Preceptron 2

Pavel Křížek Přemysl Šůcha

6. přednáška ze dne 3.4.2001

Obsah

1 Lineární diskriminační funkce

Pro zopakování: Lineární diskriminační funkce je tvořena lineární kombinací vstupního vektoru příznaků $\mathbf{x}$ se složkami x_i. Můžeme ji tedy zapsat jako vážený součet

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i = w_0 + \mathbf{w}^T \mathbf{x}, \tag{1}$$

kde w_0 prahovací váha (*bias*) a $\mathbf{w}$ je váhový vektor se složkami w_i. Rovnice $g(\mathbf{x}) = 0$ je pak rovnicí nadplochy, jejíž orientace v prostoru je dána váhovým vektorem $\mathbf{w}$, který tvoří její normálu. Orientovaná vzdálenost nadplochy vůči počátku souřadnic odpovídá podílu $\frac{-w_0}{\|\mathbf{w}\|}$.

Pro případ klasifikace vzorků do dvou tříd ω_1 a ω_2 nám lineární diskriminační funkce dává následující pravidlo: Rozhodne se pro třídu ω_1, když $g(\mathbf{x}) \geq 0$ a pro třídu ω_2, když $g(\mathbf{x}) < 0$.

1.1 Zobecněná lineární diskriminační funkce

Zobecnění lineární diskriminační funkce provedeme tak, že využijeme možnosti mapování souřadnic vektoru $\mathbf{x}$ do prostoru s vyšší dimenzí, tedy

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{\hat{d}} w_i y_i(\mathbf{x}) = w_0 + \mathbf{w}^T \mathbf{y}, \tag{2}$$

kde w_i je $\hat{d}$-rozměrný váhový vektor a funkce $y_i(\mathbf{x})$ mohou být libovolné funkce $\mathbf{x}$. Ty pak provádějí mapování z d-rozměrného prostoru $\mathbf{x}$ do $\hat{d}$-rozměrného prostoru $\mathbf{y}$. Výsledná diskriminační funkce pak není lineární v $\mathbf{x}$, ale zato ve $\mathbf{y}$. Vhodným zvolením funkcí $y_i(\mathbf{x})$ a velikosti prostoru $\hat{d}$ může realizovat jakoukoli diskriminační funkci.

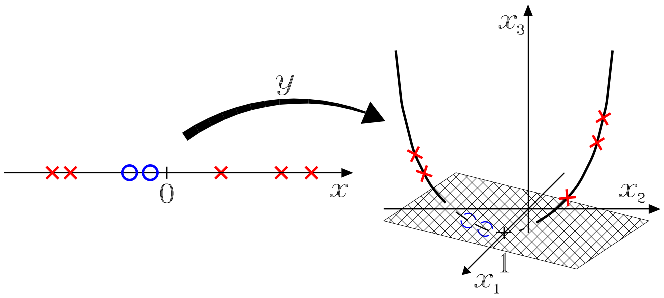
Příklad 1:
Uvažujme jednorozměrný prostor a na něm nelineárně rozložená data, viz obrázek 1. Je vidět, že na přímce není možné sestrojit lineární diskriminační funkci tak, aby správně klasifikovala obě třídy. Zvolíme-li však mapování

$$\mathbf{y} = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix},$$

promítne se přímka do paraboly ve trojrozměrném prostoru. Diskriminační funkci můžeme nyní zapsat ve tvaru

$$g(x) = w_0 y_0 + w_1 y_1 + w_2 y_2 = w_0 + w_1 x + w_2 x^2.$$

Nadplocha, která rozděluje prostor na dvě oblasti, je pak daná rovnicí $\mathbf{w}^T\mathbf{y} = 0$ a klasifikace dat do tříd je dána znaménkem diskriminační funkce.



Obrázek 1: Geometrická reprezentace mapování

1.2 Učení klasifikátoru

Učení klasifikátoru spočívá v nalezení vah w_i. Funkce $\mathbf{y}$ jsou fixní a je nutné je znát předem. Váhy w_i lze určit např. perceptronovým algoritmem (viz zápis 5. přednášky), pomocí SVM (viz kapitola 2), minimalizací střední kvadratické chyby na trénovacích datech apod.

