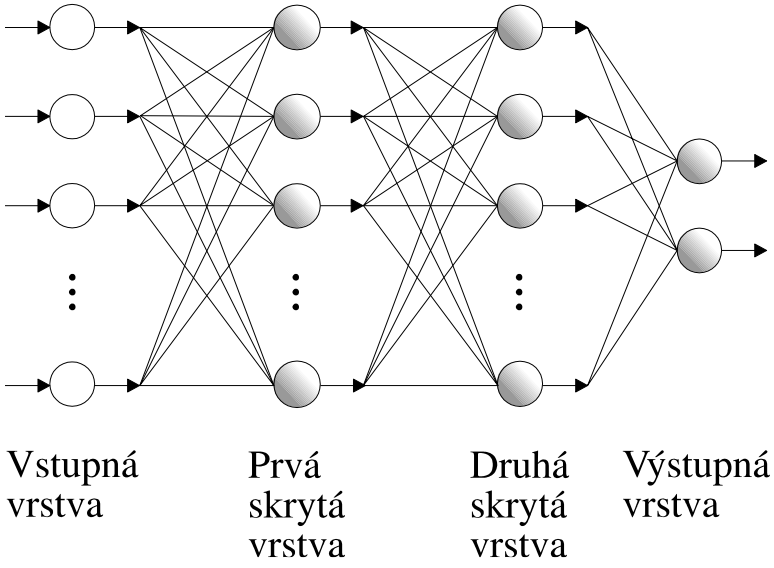
- výskum v oblasti neurónových sietí je ukončený na mŕtvom bode
- pre neurónové siete nasledovali “tiché roky” do polovice osemdesiatych rokov

- 1986: PDP group, *viacvrstvový perceptrón* trénovaný algoritmom spätného šírenia

Viacvrstvový perceptrón (MLP, multilayer perceptron)

- **niekedy aj sieť so spätným šírením (backpropagation network, BP network)**
- **dopredná sieť (viacvrstvová) trénovaná algoritmom spätného šírenia (chýb)**
- **algoritmus spätného šírenia chýb (backpropagation algorithm, error-backpropagation algorithm, BP alg.)**

- jedna alebo viacero *skrytých vrstiev* výpočtových prvkov

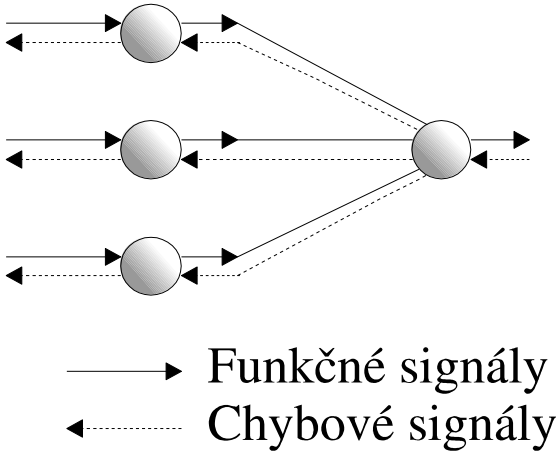


Viacvrstvový perceptrón s dvomi skrytými vrstvami (úplne prepojená sieť)

- hladká nelinearita pre každý neurón (t.j. všade diferencovateľná)
- najčastejšie sigmoidálna nelinearita - napr. logistická funkcia

$$y_j = \frac{1}{1+\exp(-v_j)}$$

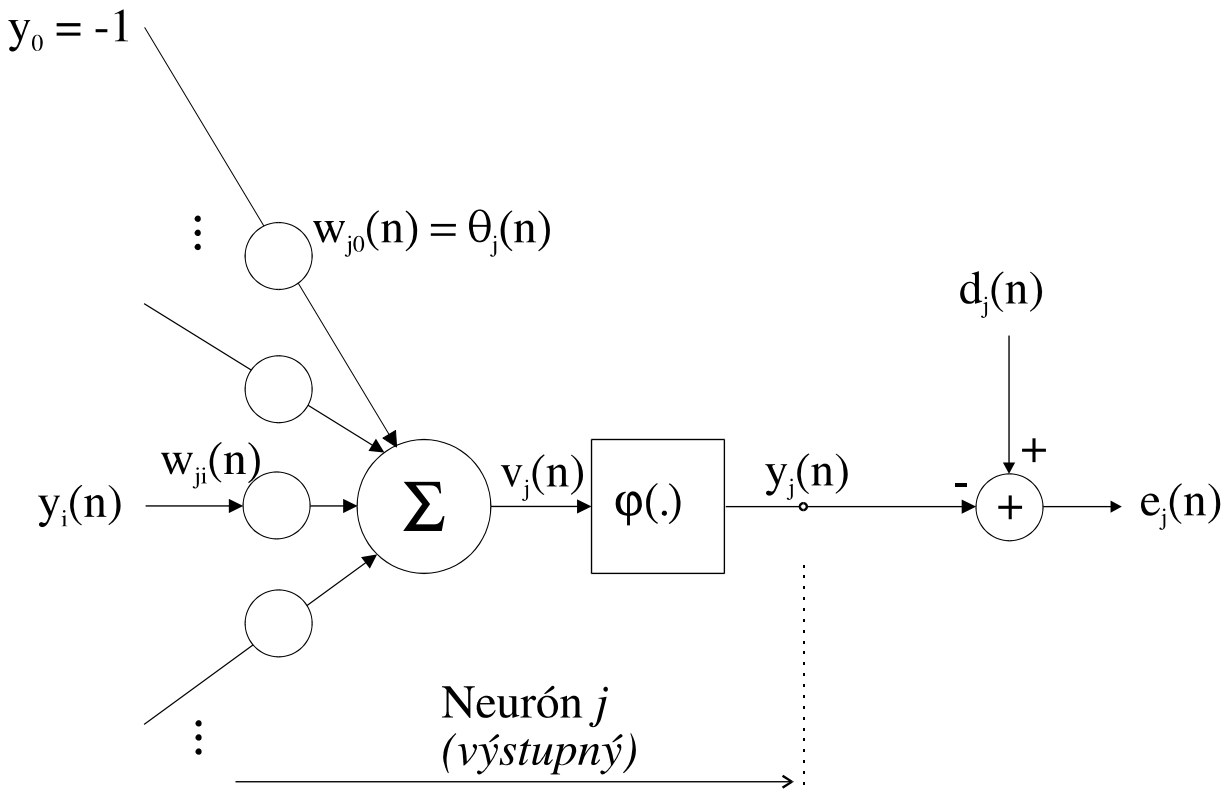
- funkčné a chybové signály:



  - Funkčný signál je signál, ktorý prichádza zo vstupu siete, šíri sa dopredne neurón po neuróne až sa objaví na výstupe siete ako výstupný signál.
  - Chybový signál vzniká vo výstupných neurónoch siete a šíri sa spätne vrstvu po vrstve po sieti (iba počas trénovania).

Algoritmus spätného šírenia (chýb) – (error) backpropagation algorithm

Prípad I: Neurón j je výstupným neurónom



- chybový signál na výstupe neurónu j : $e_j(n) = d_j(n) - y_j(n)$

- suma okamžitých kvadratických chýb: $\varepsilon(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n)$

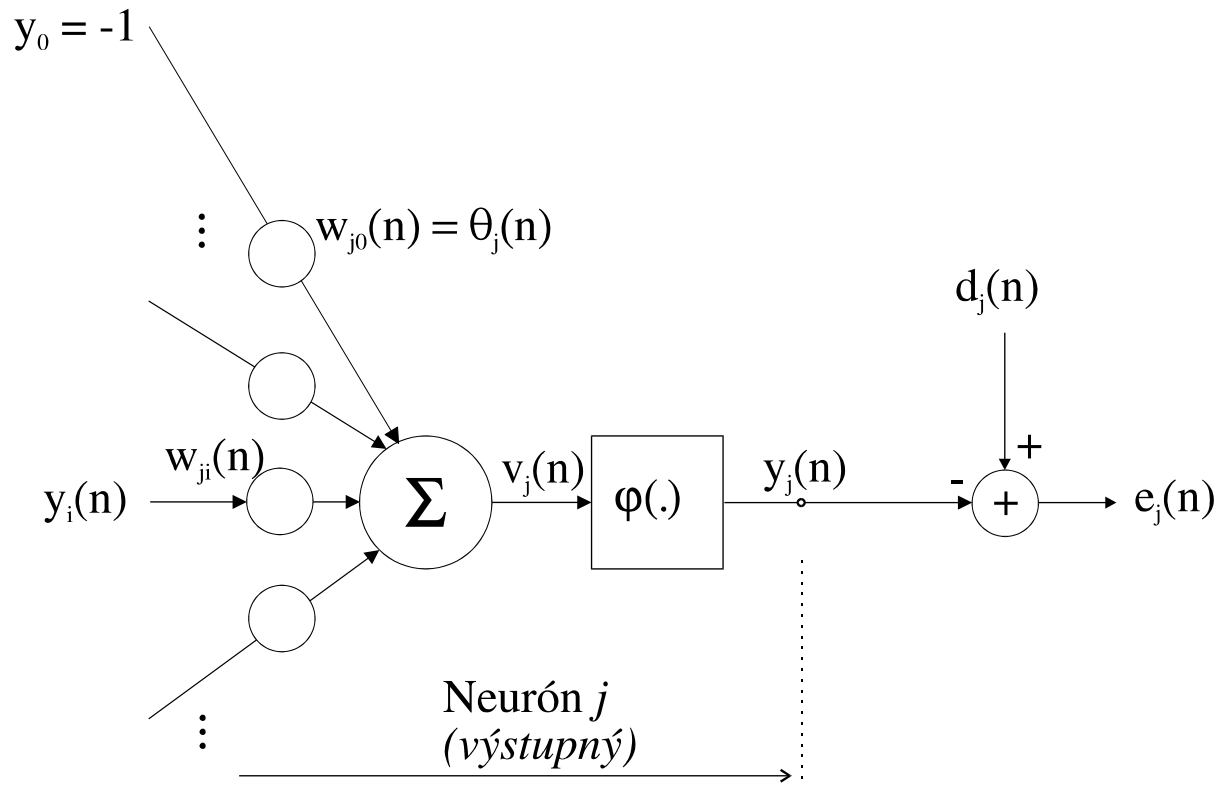
  C – množina výstupných neurónov

- priemerná kvadratická chyba: $\varepsilon_{av} = \frac{1}{N}\sum_{n=1}^{N} \varepsilon(n)$

  N – počet vzoriek (príkladov) trénovacej množiny

  - $\varepsilon(n)$ alebo ε_{av} - kriteriálna funkcia – minimalizujeme vzhľadom na váhy

- minimalizujme $\varepsilon(n)$ - úpravy váh sa robia na základe chýb počítaných pre každú vzorku privedenú do siete



$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n)$$

$$y_j(n) = \varphi_j(v_j(n))$$

- úprava $\Delta w_{ji}(n)$: úmerná okamžitému gradientu $\partial \varepsilon(n) / \partial w_{ji}(n)$

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

reťazové pravidlo ☺
(pravidlo pre výpočet parciálnej derivácie zloženej funkcie)

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n)$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n))$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)$$

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n)$$

tzv. delta pravidlo: $$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$$

znamienko mínus znamená gradientový zostup po chybovom povrchu

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

lokálny gradient $$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n))$$

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

$$\begin{pmatrix} \text{úprava} \\ \text{váhy} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{parameter} \\ \text{rýchlosti učenia} \\ \eta \end{pmatrix} \cdot \begin{pmatrix} \text{lokálny} \\ \text{gradient} \\ \delta_j(n) \end{pmatrix} \cdot \begin{pmatrix} \text{vstupný signál} \\ \text{neurónu j} \\ y_i(n) \end{pmatrix}$$

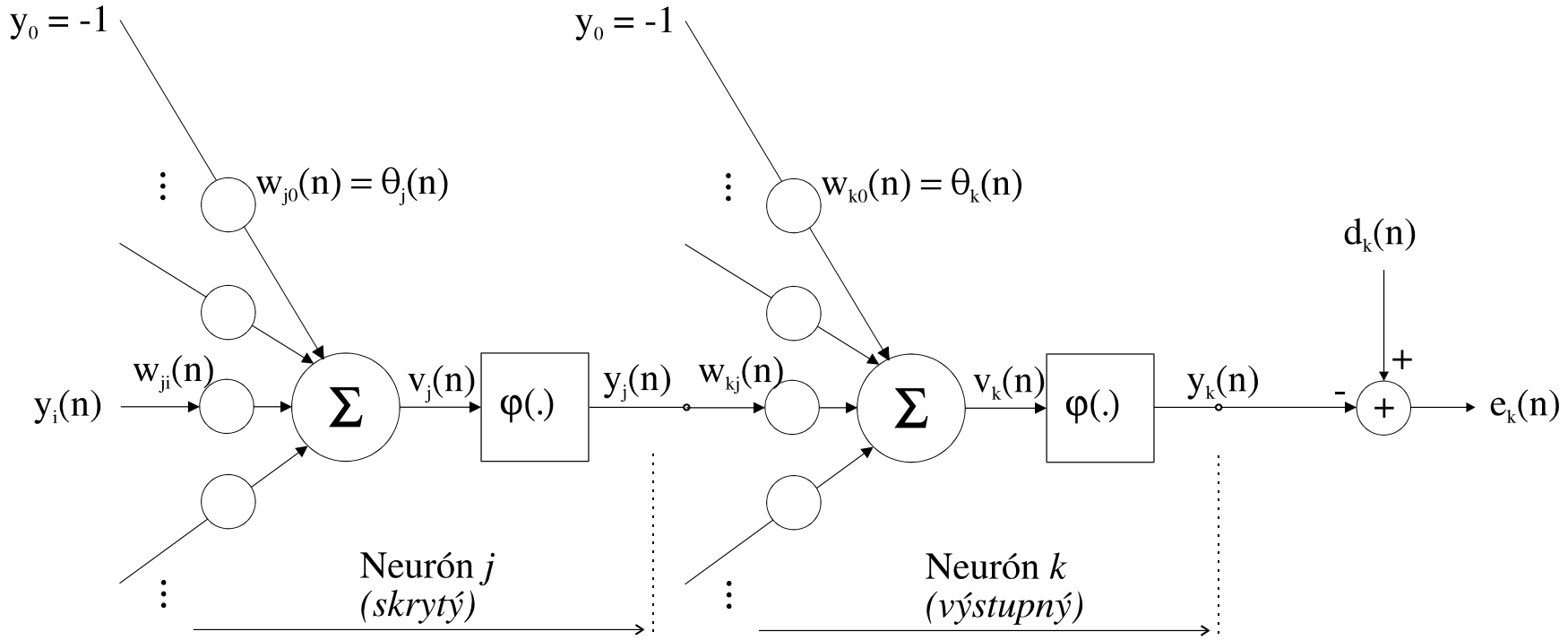
- neurón *j*:
  1. je výstupným neurónom - žiadaná odpoveď je pre výstupné neuróny priamo dostupná

$$\delta_j(n) = e_j(n)\varphi'_j(v_j(n))$$

  2. je skrytým neurónom - žiadaná odpoveď je nedostupná

PROBLÉM! – ale odpoveď dáva algoritmus spätného šírenia:
Chybový signál pre skrytý neurón musí byť vypočítaný rekurzívne podľa chybových signálov neurónov, s ktorými je priamo spojený