2 Support Vector Machines

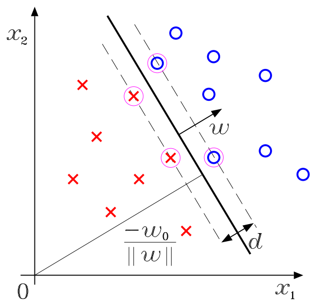
Pro zjednodušení budeme uvažovat pouze *lineární SVM*, trénovanou na *lineárně separabilních datech* klasifikovaných do dvou tříd: $\{\mathbf{x}_i, y_i\}$, kde $y_i \in \{-1, +1\}$ a $i = 1, 2, \dots, n$. Problém však lze zobecnit i pro nelineární SVM a neseparabilní data, viz zápis z 8. přednášky.

2.1 Princip

Cílem tohoto klasifikátoru je nalezení rozdělujícího pásu maximální šířky d tak, aby se při klasifikaci do tříd minimalizovalo riziko klasifikace. Výsledkem je rozdělující nadplocha, viz obrázek 2, která tvoří lineární diskriminační funkci (1), viz kapitola 1, a která leží v ose pásu. Nadplocha je tedy dána rovnicí

$$g(\mathbf{x}) = w_0 + \mathbf{w}^T \mathbf{x} = 0. \tag{3}$$

Rozhodnutí klasifikátoru SVM je dáno znaménkem diskriminační funkce. Klasifikace do jedné třídy pro $g(\mathbf{x}) \geq 0$ a do druhé třídy pro $g(\mathbf{x}) < 0$.



Obrázek 2: Rozdělující pás a diskriminační funkce klasifikátoru SVM

2.2 Učení klasifikátoru

Učení klasifikátoru SVM spočívá v tom, že se maximalizuje vzdálenost nejbližšího bodu od nadplochy, tedy

$$J(\mathbf{w}, w_0) = \max_{\mathbf{w}, w_0} \min_{\mathbf{x}_i} \left| \frac{\mathbf{w}^T \mathbf{x}_i + w_0}{\|\mathbf{w}\|} \right| . \tag{4}$$

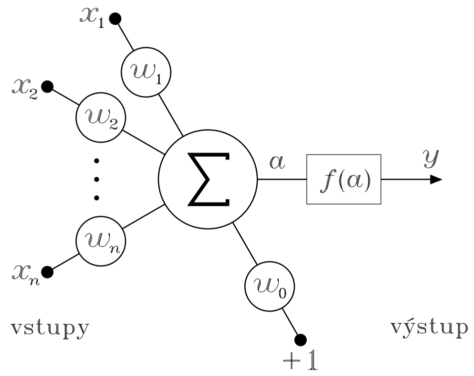
Optimalizaci lze řešit metodou kvadratického programování, kdy hledáme $\min_{\mathbf{w}, w_0} \|\mathbf{w}\|$ za podmínky $y_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1 \quad \forall i$.

3 Neuronové sítě

Pokusy s neuronovými sítěmi byly inspirovány činností mozku u živých organizmů. Dnes se využívají například pro klasifikaci, predikci, řízení, kompresi dat atd.

3.1 Neuron

Základním stavebním prvkem pro neuronové sítě je neuron. Obvykle má více než jeden vstup a pouze jeden výstup. Schéma neuronu můžeme vidět na obrázku 3.



Obrázek 3: Neuron

Funkce neuronu spočívá v tom, že se nejdříve určí vážený součet každého ze vstupů x_i. To můžeme zapsat jako $a = w_0 + \sum_{i=1}^{n} w_i x_i$, kde w_i jsou váhy jednotlivých vstupů a a tvoří diskriminační funkci. Výstup y neuronu pak závisí na vyhodnocení tohoto součtu aktivační funkcí $f(a)$

$$y = f\left(w_0 + \sum_{i=1}^{n} w_i x_i\right) . \tag{5}$$

Vstup s váhou w_0 tvoří ofset (*bias*) neuronu.

3.2 Aktivační funkce

Aktivační funkce obvykle bývá neklesající funkce. V neuronových sítích nejčastěji používáme několik základních typů, viz obrázek 4. Pro různé účely se však hodí různé funkce.