Prípad II: Neurón j je skrytým neurónom



- lokálny gradient pre skrytý neurón:

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \varepsilon(n)}{\partial y_j(n)} \varphi_j'(v_j(n))$$

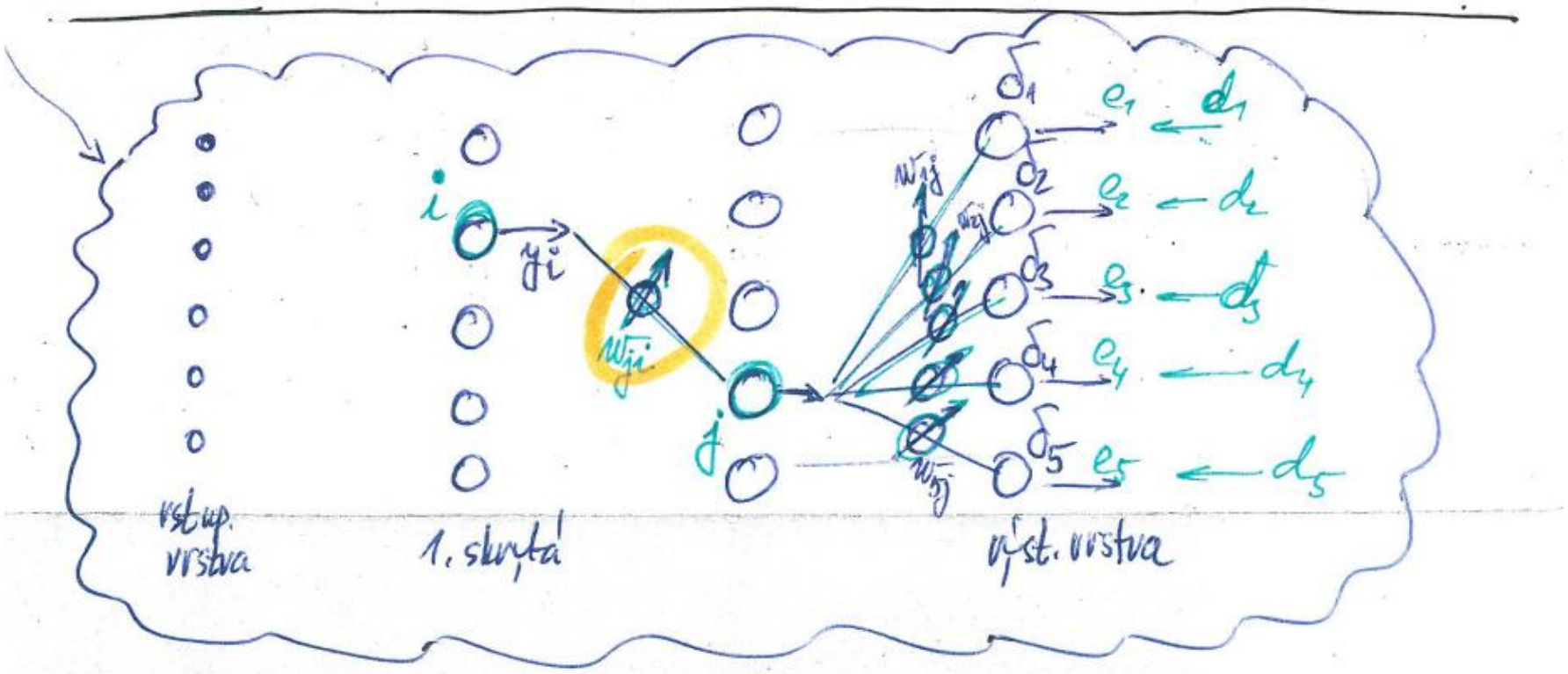
- opäť reťazové pravidlo:

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}$$

- ... (ďalšie úpravy)

$$\delta_j(n) = \varphi_j'\left(v_j(n)\right)\sum_k \delta_k(n) w_{kj}(n)$$

- vysvetlenie spätného šírenia chýb:



$$\delta_j(n) = \varphi_j'\left(v_j(n)\right)\left[w_{1j}(n)\delta_1(n) + w_{2j}(n)\delta_2(n) + \ldots + w_{5j}(n)\delta_5(n)\right]$$

Sumarizácia:

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

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

$$\begin{pmatrix} \text{úprava} \\ \text{váhy} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{parameter} \\ \text{rýchlosti učenia} \\ \eta \end{pmatrix} \cdot \begin{pmatrix} \text{lokálny} \\ \text{gradient} \\ \delta_j(n) \end{pmatrix} \cdot \begin{pmatrix} \text{vstupný signál} \\ \text{neurónu j} \\ y_i(n) \end{pmatrix}$$

- neurón *j*:
  1. je výstupným neurónom - žiadaná odpoveď je pre výstupné neuróny priamo dostupná

$$\delta_j(n) = e_j(n) \varphi_j'(v_j(n))$$

  2. je skrytým neurónom - žiadaná odpoveď je nedostupná

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

váhovaná suma lokálnych gradientov pre neuróny v ďalšej skrytej alebo výstupnej vrstve, ktoré sú spojené s neurónom *j*

- **dve fázy výpočtu**

1) dopredná fáza

- synaptické váhy sa nemenia
- funkčné signály siete sa počítajú postupne neurón po neuróne

p je celkový počet vstupov neurónu *j*

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \qquad y_j(n) = \varphi(v_j(n))$$

- neurón – vstupný:

$$y_i(n) = x_i(n)$$

- neurón – výstupný:

$$y_j(n) = o_j(n)$$

o znamená output

- dopredná fáza výpočtu začína v prvej skrytej vrstve privedením vstupného vektora a končí sa vo výstupnej vrstve určením chybových signálov pre všetky výstupné neuróny

2) spätná fáza

- začína vo výstupnej vrstve siete, z nej sa chybové signály šíria spätne postupne vrstvu po vrstve a rekurzívne sa počítajú lokálne gradienty δ pre každý neurón

- upravujú sa váhy

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

- **sigmoidálna nelinearita**
  - výpočet lokálnych gradientov δ - treba vedieť deriváciu aktivačnej funkcie
  - jediná podmienka pre aktivačnú funkciu – *diferencovateľnosť*
  - napr. logistická funkcia:

$$y_j(n) = \varphi_j\left(v_j(n)\right) = \frac{1}{1+\exp\left(-v_j(n)\right)}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'\left(v_j(n)\right) = \frac{\exp\left(-v_j(n)\right)}{\left[1+\exp\left(-v_j(n)\right)\right]^2}$$

$$\varphi_j'\left(v_j(n)\right) = y_j(n)\left[1-y_j(n)\right]$$

- neurón – výstupný:

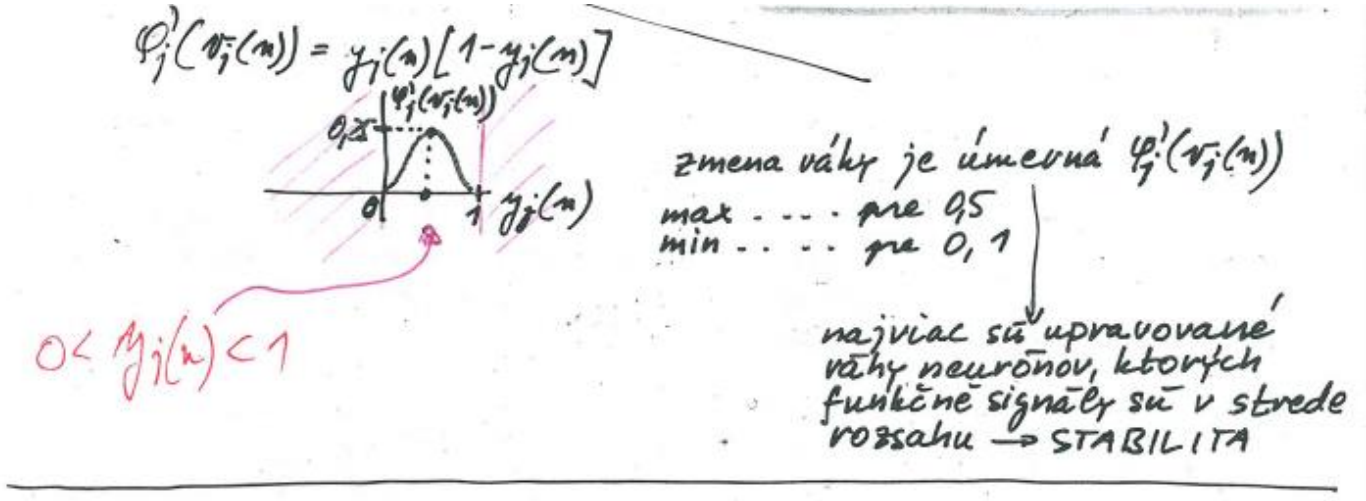
$$y_j(n) = o_j(n)$$

$$\begin{aligned}\delta_j(n) &= e_j(n)\varphi'_j(v_j(n)) \\ &= \left[d_j(n) - o_j(n)\right] o_j(n)\left[1 - o_j(n)\right]\end{aligned}$$

- neurón – skrytý:

$$\begin{aligned}\delta_j(n) &= \varphi'_j(v_j(n))\sum_k \delta_k(n) w_{kj}(n) \\ &= y_j(n)\left[1 - y_j(n)\right]\sum_k \delta_k(n) w_{kj}(n)\end{aligned}$$

- najviac sú upravované váhy tých neurónov so sigmoidálnou nelinearitou, ktorých výstupné hodnoty sú v strede svojho rozsahu
- toto prispieva k stabilite učenia



- **rýchlosť učenia**

  - parameter rýchlosti učenia η :
    - veľmi malá hodnota - malé zmeny váh siete, hladká trajektória vo váhovom priestore, učenie siete veľmi pomalé
    - príliš veľká hodnota – veľké zmeny váh, môže to viesť k nestabilite

- **momentová technika algoritmu spätného šírenia**

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1)$$

  - α - zvyčajne kladné číslo - *momentová konštanta*
    - pre konvergenciu učenia má platiť

$$0 \leq |\alpha| < 1$$

  - metóda pre zvýšenie rýchlosti učenia
  - predídenie nestability učenia

  - MOŽNOSŤ zabrániť procesu učenia skončiť v lokálnom minime

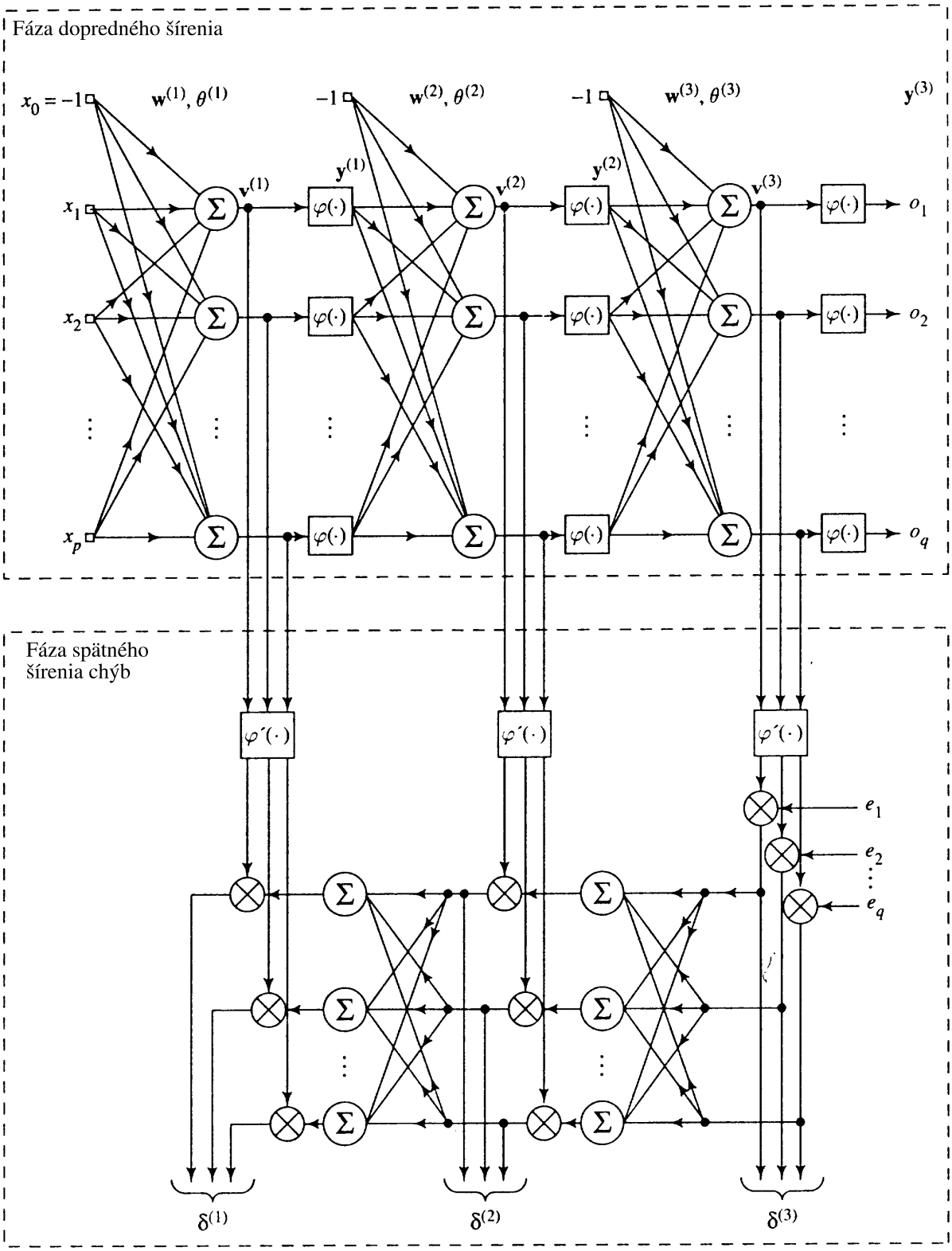
- **vzorkový a dávkový mód učenia**
  - vzorkový mód učenia - váhy upravované po každej prezentácii vzorky
  - dávkový mód učenia - váhy sa upravujú až po privedení všetkých vzoriek trénovacej množiny (t.j. po každej epoche)

epocha (cyklus) je privedenie úplnej trénovacej množiny

  - najčastejšie sa využíva vzorkový mód činnosti:
    - menej pamäte pre každú váhu
    - možnosť sieti prezentovať vzorky v náhodnom poradí - môže zamedziť ukončeniu učenia v lokálnom minime
  - dávkový mód činnosti umožňuje presne určiť gradientový vektor
  - výber módu závisí od konkrétnej úlohy

Zhrnutie algoritmu spätného šírenia

Učenie viacvrstvového perceptrónu algoritmom spätného šírenia je ilustrované na obrázku [Haykin,1994]. Zahŕňa doprednú fázu výpočtu a aj fázu spätného šírenia chýb.



Označenie v tomto obrázku je nasledovné:

$\mathbf{w}^{(l)}$ váhový vektor neurónu vo vrstve l

$\theta^{(l)}$ prah neurónu vo vrstve l

$\mathbf{v}^{(l)}$ vektor vnútorných aktivít neurónov vo vrstve l

$\mathbf{y}^{(l)}$ vektor funkčných signálov neurónov vo vrstve l

l index vrstvy, $l = 0 \ldots L$, $l = 0$ pre vstupnú a $l = L$ pre výstupnú vrstvu

$?^{(l)}$ vektor lokálnych gradientov neurónov vo vrstve l

$\mathbf{e}$ chybový vektor, jeho prvky sú $e_1, e_2, \ldots, e_q$

Uvažujme vzorkový mód činnosti. Pre trénovacie údaje $\left\{\left[\mathbf{x}(n), \mathbf{d}(n)\right], n = 1,2,\ldots,N\right\}$ algoritmus prechádza nasledovnými cyklami:

Krok 1 - inicializácia: Nastav váhy a prahy siete na malé náhodné čísla.

Krok 2 - prezentácia trénovacích údajov: Priveď do siete trénovacie príklady, ktoré tvoria jednu epochu. Pre každý príklad rob postupnosť dopredných a spätných výpočtov podľa bodov 3 a 4.

Krok 3 - dopredný výpočet: Dodaj sieti trénovací príklad $\left[\mathbf{x}(n), \mathbf{d}(n)\right]$ Počítaj funkčné signály siete pre jednotlivé neuróny:

$$v_j^{(l)}(n) = \sum_{i=0}^{p} w_{ji}^{(l)}(n) y_i^{(l-1)}(n)$$

kde pre $i = 0$ je $y_0^{(l-1)}(n) = -1$ a $w_{j0}^{(l)}(n) = \theta_j^{(l)}(n)$. Ak je sigmoidálnou nelinearitou logistická funkcia, potom

$$y_j^{(l)}(n) = \frac{1}{1 + \exp\left(-v_j^{(l)}(n)\right)}$$

Ak $l = 0$, potom nastav $y_j^{(0)}(n) = x_j(n)$, ak $l = L$, potom nastav $y_j^{(L)}(n) = o_j(n)$.

Potom počítaj chybový signál

$$e_j(n) = d_j(n) - o_j(n)$$

Krok 4 - spätná fáza výpočtu: Počítaj lokálne gradienty δ siete, postupne od výstupnej smerom k vstupnej vrstve:

$$\delta_j^{(L)}(n) = e_j^{(L)}(n) o_j(n)\left[1 - o_j(n)\right],$$

pre neurón j vo výstupnej vrstve L

$$\delta_j^{(l)}(n) = y_j^{(l)}(n)\left[1 - y_j^{(l)}(n)\right]\sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n),$$

pre neurón j v skrytej vrstve l

Uprav váhy vo vrstve l:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)\right] + \eta \delta_j^{(l)}(n) y_i^{(l-1)}(n)$$

kde η je parameter rýchlosti učenia a α je momentová konštanta.

Krok 5 - iterovanie: Iteruj výpočet prezentovaním nových epoch trénovacích príkladov, pokiaľ sa váhy siete nestabilizujú a priemerná kvadratická chyba ε_{av} je v minime alebo v akceptovateľne malej hodnote. Poradie prezentácie trénovacích príkladov by sa malo pre rôzne epochy náhodne meniť. Momentová konštanta a parameter rýchlosti učenia sa s rastúcim počtom iterácií upravujú (zvyčajne znižujú).

Tipy pre lepšiu činnosť BP algoritmu

- sigmoidálna **aktivačná funkcia** môže byť **nesymetrická** – napr. tgh

- hodnoty žiadaných odpovedí pre neuróny výstupnej vrstvy môžu byť z intervalu, ktorý má hranice posunuté o hodnotu μ od hraníc pôvodného rozsahu
  - napr. pre logistickú funkciu **namiesto (0,1) využijeme (μ ,1- μ)**

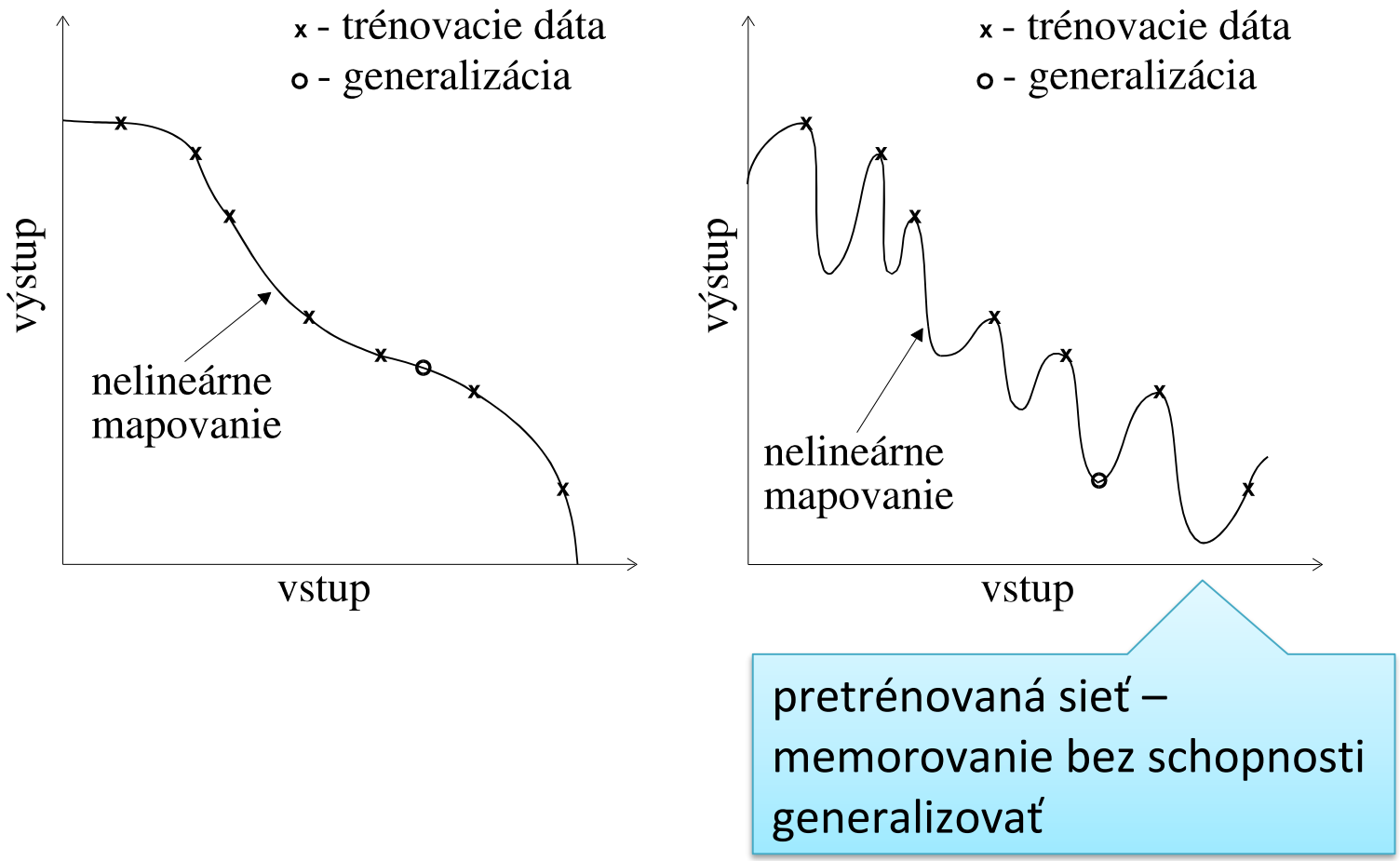
- **inicializácia váh a prahov** siete - rovnomerne rozložené hodnoty v malom rozsahu

- **parameter rýchlosti učenia** η môže mať menšiu hodnotu pre posledné vrstvy siete
  - platí tiež, že neuróny s mnohými vstupmi by mali mať menšiu hodnotu η ako neuróny s malým počtom vstupov

- pre on-line činnosť veľkých sietí je vhodnejší **vzorkový mód činnosti** - rýchlejšie trénovanie
- náhodné poradie vzoriek pre jednotlivé epochy trénovania

Generalizácia

- správanie siete bude správne pre vstupno-výstupné testovacie dáta, ktoré neboli použité počas trénovania
- neurónová sieť - nelineárne vstupno-výstupné mapovanie
- správna generalizácia a negeneralizovanie:



pretrénovaná sieť – memorovanie bez schopnosti generalizovať

Matlab

- nnd11fa – aproximácia funkcií
- nnd11gn – ukážka generalizácie
- nnd12sd1 – ukážka najstrmšieho zostupu pomocou alg. spätného šírenia
- nnd12sd2 - ukážka najstrmšieho zostupu pomocou alg. spätného šírenia

Univerzálna aproximačná teoréma

Nech $\varphi(\cdot)$ *je nekonštantná, ohraničená, monotónne rastúca, spojitá funkcia. Nech* I_p *je* p*-rozmerná jednotková hyperkocka* $[0,1]^p$. *Nech* $C(I_p)$ *znamená priestor spojitých funkcií na* I_p. *Potom ak je daná ľubovoľná funkcia* $f \in C(I_p)$ *a* $\varepsilon > 0$, *tak existuje celé číslo* M *a množina reálnych konštánt* $\alpha_i, \theta_i, w_{ij}$, *kde* $i = 1,\ldots,M$ *a* $j = 1,\ldots,p$ *takých, že môžeme definovať*

$$F(x_1,\ldots,x_p) = \sum_{i=1}^{M} \alpha_i \varphi\left(\sum_{j=1}^{p} w_{ij} x_j - \theta_i \right) \qquad (*)$$

ako aproximáciu funkcie $f(\cdot)$*; t.j.*

$$\left| F(x_1,\ldots,x_p) - f(x_1,\ldots,x_p) \right| < \varepsilon$$

pre všetky $\{x_1,\ldots,x_p\} \in I_p$.

- Táto teoréma je priamo aplikovateľná na viacvrstvový perceptrón.
- Logistická funkcia $1/[1+\exp(-v)]$ použitá ako nelinearita pre model neurónu viacvrstvového perceptrónu je nekonštantná, ohraničená, monotónne rastúca, spojitá funkcia - spĺňa podmienky kladené na funkciu $\varphi(\cdot)$.

- Rovnica (*) reprezentuje výstup viacvrstvového perceptrónu opísaného nasledovne:
  - sieť má *p* vstupných prvkov, jednu skrytú vrstvu s *M* neurónmi, vstupy sú $\{x_1,\ldots,x_p\}$
  - skrytý neurón *i* má synaptické váhy $\{w_{i1},\ldots,w_{ip}\}$ a prah θ_i
  - výstup siete je lineárna kombinácia výstupov skrytých neurónov, kde $\alpha_1,\ldots,\alpha_M$ definujú koeficienty tejto kombinácie

- **MLP s viacerými skrytými vrstvami**

- MLP s 1 skrytou vrstvou:
  - globálnainterakcia neurónov - je ťažké vylepšiť aproximáciu v jednom bode bez zhoršenia aproximácie v inom bode

- MLP s 2 skrytými vrstvami:
  - Prvá skrytá vrstva – extrakcia lokálnych príznakov

Niektoré neuróny prvej skrytej vrstvy delia vstupný priestor do oblastí (podpriestorov) a iné neuróny tejto vrstvy sa učia lokálne príznaky charakterizujúce tieto regióny.

  - Druhá skrytá vrstva – extrakcia globálnych príznakov

Neuróny druhej skrytej vrstvy kombinujú výstupy neurónov prvej skrytej vrstvy pracujúcich na podpriestore vstupných dát a učia sa globálne príznaky pre tento podpriestor; mimo neho nereagujú.

Univerzálna aproximácia

Aproximácia funkcie pomocou viacvrstvového perceptrónu trénovaného algoritmom spätného šírenia s jediným výstupným prvkom sa môže zapísať v nasledovnom kompaktnom tvare:

$$F(\mathbf{x},\mathbf{w}) = \varphi\left(\sum_j w_{oj}\,\varphi\left(\sum_k w_{jk}\,\varphi\left(\ldots\varphi\left(\sum_i w_{li}\,x_i\right)\ldots\right)\right)\right)$$

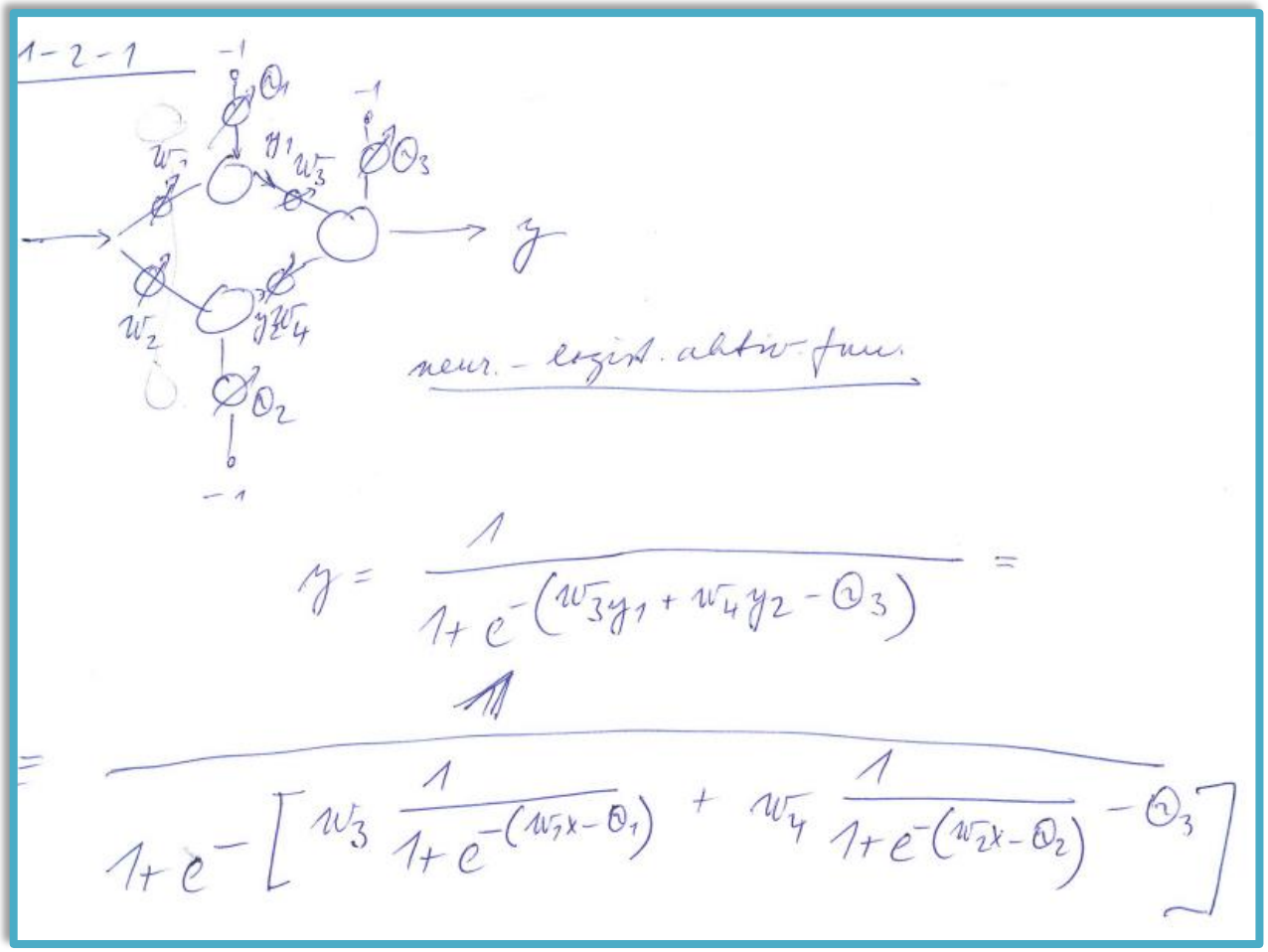
Schéma do seba vnorených nelineárnych funkcií, v prípade sigmoidálnych funkcií sa nazýva "**nested sigmoidal scheme**"

- $\varphi(\cdot)$ je sigmoidálna aktivačná funkcia
- w_{oj} je synaptická váha z neurónu j v poslednej skrytej vrstve do výstupného neurónu o a tak ďalej pre iné synaptické váhy
- x_i je *i*-ty prvok vstupného vektora $\mathbf{x}$

Váhový vektor $\mathbf{w}$ sa vzťahuje na celú množinu synaptických váh zoradených podľa vrstvy, potom podľa neurónov vo vrstve, a potom podľa poradového čísla váhy v neuróne.