Uvedeme pouze definice těch nejzákladnějších:

- **Prahovací funkce**, která může být definována například jako

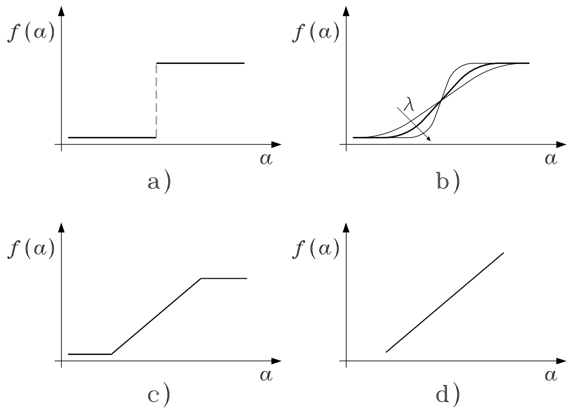
$$\mathrm{sign}(a) = \begin{cases} +1 & \text{pro } a \geq 0 \\ -1 & \text{pro } a < 0 \end{cases} \quad .$$

  Využívá se například pro úlohy klasifikace dat do tříd. V některých aplikacích však může činit problém její nediferencovatelnost v bodě a.

- **Sigmoida** je nejobvyklejší aktivační funkcí a je definována vztahem

$$f(a) = \frac{1}{1 + e^{-\lambda a}} \quad .$$

  Tato aktivační funkce je vhodná například pro adaptaci vah neuronů gradientními metodami.



Obrázek 4: Různé typy aktivačních funkcí: a) prahovací funkce sign, b) sigmoida, c) semilineární funkce, d) lineární funkce.

3.3 Topologie neuronových sítí

Neuronové sítě jsou sestaveny spojením výstupů neuronů z jedné vrstvy se vstupy neuronů jiné vrstvy. Podle způsobu jejich propojení rozlišujeme neuronové sítě (*Neural Networks - NN*) na:

- **Sítě s dopředným šířením** (*feedforward NN*), kde výstupy neuronů jedné vrstvy jsou zapojeny do vstupů v následující vrstvě a nevyskytují se zde žádné smyčky.
- **Sítě se zpětnými vazbami**, kde výstupy neuronů mohou být zapojeny na jakékoli vstupy ostatních neuronů včetně sebe sama. Zde rozlišujeme ještě dva případy:
  - *Vratné sítě* (*recurrent NN*), kde výstupy jedné vrstvy neuronů jsou zapojeny do vstupů v některé z předchozích vrstev.

– *Sítě s postranní zpětnou vazbou* (*lateral feedback NN*), kde výstupy neuronů jedné vrstvy jsou zapojeny na vstupy neuronů té samé vrstvy.

Vstupy neuronové sítě jsou většinou zavedeny na všechny neurony v první vrstvě. Říká se jí **vstupní vrstva** (*input layer*). Některé sítě mají ještě jednu vrstvu, kde do každého neuronu vstupuje právě jeden vstupní signál. Z této vrstvy jsou pak signály rozvedeny do neuronů ve vstupní vrstvě. Této vrstvě se říká **vrstva čidel** (*sensor layer*) a obvykle neprovádí žádný výpočet. Poslední vrstva sítě je **výstupní vrstva** (*output layer*). Vrstvy mezi vstupní a výstupní vrstvou se nazývají **skryté vrstvy** (*hidden layers*).

3.4 Učení neuronových sítí

Učení je procedura nalezení vah w_i. O některých metodách učení bude pojednáno v následujících kapitolách.

Ještě je nutno připomenout dva důležité termíny v oblasti NN:

- **Učení s učitelem** (*supervised*). K stanovení vah NN se použije *trénovací množina*. To je množina dvojic $\{(x_p, t_p)\}$, kde x_p jsou vstupní data a t_p je požadovaný výstup.
- **Samoorganizační** (*unsupervised*). K nastavení vah není potřeba trénovací množina. Síť se adaptuje sama během výpočtu.

3.5 Perceptron

Perceptron je speciální případ jednovrstvé neuronové sítě s prahovací aktivační funkcí. Protože výstup nabývá hodnot $y_i \in \{-1, +1\}$, používá se hlavně pro klasifikaci dat do dvou tříd. Realizuje diskriminační funkci (1), viz kapitola 1. Pro zopakování:

$$g(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x} = w_0 + \sum_{i=1}^{n} w_i x_i, \tag{6}$$

kde $\mathbf{x}$ je vektor vstupních proměnných, $\mathbf{w}$ je vektor vah a w_0 je posunutí $g(\mathbf{x})$ v prostoru vůči počátku souřadnic (*bias*).

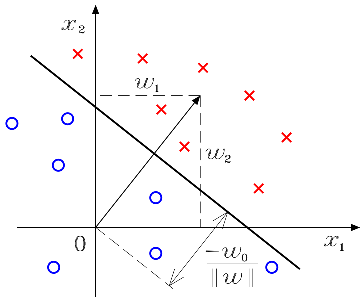
Klasifikace dat do tříd pak probíhá tak, že je-li $g(\mathbf{x}) \geq 0$, pak bude vzorek klasifikován do jedné třídy. V opačném případě pak do druhé třídy.