Takáto schéma je *univerzálnym aproximátorom* - viacvrstvový perceptrón trénovaný algoritmom spätného šírenia môže aproximovať ľubovoľnú spojitú funkciu do požadovaného stupňa presnosti za predpokladu, že k dispozícii je dostatočne veľa skrytých neurónov.

Výstup MLP konfigurácie 1-2-1



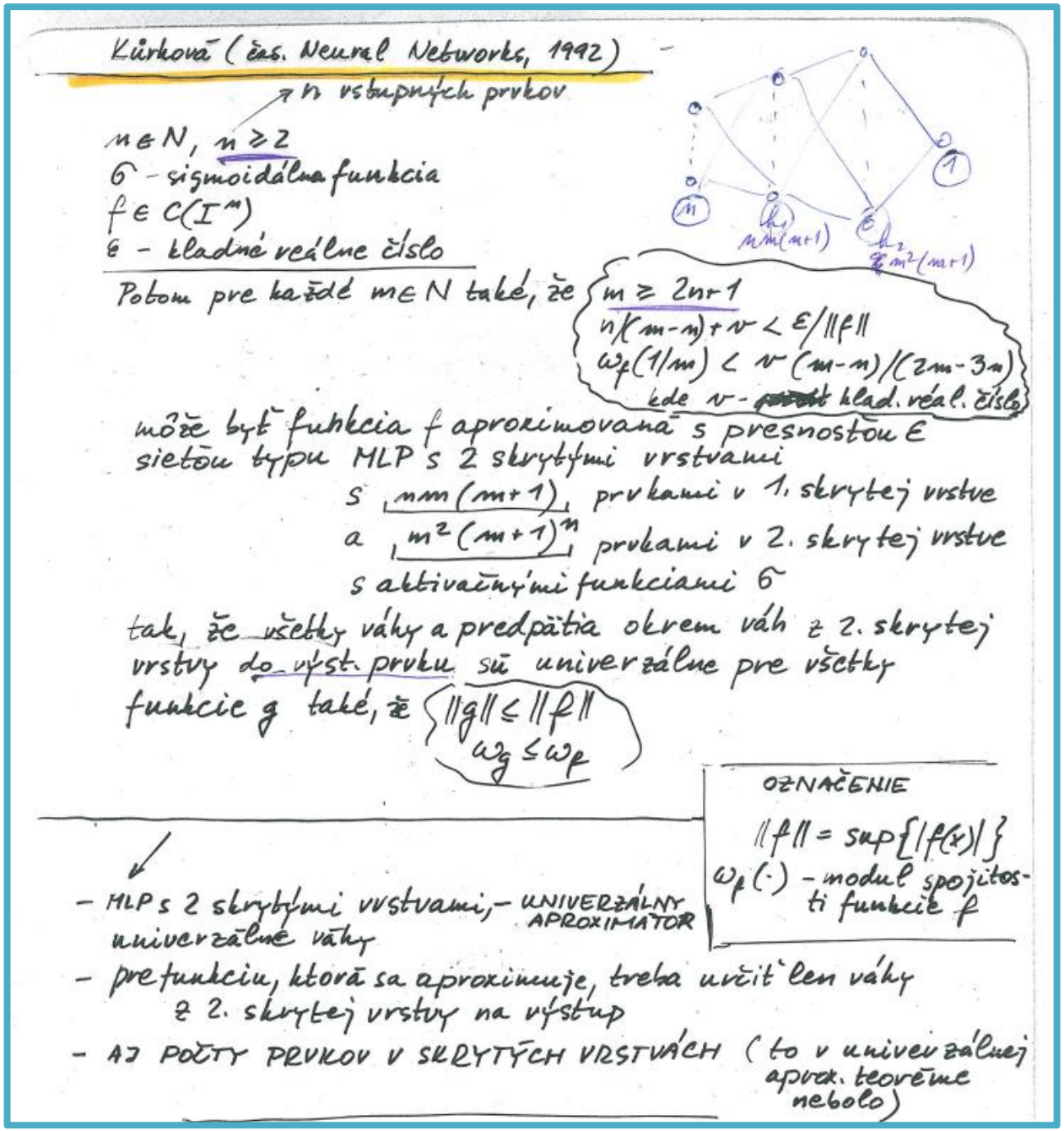
Teorémy o aproximačných schopnostiach MLP

- any continuous function (1 variable or more) on a compact set can be approximated to any accuracy by a linear combination ofsigmoids
  - -> function approximation
    - [for example: K.Hornik et al. Multilayer Feedforward Networks are Universal Approximators, Neural Networks, Vol. 2, pp 359-366 (1989)]

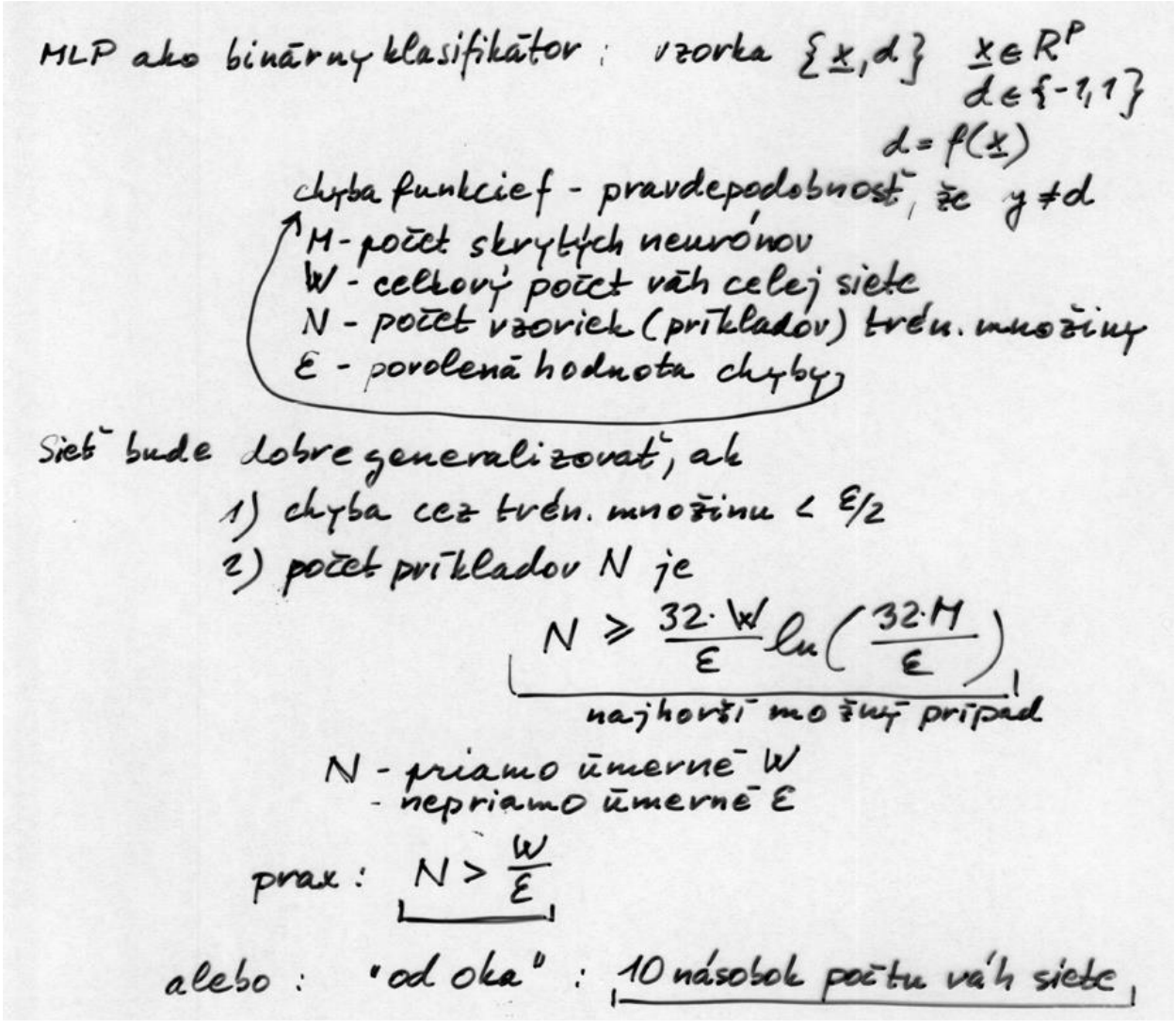
- trained with f(x) = 1 for signal and = 0 for background, the NN function approximates the probability of signal knowing x
  - -> classification (cf Neyman-Pearson test)
    - [for example: D.W.Ruck et al. The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function, IEEE Transactions on Neural Networks, Vol. 1, nr 4, pp 296-298 (1990)]

http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/INFO/Workshops/9905/slides/thu.7/talk_mai99.pdf

V. Kůrková:
Kolmogorov's theorem and multilayer neural networks,
Neural Networks 5: 501-506, 1992



Návrh MLP – „viac hranie sa ako veda“ - napr. veľkosť trénovacej množiny



Ďalšie spôsoby trénovania MLP - premenlivá rýchlosť učenia, konjugovaný gradient, Levenberg–Marquardt

- nnd12vl
  - Variable learning rate backpropagation demonstration
    - *You can improve the performance of the steepest descent algorithm if you allow the learning rate to change during the training process. An adaptive learning rate attempts to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface.*
- nnd12mo
  - Momentum backpropagation demonstration
    - *Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Acting like a lowpass filter, momentum allows the network to ignore small features in the error surface. Without momentum a network can get stuck in a shallow local minimum. With momentum a network can slide through such a minimum.*

- nnd12ls
  - ➢ Conjugate gradient line search

- nnd12cg
  - ➢ Conjugate gradient backpropagation demonstration
    - *The **basic backpropagation algorithm** adjusts the weights in the steepest descent direction (negative of the gradient), the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence.*
    - *In the **conjugate gradient algorithms** a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. This section presents four variations of conjugate gradient algorithms.*
    - *In most of the training algorithms discussed up to this point, a learning rate is used to determine the length of the weight update (step size). In most of the conjugate gradient algorithms, the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size that minimizes the performance function along that line. There are five different search functions included in the toolbox, and these are discussed in Line Search Routines.*

- nnd12m
  - Marquardt backpropagation demonstration

- nnd12ms
  - Marquardt step

Levenberg-Marquardt

Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as

$$\mathbf{H} = \mathbf{J}^T\mathbf{J}$$

and the gradient can be computed as

$$\mathbf{g} = \mathbf{J}^T\mathbf{e}$$

where $\mathbf{J}$ is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and $\mathbf{e}$ is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique (see [HaMe94]) that is much less complex than computing the Hessian matrix.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

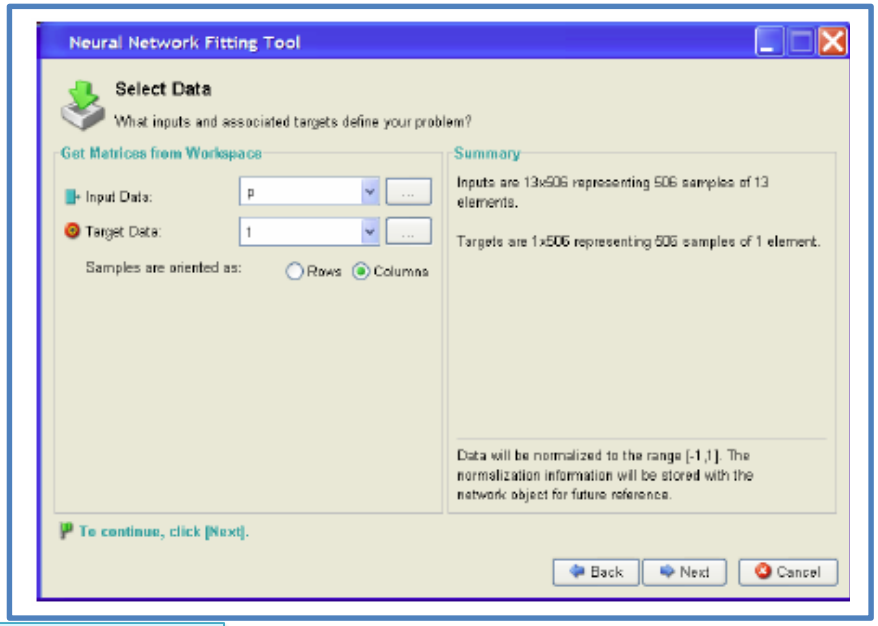
$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}]^{-1}\mathbf{J}^T\mathbf{e}$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function is always reduced at each iteration of the algorithm.

zdroj: Matlab – dokumentácia

- nntool
  - neural network toolbox
  - napr.
    - net=newff([-1 2; 0 5],[3,1],{'tansig','purelin'},'traingd');

- nftool
  - Neural Network Fitting Tool
    - load housing
    - nftool



Takéto tools: podľa verzie Matlabu, existujú aj
- nctool - neural network classification tool
- nprtool - neural network pattern recognition tool

Zjednodušovanie sietí - network pruning:

- používa sa aj pre iné siete, nielen pre MLP
- cieľ – minimalizovať veľkosť siete pri zachovaní správnej činnosti
  - menšia sieť lepšie generalizuje

1. rast siete (network growing)
   - začína sa s min. konfiguráciou, potom pridanie nových neurónov alebo vrstiev
   - napr. kaskádová korelácia (cascade correlation)
     - začína sa s min. konfiguráciou (iba vst. a výst. vrstva), potom postupné pridávanie skrytých neurónov
     - nie do vrstiev, ale kaskádovanie – pridaný neurón dostáva vstupy zo vstupnej vrstvy a aj od všetkých dovtedy pridaných neurónov
     - vstupné váhy pridaného neurónu sa zmrazia, trénujú sa iba výstupné váhy

2. zmenšovanie siete (network pruning)
   - útlm váh (weight decay)
     - niektorým váham v sieti sú vnútené hodnoty blízke nule, iným je dovolené mať veľké hodnoty
     - váhy sa rozdelia na „užitočné“ a „prebytočné“
     - zmenšuje sa počet váh, nie neurónov
   - eliminácia váh (weight elimination)
     - najprv trénovanie, potom odstránenie neurónov
   - optimal brain damage ☺
     - využívajú sa druhé derivácie chybového povrchu

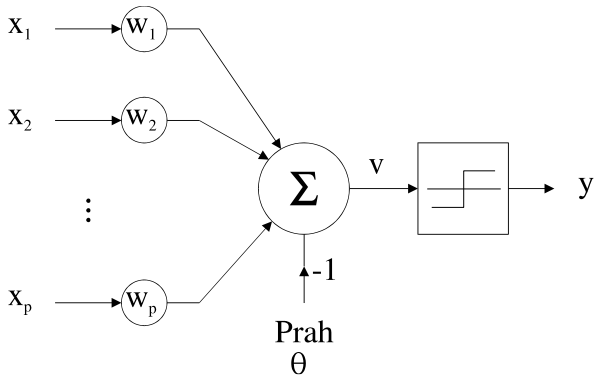
decay – zánik, úpadok, hnitie, rozklad, útlm

3. iné
   - evolučné algoritmy (evolutionary computing)
     - genetické algoritmy
       - ✓ napr. pre určenie počtu parametrov, na ktoré sa má pruning aplikovať (počas učenia podľa vývoja váh a vývoja generalizácie siete)

PERCEPTRÓN A VIACVRSTVOVÝ PERCEPTRÓN

4.1 Jednovrstvový perceptrón

Jednovrstvový perceptrón ("single-layer perceptron"), nazývaný jednoducho *perceptrón* ("perceptron") [Ros62], [Hay94], je jednoduchá neurónová sieť pre klasifikáciu špeciálneho typu vzoriek, ktoré sú *lineárne separovateľné*.



Obr. 4.1 *Jednovrstvový perceptrón*

Jednovrstvový perceptrón znázornený na obr. 4.1 pozostáva iba z jediného neurónu. Takýto perceptrón je schopný klasifikovať vzorky iba do dvoch tried. Ak výstupná vrstva obsahuje viac neurónov, je možné klasifikovať aj viac tried vzoriek, ktoré však musia byť lineárne separovateľné.

Základným prvkom perceptrónu je McCullochov-Pittsov model neurónu, ktorý sa skladá z lineárneho kombinátora a prahového zariadenia na výstupe. Synaptické váhy perceptrónu sú označené ako $w_1, w_2, \ldots, w_p$. Sumátor počíta lineárnu kombináciu vstupov $x_1, x_2, \ldots, x_p$ a berie do úvahy aj prah θ. Táto suma je vstupom do prahového zariadenia; výstup neurónu je +1, ak je vstup prahového zariadenia kladný a -1, ak je vstup záporný.
Výstup lineárneho kombinátora (vstup prahového zariadenia) je teda

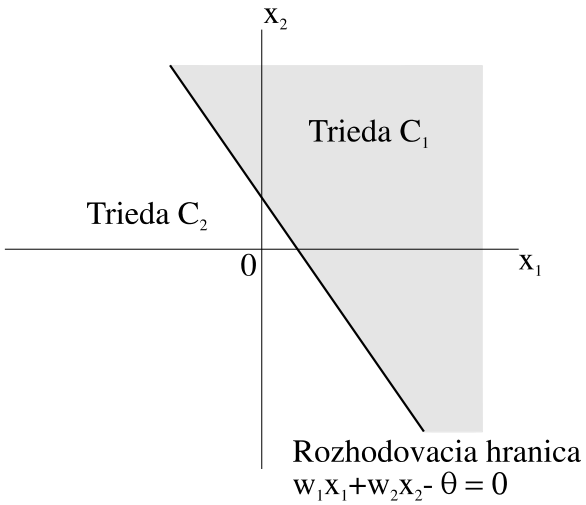
$$v = \sum_{i=1}^{p} w_i x_i - \theta \qquad (4.1)$$

Úlohou perceptrónu je klasifikovať vstupy $x_1, x_2, \ldots, x_p$ do jednej z tried C_1, C_2. Rozhodovacím pravidlom pre klasifikáciu je priradenie bodu reprezentovaného vstupmi $x_1, x_2, \ldots, x_p$ do triedy C_1, ak výstup perceptrónu je +1 a do triedy C_2, ak výstup je -1.

Činnosť takéhoto klasifikátora môže ozrejmiť graf rozhodovacích regiónov v p-rozmernom priestore, ktorý zahŕňa p vstupných premenných $x_1, x_2, \ldots, x_p$. V prípade najjednoduchšieho perceptrónu (iba jeden neurón vo výstupnej vrstve) sú definované dva rozhodovacie regióny oddelené hyperrovinou definovanou rovnicou

$$\sum_{i=1}^{p} w_i x_i - \theta = 0 \qquad (4.2)$$

Toto je znázornené na obr. 4.2 pre prípad dvoch vstupných premenných x_1 a x_2 - v tomto prípade je rozhodovacou hranicou priamka. Bod (x_1, x_2) ležiaci nad hraničnou priamkou je priradený do triedy C_1, bod (x_1, x_2) ležiaci pod hraničnou priamkou je priradený do triedy C_2. Úlohou prahu θ je posúvať hraničnú priamku voči počiatku.



Obr. 4.2 *Lineárna separovateľnosť - dvojrozmerné vstupy, dve triedy*

Synaptické váhy perceptrónu sú upravované iteračne podľa algoritmu, ktorý teraz uvedieme. Použijeme model perceptrónu z obr. 4.3, ktorý je ekvivalentný s obr. 4.1 - podobne ako v prvej kapitole je prah reprezentovaný váhou, ktorá dostáva konštantný vstup rovnajúci sa -1 a má možnosť sa adaptovať.

V n-tej iterácii označme $(p+1)$-rozmerné vektory vstupov $\mathbf{x}(n)$ a váh $\mathbf{w}(n)$ ako $\mathbf{x}(n) = [-1, x_1(n), x_2(n), \ldots, x_p(n)]^T$, $\mathbf{w}(n) = [\theta(n), w_1(n), w_2(n), \ldots, w_p(n)]^T$, prah ako $\theta(n)$, aktuálnu odpoveď ako $y(n)$, žiadanú odpoveď ako $d(n)$ a parameter rýchlosti učenia ako η ($0 < \eta \leq 1$).

Krok 1 - inicializácia: Nastav $\mathbf{w}(0) = \mathbf{0}$. Potom pre čas $n = 1,2,\ldots$ preveď nasledovné kroky.

Krok 2 - aktivácia: V čase n priveď vstupný vektor $\mathbf{x}(n)$ a žiadanú odpoveď $d(n)$.

Krok 3 - výpočet aktuálnej odpovede:

$$y(n) = \operatorname{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)] \qquad (4.3)$$

kde sgn je funkcia signum.

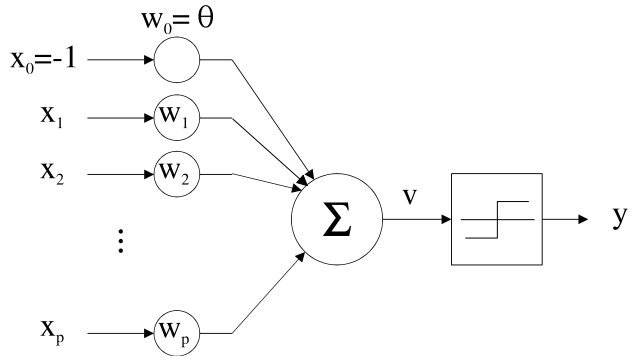
Krok 4 - adaptácia váhového vektora:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n) \qquad (4.4)$$

kde

$$d(n) = \begin{cases} +1 & \text{ak } \mathbf{x}(n) \in C_1 \\ -1 & \text{ak } \mathbf{x}(n) \in C_2 \end{cases} \tag{4.5}$$

Krok 5: Inkrementuj n a choď na krok 2.



Obr. 4.3 *Ekvivalentný model perceptrónu*

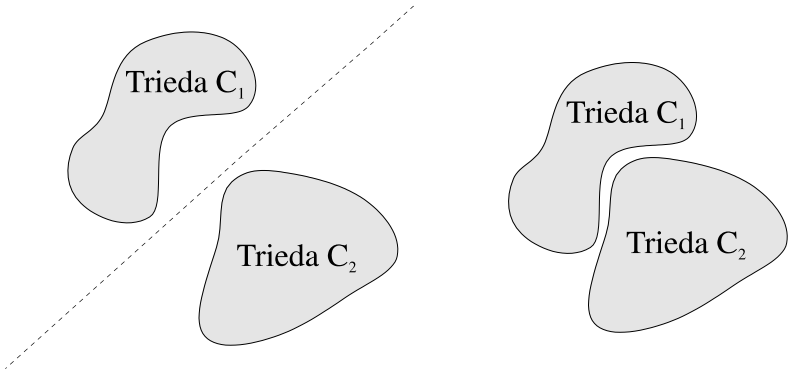
Rovnica (4.4) reprezentuje klasické učenie korigujúce chybu, kde η je parameter rýchlosti učenia, a rozdiel $d(n) - y(n)$ reprezentuje chybový signál. Na parameter η sa kladú (ako vo väčšine prípadov pre parametre rýchlosti učenia neurónových sietí) dve protichodné požiadavky: Priemerovanie predchádzajúcich vstupov pre stabilné nastavenie váh vyžaduje malú hodnotu η, zatiaľ čo rýchla adaptácia váh vyžaduje veľké η.

Vlastnosti perceptrónu boli uvedené pre McCullochov-Pittsov model neurónu, kde nelinearita je reprezentovaná prahovým zariadením na výstupe neurónu. Vynára sa otázka, či pre iný druh nelinearity, napr. sigmoidálnej, nebude správanie perceptrónu lepšie. Odpoveďou je, že činnosť jednovrstvového perceptrónu nezávisí od typu nelinearity využitej v modeli neurónu. Všeobecne platí: Pokiaľ pre jednovrstvový perceptrón využijeme model neurónu, ktorý sa skladá z lineárneho kombinátora a nelineárneho prvku, potom nezávisle od formy použitej nelinearity *vie jednovrstvový perceptrón klasifikovať iba lineárne separovateľné vzorky*.

Lineárna separovateľnosť vyžaduje, aby vzorky, ktoré sa majú klasifikovať, boli dostatočne navzájom separované - t.j. musí byť zaručené, že rozhodovacími plochami sú hyperroviny. Príkladom je situácia pre jednovrstvový perceptrón s dvoma vstupmi na obr. 4.4a. Ak však vzorky z tried C_1 a C_2 sú príliš blízko seba (obr. 4.4b), potom sú nelineárne separovateľné a perceptrón nie je schopný klasifikovať ich správne.

Možnosti použitia perceptrónu analyzovali Minsky a Papert [Min69]. Ukázali, že perceptrón nie je schopný (okrem iných logických funkcií) implementovať ani funkciu XOR (exclusive or), urobili prehľad zlepšení perceptrónu a poukázali na to, že ani tieto zlepšenia nevedú k správnym výsledkom. Zhrnutím bolo, že výskum v oblasti neurónových sietí je ukončený na mŕtvom bode. Potom pre neurónové siete nasledovali "tiché roky" až do polovice osemdesiatych rokov. História však ukázala, že takéto závery sú chybné a dnes máme k dispozícii viacero zložitejších modelov neurónových sietí prevyšujúcich schopnosti perceptrónu. V roku 1986 sa začal "boom" pre neurónové siete uvedením publikácií od PDP skupiny ("parallel distributed processing") [Rum86a], kde bol opísaný *viacvrstvový perceptrón*

trénovaný algoritmom spätného šírenia [Rum86b]. Iným silným a v súčasnosti veľmi využívaným modelom je *RBF sieť* ("radial basis function network").



a) b)

Obr. 4.4 *a) Lineárne separovateľné vzorky*
b) nelineárne separovateľné vzorky

4.2 Viacvrstvový perceptrón

Teraz sa budeme zaoberať dôležitou triedou neurónových sietí, a *to viacvrstvovými doprednými sieťami*. Takéto siete sa skladajú z vrstvy vstupných prvkov (*vstupná vrstva*), ktoré distribuujú vstupné signály do siete, z jednej alebo viacerých *skrytých vrstiev* výpočtových prvkov a z *výstupnej vrstvy* výpočtových prvkov. Vstupné signály sa v sieti šíria dopredným smerom postupne od vrstvy k vrstve. Takéto neurónové siete sú nazývané *viacvrstvové perceptróny* ("multilayer perceptron") [Hay94], [Bis96], [Hech90], [Nov92], [Kva97], [Cich93] (a takmer každá publikácia zaoberajúca sa základmi neurónových sietí), ktoré sú zovšeobecnením jednovrstvového perceptrónu. Viacvrstvové perceptróny sa dajú aplikovať na riešenie náročných problémov. Trénujú sa s učiteľom - využívajú populárny *algoritmus spätného šírenia* ("backpropagation algorithm") [Rum86b], ktorý je založený na pravidle učenia korigujúcom chybu.

Viacvrstvový perceptrón má niekoľko dôležitých charakteristík:

- Model každého výpočtového neurónu v sieti obsahuje na výstupe *nelinearitu*. Veľmi dôležité je, že táto nelinearita musí byť hladká (t.j. všade diferencovateľná) - toto je základný rozdiel voči jednovrstvovému perceptrónu s prahovým zariadením na výstupe neurónu. Najpoužívanejšou formou nelinearity je sigmoidálna nelinearita definovaná napr. logistickou funkciou

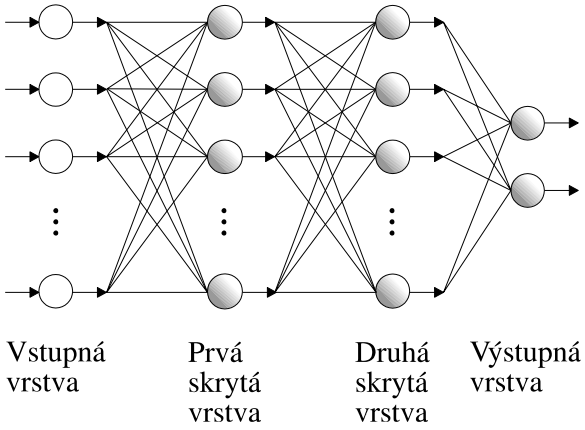
$$y_j = \frac{1}{1+\exp\left(-v_j\right)} \tag{4.6}$$

kde v_j je úroveň vnútornej aktivity neurónu j a y_j je jeho výstup. Prítomnosť nelinearít je

veľmi dôležitá, pretože inak by sa vstupno-výstupná charakteristika siete dala zredukovať na jednovrstvový perceptrón.