Příklad 2:
Ukázka použití perceptronu pro klasifikaci lineárně separabilních dat do dvou tříd.

Perceptron nyní realizuje diskriminační funkci (6), kde $n = 2$. Rozdělující nadplocha je zde dána rovnicí

$$w_0 + w_1 x_1 + w_2 x_2 = 0.$$

Geometrickou reprezentaci diskriminační funkce, vektoru vah a jednotlivých klasifikovaných vstupních vzorků můžeme vidět na obrázku 5.



Obrázek 5: Klasifikace do dvou tříd perceptronem

Učení

Učení perceptronu probíhá podle *perceptronového algoritmu*. Ten je společně s podrobnějším popisem perceptronu popsán v zápisu z 5. přednášky.

3.6 Mnohavrstvý perceptron

Mnohavrstvý perceptron (*Multi-Layer Perceptron*) je jednou z dalších aplikací neuronových sítí. Má strukturu sítě s dopředným šířením. Na obrázku 6 je uveden příklad dvouvrstvé sítě. Výstup takové sítě můžeme zapsat ve tvaru

$$y_k = \tilde{f}\left(\sum_{j=1}^{k} w_{kj}^{(2)} f\left(\sum_{i=1}^{M} w_{ji}^{(1)} x_i\right)\right), \tag{7}$$

kde

$w_{ji}^{(1)}$ jsou váhy v první vrstvě,

$w_{kj}^{(2)}$ jsou váhy ve druhé vrstvě,

f je aktivační funkce v první vrstvě (sigmoida),

$\tilde{f}$ je aktivační funkce ve druhé vrstvě (sigmoida),

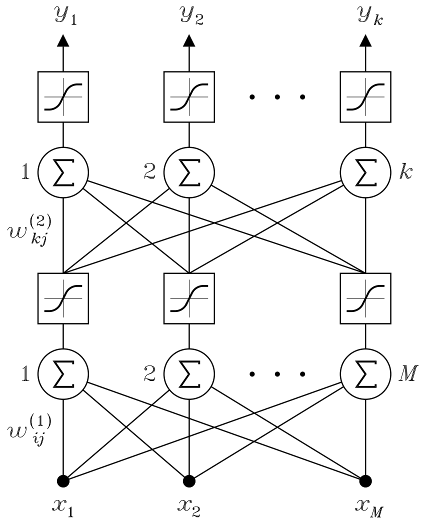
M je počet neuronů v první vrstvě,

k je počet neuronů ve druhé vrstvě.

Mnohavrstvý perceptron má *univerzální aproximační schopnost*. V prostoru dokáže vytvořit jakoukoli nadplochu, která od sebe odděluje jednotlivé datové třídy. Klasifikace do tříd probíhá na základě vyhledání maxima ve výstupním vektoru $\mathbf{y}$.

Učení

Mnohavrstvý perceptron se učí jednou z metod gradientní optimalizace, a sice algoritmem zpětného šíření (*backpropagation*). Ten je podrobněji popsán v následující kapitole.

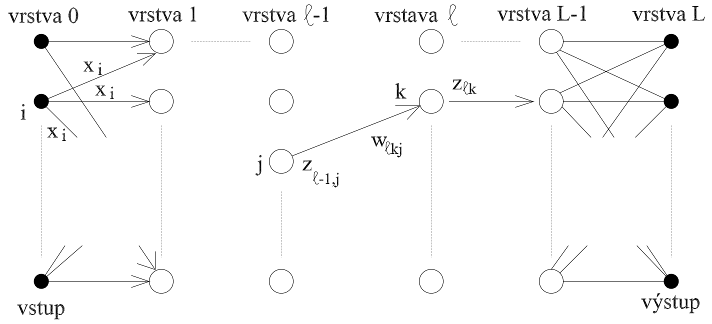


Obrázek 6: Schéma dvouvrstvého perceptronu

3.7 Neuronové sítě se zpětným šířením

3.7.1 Úvod

Tyto neuronové sítě (*The backpropagation NN - BPNN*) jsou základní aplikací NN. Mají strukturu sítě s dopředným šířením (*feedforward NN*), tzn., že neobsahují zpětné vazby, viz obrázek 7. Každý neuron má na svůj vstup přiveden výstup všech neuronů z předchozí vrstvy (vyjímkou je vrstva senzorů).