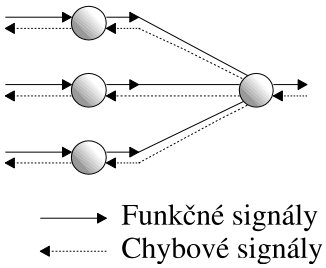
- Ďalšou charakteristikou je existencia jednej alebo viac *skrytých vrstiev*. Sú to práve skryté neuróny, ktoré umožnujú sieti naučiť sa komplexné úlohy, pretože vyberajú zo vstupných vzoriek najdôležitejšie príznaky. Názov skrytý neurón znamená, že takýto neurón nie je súčasťou vstupnej, ani výstupnej vrstvy, čiže zvonku siete je "neprístupný".
- Dôležitou charakteristikou je tiež *vysoký stupeň prepojenia* siete, určený synapsami siete.

Práve kombinácia týchto troch charakteristík spolu so schopnosťou učiť sa skúsenosťou robí viacvrstvový perceptrón tak výpočtovo silným prostriedkom. Tieto charakteristiky sú ale aj príčinou ťažkostí či nedostatkov v súčasnom stave poznatkov o správaní sa siete. *Distribuovaná forma nelinearity* a vysoká prepojenosť siete veľmi sťažuje teoretickú analýzu viacvrstvového perceptrónu. Existencia skrytých neurónov robí zase problémy pri vizualizácii procesu učenia.



Obr. 4.5 *Viacvrstvový perceptrón s dvomi skrytými vrstvami*

Na obr. 4.5 je znázornený viacvrstvový perceptrón s dvomi skrytými vrstvami. Zobrazená sieť je *úplne prepojená*, čo znamená, že každý neurón v ľubovoľnej vrstve siete je spojený so všetkými neurónmi v predchádzajúcej vrstve.



Obr. 4.6 *Šírenie funkčných a chybových signálov vo viacvrstvovom perceptróne*

Na obr. 4.6 je znázornená iba časť viacvrstvového perceptrónu. Môžeme tu vidieť dva druhy signálov: funkčné a chybové signály.

- *Funkčný signál* je signál, ktorý prichádza zo vstupu siete, šíri sa dopredne neurón po

neuróne až sa objaví na výstupe siete ako výstupný signál.

- *Chybový signál* vzniká vo výstupných neurónoch siete a šíri sa spätne vrstvu po vrstve po sieti (*iba* počas trénovania).

Každý skrytý a výstupný neurón viacvrstvového perceptrónu robí dva výpočty:

- Výpočet funkčného signálu, ktorý je výstupom neurónu a je vyjadrený ako nelineárna funkcia vstupných signálov a synaptických váh spojených s týmto neurónom.
- Výpočet okamžitého odhadu gradientového vektora, t.j. gradientu chybového povrchu vzhľadom na váhy pripojené k danému neurónu; gradientový vektor je potrebný pre fázu spätného šírenia chýb v sieti.

4.2.1 Algoritmus spätného šírenia

Chybový signál na výstupe neurónu j pre iteráciu n (čiže pri privedení n-tej trénovacej vzorky) je definovaný ako

$$e_j(n) = d_j(n) - y_j(n) \tag{4.7}$$

pričom predpokladáme, že neurón j je výstupným neurónom, $d_j(n)$ je žiadaná a $y_j(n)$ aktuálna odpoveď.

Môžeme zadefinovať okamžitú hodnotu kvadratickej chyby pre neurón j ako $\frac{1}{2}e_j^2(n)$. Keď spočítame takúto chybu pre všetky neuróny vo výstupnej vrstve, dostaneme *sumu okamžitých kvadratických chýb*

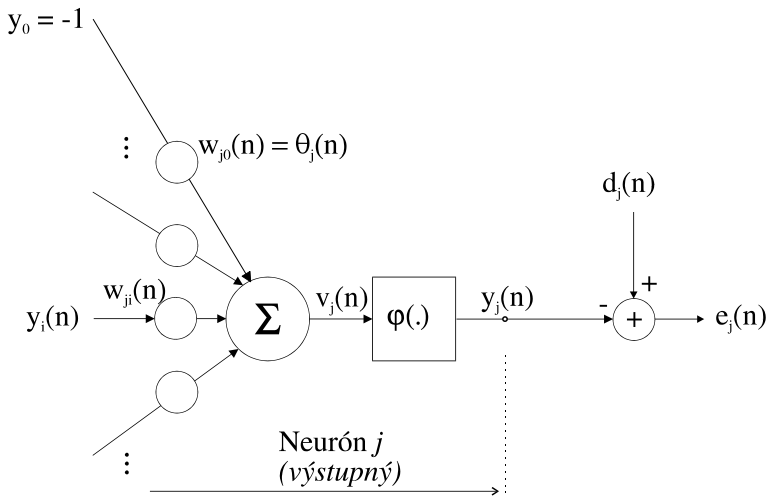
$$\varepsilon(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{4.8}$$

kde množina C zahŕňa všetky neuróny výstupnej vrstvy. Nech N je celkový počet vzoriek trénovacej množiny. Potom *priemernú kvadratickú chybu* dostaneme ako

$$\varepsilon_{av} = \frac{1}{N}\sum_{n=1}^{N} \varepsilon(n) \tag{4.9}$$

Táto chyba je funkciou voľných parametrov siete, t.j. váh a prahov siete. Pre danú trénovaciu množinu reprezentuje ε_{av} *účelovú (kriteriálnu) funkciu*. Cieľom učenia je minimalizovať ε_{av} vzhľadom na váhy. Budeme uvažovať učenie, pri ktorom sú váhy upravované vzorku po vzorke, t.j. na základe chýb počítaných pre každú vzorku privedenú do siete. Takto dostaneme *odhad* zmien váh, ktoré by vyplývali z výpočtu zmien váh na základe ε_{av} - čiže až po privedení celej trénovacej množiny.

Na obr. 4.7 je znázornený výstupný neurón j so vstupnými signálmi pochádzajúcimi od neurónov predchádzajúcej vrstvy.



Obr. 4.7 *Detail výstupného neurónu j*

Vnútorná aktivita neurónu j (vstup do nelinearity) je

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{4.10}$$

kde p je celkový počet vstupov neurónu j (prah nepočítame) a $w_{ji}(n)$ je synaptická váha spájajúca výstup neurónu i so vstupom neurónu j pri iterácii n. Funkčný signál na výstupe neurónu j pri iterácii n je

$$y_j(n) = \varphi_j\left(v_j(n)\right) \tag{4.11}$$

Algoritmus spätného šírenia upravuje váhu $w_{ji}(n)$ o hodnotu $\Delta w_{ji}(n)$, ktorá je úmerná okamžitému gradientu $\partial\varepsilon(n)/\partial w_{ji}(n)$. Podľa reťazového pravidla (pravidlo pre výpočet parciálnej derivácie zloženej funkcie) môžeme tento gradient vyjadriť nasledovne:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{4.12}$$

Gradient $\partial\varepsilon(n)/\partial w_{ji}(n)$ určuje smer pohybu po chybovom povrchu pre váhu $w_{ji}(n)$.
Diferencovaním oboch strán rovnice (4.8) podľa $e_j(n)$ dostaneme

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n) \tag{4.13}$$

Diferencovaním oboch strán rovnice (4.7) podľa $y_i(n)$ dostaneme

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{4.14}$$

Podobne diferencovaním rovnice (4.11) podľa $v_j(n)$ dostaneme

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'\left(v_j(n)\right) \tag{4.15}$$

kde čiarka v hornom indexe znamená diferencovanie podľa argumentu.
Nakoniec diferencovaním (4.10) podľa $w_{ji}(n)$ dostaneme

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{4.16}$$

Dosadenie rovníc (4.13) až (4.16) do (4.12) vedie k vzťahu

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'\left(v_j(n)\right)y_i(n) \tag{4.17}$$

Úprava $\Delta w_{ji}(n)$ sa na váhu $w_{ji}(n)$ aplikuje podľa *delta pravidla*

$$\Delta w_{ji}(n) = -\eta\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{4.18}$$

kde η je konštanta určujúca rýchlosť učenia - nazýva sa *parameter rýchlosti učenia* algoritmu spätného šírenia. Použitie znamienka mínus v (4.18) znamená gradientový zostup po chybovom povrchu. Dosadenie (4.17) do (4.18) vedie k vzťahu

$$\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n) \tag{4.19}$$

kde $\delta_j(n)$ je *lokálny gradient* definovaný ako

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi_j'\left(v_j(n)\right) \tag{4.20}$$

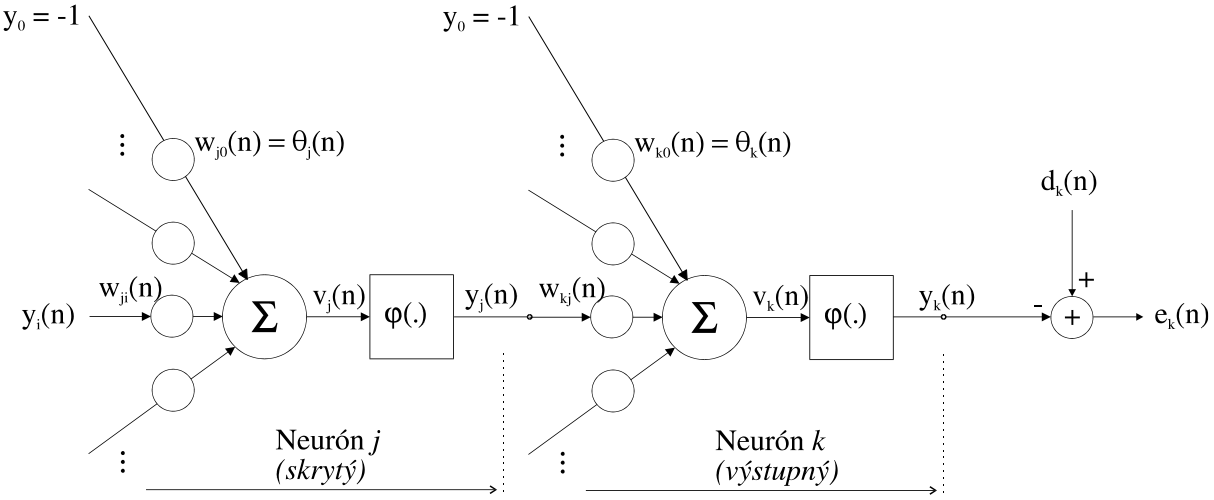
Rovnice (4.19) a (4.20) poukazujú na kľúčovú úlohu chybového signálu $e_j(n)$ pri výpočte úpravy váhy $\Delta w_{ji}(n)$. Tu môžu nastať dva prípady. Buď neurón j je výstupným neurónom a nevzniká žiadny problém, pretože žiadaná odpoveď je pre výstupné neuróny priamo dostupná, alebo neurón j je skrytým neurónom a žiadaná odpoveď je preň nedostupná. Hoci žiadanú odpoveď v tomto prípade nepoznáme, každý skrytý neurón nesie svoju časť zodpovednosti za chyby na výstupoch siete. Otázkou je, ako dať skrytým neurónom najavo, že nesú zodpovednosť za nesprávne či nepresné výstupy siete. Riešenie poskytuje algoritmus spätného šírenia.

Prípad I: Neurón j je výstupným neurónom

V tomto prípade môžeme neurónu priviesť žiadanú odpoveď. Pre výpočet chybového signálu $e_j(n)$ môžeme použit rovnicu (4.7). Po výpočte chybového signálu je jednoduché určiť lokálny gradient $\delta_j(n)$ podľa (4.20).

Prípad II: Neurón j je skrytým neurónom

V tomto prípade žiadanú odpoveď pre neurón nepoznáme. Chybový signál pre skrytý neurón musí byť vypočítaný rekurzívne podľa chybových signálov neurónov, s ktorými je priamo spojený. Uvažujme situáciu podľa obr. 4.8, kde neurón j je skrytým neurónom.



Obr. 4.8 *Detail výstupného neurónu k spojeného so skrytým neurónom j*

Podobne ako v (4.20) je lokálny gradient $\delta_j(n)$ pre skrytý neurón j

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \varepsilon(n)}{\partial y_j(n)} \varphi_j'(v_j(n)) \tag{4.21}$$

kde sme využili (4.15). Výpočet parciálnej derivácie $\partial \varepsilon(n) / \partial y_j(n)$ môže byť urobený nasledovne:
Z obr. (4.8) vyplýva, že

$$\varepsilon(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n) \tag{4.22}$$

kde neurón k je výstupný. Diferencovaním rovnice (4.22) podľa $y_j(n)$ dostávame

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} \tag{4.23}$$

Použitím reťazového pravidla dostaneme

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{4.24}$$

Z obr. 4.8 je zrejmé, že

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k\left(v_k(n)\right) \tag{4.25}$$

čiže

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'\left(v_k(n)\right) \tag{4.26}$$

Z obr. 4.8 je tiež vidieť, že

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n) \tag{4.27}$$

kde q je celkový počet vstupov (prah nepočítajúc) neurónu k.
Diferencovanie (4.27) podľa $y_j(n)$ vedie k vzťahu

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{4.28}$$

Použitím (4.26) a (4.28) určíme žiadanú parciálnu deriváciu

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = -\sum_k e_k(n) \varphi_k'\left(v_k(n)\right) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n) \tag{4.29}$$

kde sme použili definíciu lokálneho gradientu $\delta_j(n)$ podľa (4.20) s indexom k namiesto j.
Dosadením (4.29) do (4.21) dostaneme lokálny gradient $\delta_j(n)$ pre skrytý neurón j

$$\delta_j(n) = \varphi_j'\left(v_j(n)\right) \sum_k \delta_k(n) w_{kj}(n) \tag{4.30}$$

Člen $\varphi_j'\left(v_j(n)\right)$ zahrnutý v (4.30) pre výpočet lokálneho gradientu $\delta_j(n)$ závisí iba od

aktivačnej funkcie skrytého neurónu j. Členy v sumácii cez k závisia od dvoch faktorov. Členy $\delta_k(n)$ vyžadujú znalosť všetkých chybových signálov $e_k(n)$ ktoré ležia v najbližšej nasledujúcej skrytej vrstve neurónu j a sú s ním spojené. Členy $w_{kj}(n)$ opisujú synaptické váhy týchto spojení.

Zhrnutím výsledkov, ktoré sme dostali pre algoritmus spätného šírenia, sú nasledovné skutočnosti:
Úprava $\Delta w_{ji}(n)$ pre váhu spojenia neurónu i a neurónu j je definovaná delta pravidlom

$$\begin{pmatrix}\text{úprava}\\ \text{váhy}\\ \Delta w_{ji}(n)\end{pmatrix} = \begin{pmatrix}\text{parameter}\\ \text{rýchlosti učenia}\\ \eta\end{pmatrix} \cdot \begin{pmatrix}\text{lokálny}\\ \text{gradient}\\ \delta_j(n)\end{pmatrix} \cdot \begin{pmatrix}\text{vstupný signál}\\ \text{neurónu j}\\ y_i(n)\end{pmatrix} \qquad (4.31)$$

(vstupný signál neurónu j je zároveň výstupným signálom neurónu i).
Lokálny gradient $\delta_j(n)$ závisí od toho, či neurón j je skrytý alebo výstupný:

- Ak je neurón j výstupný, potom $\delta_j(n)$ sa rovná súčinu derivácie $\varphi_j'(v_j(n))$ a chybového signálu $e_j(n)$ podľa rovnice (4.20)
- Ak je neurón j skrytý, potom $\delta_j(n)$ sa rovná súčinu derivácie $\varphi_j'(v_j(n))$ a váhovanej sume lokálnych gradientov pre neuróny v ďalšej skrytej alebo výstupnej vrstve, ktoré sú spojené s neurónom j podľa rovnice (4.30).

Dve fázy výpočtu

Pri použití algoritmu spätného šírenia sú dve fázy výpočtu: dopredná a spätná.

Pri *doprednej fáze* sa synaptické váhy nemenia, funkčné signály siete sa počítajú postupne neurón po neuróne. Funkčný signál na výstupe neurónu j je

$$y_j(n) = \varphi(v_j(n)) \qquad (4.32)$$

kde $v_j(n)$ je úroveň vnútornej aktivity neurónu j definovaná ako

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \qquad (4.33)$$

kde p je celkový počet vstupov neurónu j (prah nepočítame) a $w_{ji}(n)$ je synaptická váha spájajúca výstup neurónu i so vstupom neurónu j a $y_i(n)$ je vstupný signál neurónu j (výstupný signál neurónu i). Ak je neurón j v prvej skrytej vrstve siete, potom sa index i vzťahuje na i-ty vstupný prvok siete, čiže

$$y_i(n) = x_i(n) \qquad (4.34)$$

kde $x_i(n)$ je i-ty prvok vstupnej vzorky. Ak je neurón j vo výstupnej vrstve siete, potom sa index j vzťahuje na j-ty výstupný prvok siete, čiže

$$y_j(n) = o_j(n) \tag{4.35}$$

kde $o_j(n)$ je j-ty prvok výstupnej vzorky. Tento výstup sa porovná so žiadanou odpoveďou a takto prídeme k chybovému signálu $e_j(n)$ pre výstupný neurón j. Dopredná fáza výpočtu teda začína v prvej skrytej vrstve privedením vstupného vektora a končí sa vo výstupnej vrstve určením chybových signálov pre všetky výstupné neuróny.

Spätná fáza výpočtu sa začína vo výstupnej vrstve siete, z ktorej sa chybové signály šíria spätne postupne vrstvu po vrstve a rekurzívne sa počíta lokálny gradient δ pre každý neurón. Tento rekurzívny proces umožní úpravu váh podľa delta pravidla uvedeného v rovnici (4.31). Pre neurón výstupnej vrstvy sa δ rovná súčinu chybového signálu a prvej derivácie jeho nelinearity. Vzťah (4.31) je teda priamo použiteľný pre úpravu váh všetkých spojení vedúcich do výstupnej vrstvy. Keď sú už známe lokálne gradienty pre neuróny vo výstupnej vrstve, môžeme použiť (4.30) pre výpočet lokálnych gradientov pre neuróny najbližšej skrytej vrstvy, a teda vypočítať aj zmeny všetkých váh spojení výstupnej a najbližšej skrytej vrstvy. Takýto rekurzívny výpočet pokračuje postupne pre ďalšie vrstvy, kde sú už vždy zohľadnené zmeny váh urobené v predchádzajúcich vrstvách.

Sigmoidálna nelinearita

Výpočet lokálnych gradientov δ pre neuróny viacvrstvového perceptrónu vyžaduje derivácie aktivačných funkcií neurónov. Jedinou podmienkou, ktorú vyžadujeme pre aktivačnú funkciu je *diferencovateľnosť*. Príkladom spojito diferencovateľnej nelineárnej funkcie najčastejšie využívanej pre viacvrstvový perceptrón je *sigmoidálna nelinearita*, ktorej reprezentantom je napr. *logistická funkcia*. Pre neurón j jej tvar je

$$y_j(n) = \varphi_j\left(v_j(n)\right) = \frac{1}{1+\exp\left(-v_j(n)\right)}, \qquad -\infty < v_j(n) < \infty \tag{4.36}$$

Rozsah funkčných hodnôt tejto funkcie je $0 \le y_j \le 1$. Iným príkladom je hyperbolický tangens, ktorý je vzhľadom k počiatku antisymetrický s rozsahom hodnôt $-1 \le y_j \le 1$.

Diferencovaním oboch strán rovnice (4.36) podľa $v_j(n)$ dostávame

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'\left(v_j(n)\right) = \frac{\exp\left(-v_j(n)\right)}{\left[1+\exp\left(-v_j(n)\right)\right]^2} \tag{4.37}$$

Ak použijeme vzťah (4.36) na elimináciu člena $\exp\left(-v_j(n)\right)$ z rovnice (4.37), dostaneme

$$\varphi_j'\left(v_j(n)\right) = y_j(n)\left[1-y_j(n)\right] \tag{4.38}$$

Pre výstupný neurón j platí $y_j(n) = o_j(n)$. Lokálny gradient pre takýto neurón je

$$\begin{aligned}\delta_j(n) &= e_j(n)\varphi_j'(v_j(n)) \\ &= \left[d_j(n) - o_j(n)\right] o_j(n)\left[1 - o_j(n)\right]\end{aligned} \quad \text{pre výstupný neurón } j \qquad (4.39)$$

kde $o_j(n)$ je výstup neurónu j a $d_j(n)$ je jeho žiadaná odpoveď. Pre skrytý neurón j môžeme lokálny gradient vyjadriť v tvare

$$\begin{aligned}\delta_j(n) &= \varphi_j'(v_j(n))\sum_k \delta_k(n) w_{kj}(n) \\ &= y_j(n)\left[1 - y_j(n)\right]\sum_k \delta_k(n) w_{kj}(n)\end{aligned} \quad \text{pre skrytý neurón } j \qquad (4.40)$$

Derivácia $\varphi_j'(v_j(n))$ má podľa rovnice maximum (rovnajúce sa 0.25) pre $y_j(n) = 0.5$ a minimálnu hodnotu rovnajúcu sa nule pre $y_j(n) = 0$ a $y_j(n) = 1$. Pretože zmena váhy je úmerná derivácii $\varphi_j'(v_j(n))$, najviac sú upravované váhy tých neurónov so sigmoidálnou nelinearitou, ktorých výstupné hodnoty sú v strede svojho rozsahu. Práve táto vlastnosť algoritmu spätného šírenia prispieva k jeho stabilite.

Rýchlosť učenia

Algoritmus spätného šírenia znamená pohyb po chybovom povrchu v priestore váh podľa metódy najstrmšieho zostupu. Ak nastavíme parameter rýchlosti učenia η na veľmi malú hodnotu, dôsledkom budú malé zmeny váh siete a trajektória vo váhovom priestore bude hladká. V tomto prípade však bude učenie siete veľmi pomalé. Ak zvolíme parameter η príliš veľký, zmeny váh budú veľké, čo však môže viesť k nestabilite.

Jednoduchou metódou pre zvýšenie rýchlosti učenia a zároveň predídenie nestabilite je momentová technika - zahrnutie *momentového člena* (“momentum term”) do rovnice (4.19), čo vedie k vzťahu

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1) \qquad (4.41)$$

kde α je zvyčajne kladné číslo nazývané *momentová konštanta*. Pre konvergenciu učenia má platiť $0 \le |\alpha| < 1$. Pre $\alpha = 0$ algoritmus spätného šírenia pracuje bez momentovej techniky, záporné α sa nezvykne využívať.

Dôsledkom využitia momentovej techniky je, že ak smer zostupu po chybovom povrchu $\partial\varepsilon/\partial w_{ji}$ má rovnaké znamienko po viac iterácií, tak úprava váhy bude veľká, čiže momentový člen urýchľuje zostup po chybovom povrchu. Ak $\partial\varepsilon/\partial w_{ji}$ počas iterácií začne striedať znamienko, úprava váhy bude malá, čo má stabilizačný efekt pri oscilujúcich zmenách smeru pohybu po chybovom povrchu.

Momentový člen môže čiastočne aj zabrániť procesu učenia skončiť v lokálnom minime, toto

však nemusí byť pravidlom.

Poznámka: V predchádzajúcich úvahách sme parameter rýchlosti učenia η považovali za konštantu. V prípade potreby možno byť tento parameter zadefinovaný ako η_{ji}, t.j. rôzny pre rozličné spojenia, prípadne rôzny pre rôzne časti siete. Tiež je možné počas učenia niektoré váhy neupravovať.

Vzorkový a dávkový mód učenia

Pri použití *vzorkového módu* učenia sú váhy upravované po každej prezentácii vzorky. Nech po privedení n-tej vzorky je $\Delta w_{ji}(n)$ zmena váhy w_{ji}. Potom zmena váhy spriemerovaná cez celú trénovaciu množinu s N vzorkami je

$$\Delta \hat{w}_{ji} = \frac{1}{N}\sum_{n=1}^{N}\Delta w_{ji}(n) = -\frac{\eta}{N}\sum_{n=1}^{N}\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -\frac{\eta}{N}\sum_{n=1}^{N}e_j(n)\frac{e_j(n)}{\partial w_{ji}(n)} \qquad (4.42)$$

kde sme využili rovnice (4.18) a (4.8).

Pri *dávkovom móde* učenia sa váhy upravujú až po privedení všetkých vzoriek trénovacej množiny, t.j. po každej epoche (*epocha* je privedenie úplnej trénovacej množiny počas učenia). V tomto prípade je zmena váhy [Hay94]

$$\Delta w_{ji} = -\frac{\eta}{N}\sum_{n=1}^{N}e_j(n)\frac{e_j(n)}{\partial w_{ji}} \qquad (4.43)$$

Z predchádzajúcich dvoch rovníc vidíme, že priemerná zmena váhy $\Delta \hat{w}_{ji}$ je iba odhadom pre Δw_{ji} dávkového módu.

Najčastejšie sa využíva vzorkový mód činnosti, pretože vyžaduje menej pamäte pre každú váhu, v tomto móde máme tiež možnosť sieti prezentovať vzorky v náhodnom poradí - pohyb po chybovom povrchu je v tomto prípade stochastický, čo môže zamedziť ukončeniu učenia v lokálnom minime. Dávkový mód činnosti však umožňuje presne určiť gradientový vektor. Výber módu teda závisí od konkrétnej úlohy.

Kritérium pre ukončenie učenia

Vo všeobecnosti sa nedá ukázať, že algoritmus spätného šírenia skonvergoval a tiež nie sú dobre definované kritériá pre ukončenia učenia. Existuje niekoľko “rozumných” kritérií pre ukončenie úpravy váh. Ich zhrnutie je uvedené napr. v [Hay94].

Tu uvedieme veľmi jednoduché, ale zároveň veľmi silné a mimoriadne často používané kritérium: Po každej iterácii (prípadne po každej epoche trénovania) sa sieť testuje z hľadiska generalizácie. Učenie sa skončí, keď sieť už generalizuje primerane našim požiadavkám, prípadne schopnosť siete generalizovať dosiahla svoje maximum, viď kap. 3.2 (rozdelenie dostupných údajov na trénovaciu a testovaciu množinu), takýto postup sa nazýva aj vzájomné

overovanie (“cross-validation”).

4.2.2 Zhrnutie algoritmu spätného šírenia

Učenie viacvrstvového perceptrónu algoritmom spätného šírenia je ilustrované na obr. 4.9. Zahŕňa doprednú fázu výpočtu a aj fázu spätného šírenia chýb.
Označenie v tomto obrázku je nasledovné:

- $\mathbf{w}^{(l)}$ váhový vektor neurónu vo vrstve l
- $\theta^{(l)}$ prah neurónu vo vrstve l
- $\mathbf{v}^{(l)}$ vektor vnútorných aktivít neurónov vo vrstve l
- $\mathbf{y}^{(l)}$ vektor funkčných signálov neurónov vo vrstve l
- l index vrstvy, $l = 0 \ldots L$, $l = 0$ pre vstupnú a $l = L$ pre výstupnú vrstvu
- $?^{(l)}$ vektor lokálnych gradientov neurónov vo vrstve l
- $\mathbf{e}$ chybový vektor, jeho prvky sú $e_1, e_2, \ldots, e_q$

Uvažujme vzorkový mód činnosti. Pre trénovacie údaje $\{[\mathbf{x}(n), \mathbf{d}(n)], n = 1,2,\ldots,N\}$ algoritmus prechádza nasledovnými cyklami:

Krok 1 - inicializácia: Nastav váhy a prahy siete na malé náhodné čísla.

Krok 2 - prezentácia trénovacích údajov: Priveď do siete trénovacie príklady, ktoré tvoria jednu epochu. Pre každý príklad rob postupnosť dopredných a spätných výpočtov podľa bodov 3 a 4.

Krok 3 - dopredný výpočet: Dodaj sieti trénovací príklad $[\mathbf{x}(n), \mathbf{d}(n)]$ Počítaj funkčné signály siete pre jednotlivé neuróny:

$$v_j^{(l)}(n) = \sum_{i=0}^{p} w_{ji}^{(l)}(n) y_i^{(l-1)}(n) \tag{4.44}$$

kde pre $i = 0$ je $y_0^{(l-1)}(n) = -1$ a $w_{j0}^{(l)}(n) = \theta_j^{(l)}(n)$. Ak je sigmoidálnou nelinearitou logistická funkcia, potom

$$y_j^{(l)}(n) = \frac{1}{1 + \exp\left(-v_j^{(l)}(n)\right)} \tag{4.45}$$

Ak $l = 0$, potom nastav $y_j^{(0)}(n) = x_j(n)$, ak $l = L$, potom nastav $y_j^{(L)}(n) = o_j(n)$.
Potom počítaj chybový signál

$$e_j(n) = d_j(n) - o_j(n) \tag{4.46}$$

Krok 4 - spätná fáza výpočtu: Počítaj lokálne gradienty δ siete, postupne od výstupnej smerom k vstupnej vrstve:

$$\delta_j^{(L)}(n) = e_j^{(L)}(n) o_j(n)[1 - o_j(n)], \tag{4.47}$$

pre neurón j vo výstupnej vrstve L

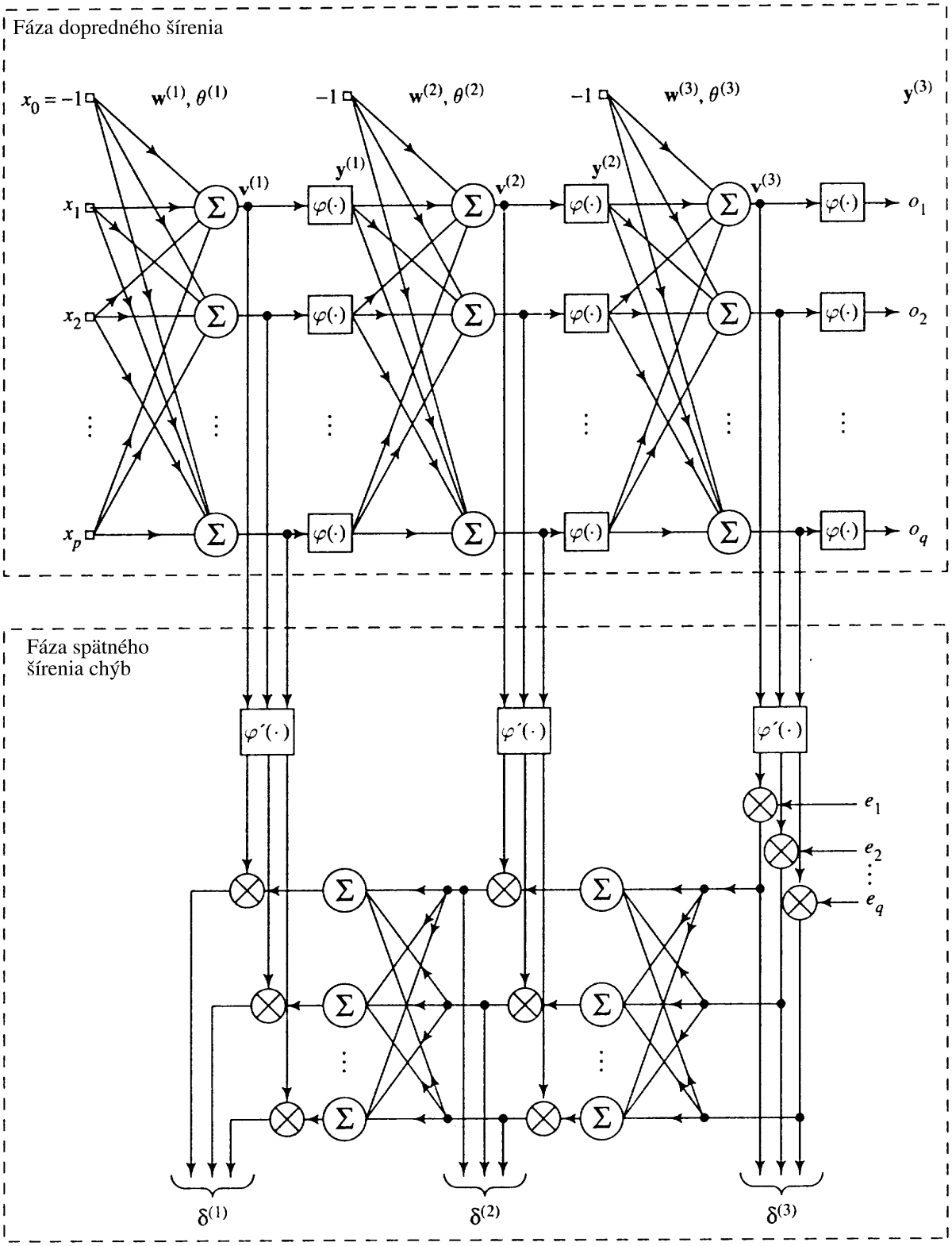
$$\delta_j^{(l)}(n) = y_j^{(l)}(n)\left[1 - y_j^{(l)}(n)\right] \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n), \tag{4.48}$$

pre neurón j v skrytej vrstve l

Uprav váhy vo vrstve l:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta\delta_j^{(l)}(n)y_i^{(l-1)}(n) + \alpha\left[w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)\right] \qquad (4.49)$$

kde η je parameter rýchlosti učenia a α je momentová konštanta.



Obr. 4.9 *Dopredná a spätná fáza výpočtu pre algoritmus spätného šírenia [Hay94]*

Krok 5 - iterovanie: Iteruj výpočet prezentovaním nových epoch trénovacích príkladov, pokiaľ sa váhy siete nestabilizujú a priemerná kvadratická chyba ε_{av} je v minime alebo v akceptovateľne malej hodnote. Poradie prezentácie trénovacích príkladov by sa malo pre rôzne epochy náhodne meniť. Momentová konštanta a parameter rýchlosti učenia sa s rastúcim počtom iterácií upravujú (zvyčajne znižujú).

4.2.3 Tipy pre zlepšenie činnosti algoritmu spätného šírenia

Často sa hovorí, že návrh neurónovej siete využitím algoritmu spätného šírenia je viac umenie ako veda - mnoho faktorov pri trénovaní siete je výsledkom osobnej skúsenosti. Aj napriek tomu však existuje zopár možných spôsobov, ako zlepšiť činnosť tohto algoritmu:

- Sigmoidálna aktivačná funkcia môže byť nesymetrická - použitie funkcie hyperbolický tangens môže niekedy viesť k lepším výsledkom.
- Hodnoty žiadaných odpovedí pre neuróny výstupnej vrstvy môžu byť z intervalu, ktorý má hranice posunuté o primeranú hodnotu μ od hraníc pôvodného rozsahu. Napr. pre logistickú funkciu namiesto intervalu (0,1) využijeme (μ ,1- μ).
- Inicializácia váh a prahov siete by mala byť urobená v hodnotách rovnomerne rozložených v malom rozsahu.
- Parameter rýchlosti učenia η môže mať menšiu hodnotu pre posledné vrstvy siete v porovnaní s prvými vrstvami. Platí tiež, že neuróny s mnohými vstupmi by mali mať menšiu hodnotu η ako neuróny s malým počtom vstupov.
- Pre on-line činnosť veľkých sietí je vhodnejší vzorkový mód činnosti, čo vedie k rýchlejšiemu trénovaniu.
- Poradie vzoriek pre jednotlivé epochy trénovania by sa malo náhodne meniť.

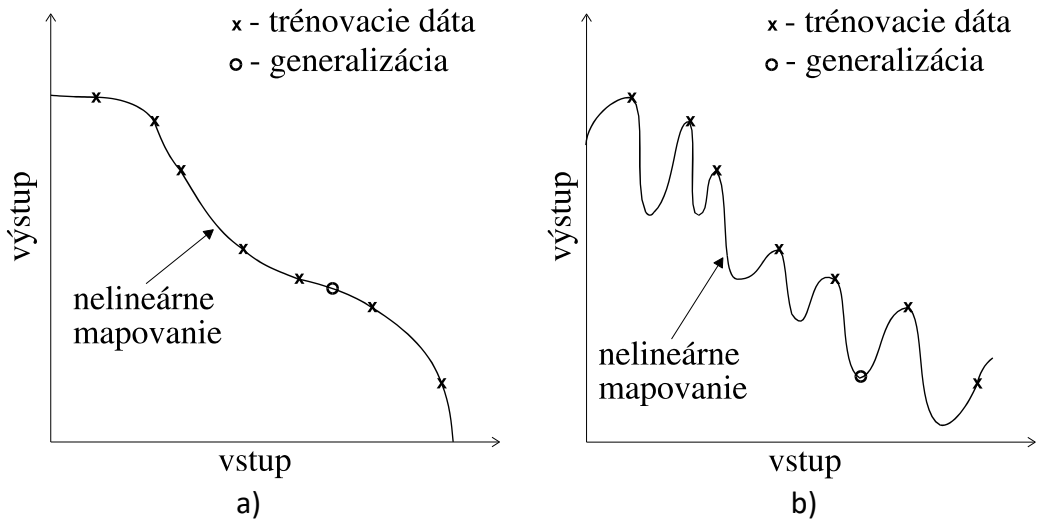
4.2.4 Generalizácia

Pri trénovaní siete algoritmom spätného šírenia očakávame, že sieť zgeneralizuje, t.j. vstupno-výstupné správanie siete bude správne pre vstupno-výstupné testovacie dáta, ktoré neboli použité počas trénovania. Termín *generalizácia* pochádza z psychológie.

Sieť môžeme považovať za nelineárne vstupno-výstupné mapovanie. Z tohto pohľadu sa môžeme pozerať na generalizáciu nie ako na mystickú vlastnosť neurónových sietí, ale ako na efekt dobrej nelineárnej interpolácie vstupných dát.

Na obr. 4.10a sú ilustrované dôsledky generalizácie siete. Nelineárne vstupno-výstupné mapovanie znázornené krivkou je výsledkom učenia bodov označených ako trénovacie dáta. Pod pojmom generalizácia teda rozumieme interpoláciu neurónovou sieťou. Ak sa však neurónová sieť učí príliš veľa špecifických vstupno-výstupných párov, sieť môže začať memorovať (učiť sa naspamäť) tieto špecifické páry a nebude schopná generalizovať podobné vstupno-výstupné páry. V tomto prípade je sieť pretrénovaná (často k tomuto javu dochádza, keď sieť obsahuje viac skrytých neurónov, než je skutočne potrebné). Takýto prípad je znázornený na obr. 4.10b pre rovnaké trénovacie dáta ako v obr. 4.10a. Memorovanie je vlastne vyhľadávacia tabuľka (“lookup table”), čoho dôsledkom je, že vstupno-výstupné

mapovanie nie je hladké.



Obr. 4.10 *Správna (a) a nesprávna (b) generalizácia*

4.2.5 Aproximácia funkcií

Na viacvrstvový perceptrón trénovaný algoritmom spätného šírenia sa môžeme pozerať ako na nástroj umožňujúci implementovať nelineárne vstupno-výstupné mapovanie. Nech p znamená počet vstupných prvkov viacvrstvového perceptrónu a q je počet neurónov vo výstupnej vrstve. Viacvrstvový perceptrón teda reprezentuje mapovanie z p-rozmerného euklidovského priestoru vstupov do q-rozmerného euklidovského priestoru výstupov (toto mapovanie je nekonečne veľa krát spojito diferencovateľné). Otázkou je, aký je minimálny počet skrytých vrstiev, aby viacvrstvový perceptrón implementoval ľubovoľné spojité mapovanie.

4.2.5.1 Univerzálna aproximačná teoréma

Univerzálna aproximačná teoréma je zosumarizovaná v [Hay94], opiera sa o práce [Cyb89], [Hor89], [Fun89].

Nech $\varphi(\cdot)$ je nekonštantná, ohraničená, monotónne rastúca, spojitá funkcia. Nech I_p je p-rozmerná jednotková hyperkocka $[0,1]^p$. Nech $C(I_p)$ znamená priestor spojitých funkcií na I_p. Potom ak je daná ľubovoľná funkcia $f \in C(I_p)$ a $\varepsilon > 0$, tak existuje celé číslo M a množina reálnych konštánt $\alpha_i, \theta_i, w_{ij}$, kde $i = 1, \cdots, M$ a $j = 1, \ldots, p$ takých, že môžeme definovať

$$F(x_1, \ldots, x_p) = \sum_{i=1}^{M} \alpha_i \varphi\left(\sum_{j=1}^{p} w_{ij} x_j - \theta_i \right) \tag{4.50}$$

ako aproximáciu funkcie $f(\cdot)$*; t.j.*

$$\left|F(x_1,\ldots,x_p) - f(x_1,\ldots,x_p)\right| < \varepsilon \qquad (4.51)$$

pre všetky $\{x_1,\ldots x_p\} \in I_p$.

Táto teoréma ja priamo aplikovateľná na viacvrstvový perceptrón. Môžeme konštatovať, že logistická funkcia $1/[1+\exp(-v)]$ použitá ako nelinearita pre model neurónu viacvrstvového perceptrónu je nekonštantná, ohraničená, monotónne rastúca, spojitá funkcia, a teda spĺňa podmienky kladené na funkciu $\varphi(\cdot)$. Rovnica (4.50) reprezentuje výstup viacvrstvového perceptrónu opísaného nasledovne:

- Sieť má p vstupných prvkov, jednu skrytú vrstvu s M neurónmi, vstupy sú $\{x_1,\ldots x_p\}$.
- Skrytý neurón i má synaptické váhy $\{w_{i1},\ldots,w_{ip}\}$ a prah θ_i.
- Výstup siete je lineárna kombinácia výstupov skrytých neurónov, kde $\alpha_1,\ldots,\alpha_M$ definujú koeficienty tejto kombinácie.

Univerzálna aproximačná teoréma je *existenčná* teoréma - poskytuje matematické potvrdenie aproximácie ľubovoľnej spojitej funkcie. Rovnica (4.50) zovšeobecňuje aproximáciu konečnými Fourierovymi radmi. K univerzálnej aproximačnej teoréme je možné urobiť nasledujúci komentár:

- Táto teoréma hovorí, že jedna skrytá vrstva je pre viacvrstvový perceptrón postačujúca na aproximáciu (v medziach presnosti ε) danej trénovacej množiny reprezentovanej množinou vstupov $x_1,\ldots,x_p$ a žiadaným výstupom $f(x_1,\ldots,x_p)$.
- Teoréma nehovorí, či je jedna skrytá vrstva optimálna (vzhľadom na proces učenia, či jednoduchosť implementácie)
- Teoréma je existenčným dôkazom; čiže nehovorí, ako pre danú aproximáciu viacvrstvový perceptrón skonštruovať
- Teoréma predpokladá, že funkcia, ktorú ideme aproximovať, je daná a že máme k dispozícii skrytú vrstvu s neobmedzeným počtom neurónov. Tieto predpoklady sú pre väčšinu aplikácií viacvrstvových perceptrónov porušené.

Problém s viacvrstvovým perceptrónom s jednou skrytou vrstvou spočíva v tom, že v takomto prípade je interakcia neurónov globálna - je ťažké vylepšiť aproximáciu v jednom bode bez zhoršenia aproximácie v inom bode [Hay94]. *Viacvrstvový perceptrón s dvoma skrytými vrstvami* sa môže z hľadiska aproximácie správať lepšie:

- Lokálne príznaky sú extrahované v prvej skrytej vrstve. Niektoré neuróny prvej skrytej vrstvy delia vstupný priestor do oblastí (podpriestorov) a iné neuróny tejto vrstvy sa učia lokálne príznaky charakterizujúce tieto regióny.
- Globálne príznaky sa extrahujú v druhej skrytej vrstve. Neuróny druhej skrytej vrstvy kombinujú výstupy neurónov prvej skrytej vrstvy pracujúcich na podpriestore vstupných dát a učia sa globálne príznaky pre tento podpriestor; mimo neho nereagujú.

Aproximačné vlastnosti viacvrstvových dopredných sietí sa neustále študujú, viď napr. [Hor91], [Kůr92].

4.2.5.2 Univerzálna aproximácia

Aproximácia funkcie pomocou viacvrstvového perceptrónu trénovaného algoritmom spätného šírenia s jediným výstupným prvkom sa môže zapísať v nasledovnom kompaktnom tvare:

$$F(\mathbf{x},\mathbf{w}) = \varphi\left(\sum_j w_{oj}\,\varphi\left(\sum_k w_{jk}\,\varphi\left(\ldots\varphi\left(\sum_i w_{li}x_i\right)\ldots\right)\right)\right) \tag{4.52}$$

kde $\varphi(\cdot)$ je sigmoidálna aktivačná funkcia, w_{oj} je synaptická váha z neurónu j v poslednej skrytej vrstve do výstupného neurónu o a tak ďalej pre iné synaptické váhy, x_i je i-ty prvok vstupného vektora $\mathbf{x}$. Váhový vektor $\mathbf{w}$ sa vzťahuje na celú množinu synaptických váh zoradených podľa vrstvy, potom podľa neurónov vo vrstve, a potom podľa poradového čísla váhy v neuróne. Schéma do seba vnorených nelineárnych funkcií (v prípade sigmoidálnych funkcií nazývaná "nested sigmoidal scheme") podľa (4.52) je pre klasickú aproximačnú teóriu neobvyklá. Takáto schéma je teda *univerzálnym aproximátorom* - viacvrstvový perceptrón trénovaný algoritmom spätného šírenia môže aproximovať ľubovoľnú spojitú funkciu do požadovaného stupňa presnosti za predpokladu, že k dispozícii je dostatočne veľa skrytých neurónov.