Obrázek 7: Obecná struktura sítě se zpětným šířením

Vzhledem k velkému počtu indexů na obrázku 7 budeme používat následující značení:

- $z_{\ell k}$ je výstup neuronu k z vrstvy ℓ,
- $w_{\ell kj}$ je váha, s jakou výstup neuronu j z vrstvy $\ell - 1$ působí na vstup neuronu k ve vrstvě ℓ,
- $\mathbf{x}_p$ je vstupní prvek číslo p z trénovací množiny,
- $\mathbf{t}_p(\mathbf{x}_p)$ je požadovaný výstup odpovídající vstupnímu prvku $\mathbf{x}_p$ z trénovací množiny (v době učení),

- N_ℓ je počet neuronů ve vrstvě ℓ,
- L je počet vrstev (vstupní vrstva má číslo 0 a výstupní L),
- P je počet prvků v trénovací množině, $p = \overline{1, P}$,

Trénovací množina je pak $\{(\mathbf{x}_p, \mathbf{t}_p)\}_{p=\overline{1,P}}$.

3.7.2 Odezva BPNN

Každý neuron počítá váženou sumu vstupů, kterou vyhodnotí aktivační funkcí. Výstup k-tého neuronu ve vrstvě ℓ tedy je (zatím neuvažujeme ofset)

$$z_{\ell k} = f\left(\sum_{j=1}^{N_{\ell-1}} w_{\ell kj} z_{\ell-1,j}\right). \tag{8}$$

Výstup podle (8) musí být vypočten pro každou vrstvu počínaje vstupní vrstvou až k vrstvě výstupní.

Mnohem přehlednější je však maticový zápis. Pro každou vrstvu lze sestavit matici vah

$$\mathbf{W}_\ell = \begin{pmatrix} w_{\ell 11} & \cdots & w_{\ell 1 N_{\ell-1}} \\ \vdots & \ddots & \vdots \\ w_{\ell N_\ell 1} & \cdots & w_{\ell N_\ell N_{\ell-1}} \end{pmatrix}. \tag{9}$$

Je vidět, že váhy každého neuronu jsou v této matici uspořádány do řádků. Dále zavedeme vektor výstupů předchozí vrstvy:

$$\mathbf{z}_{\ell-1}^T = \begin{pmatrix} z_{\ell-1,1} & \cdots & z_{\ell-1,N_{\ell-1}} \end{pmatrix}. \tag{10}$$

Výstup aktualní hladiny ℓ můžeme potom zapsat jako

$$\mathbf{z}_\ell^T = f(\mathbf{a}_\ell^T) = \begin{pmatrix} f(a_{\ell 1}) & \cdots & f(a_{\ell N_\ell}) \end{pmatrix}, \tag{11}$$

kde $\mathbf{a}_\ell = \mathbf{W}_\ell \mathbf{z}_{\ell-1}$.

3.7.3 Učení metodou zpětného šíření

Učení BPNN probíhá s učitelem (*supervised*). Váhy jednotlivých neuronů hledáme tak, aby jsme minimalizovali chybovou funkci.

Definice 3.1 *Pro libovolný vstup* $\mathbf{x}$ *a jemu odpovídající požadovaný výstup* $\mathbf{t}$ *definujeme* ***sumu kvadratické chybové funkce*** *(*sum-of-squares-error function*)* $E(W)$ *(E je závislá na všech vahách W):*

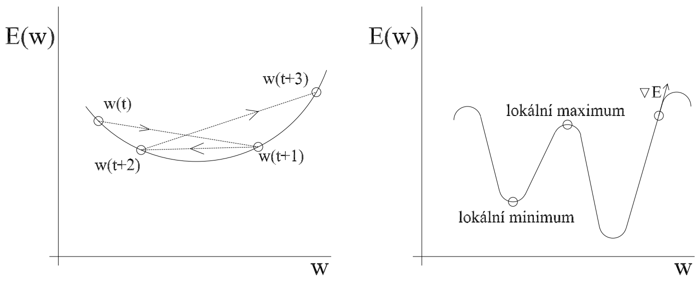
$$E(W) = \frac{1}{2} \sum_{q=1}^{N_L} \left(z_{Lq}(\mathbf{x}) - t_q(\mathbf{x})\right)^2, \tag{12}$$

kde z_{Lq} je výstup neuronu q ve výstupní vrstvě.

Abychom dosáhli co nejlepší spolehlivosti BPNN, potřebujeme tuto funkci minimalizovat. Gradientní vektor $\nabla E = \{\frac{\partial E(W)}{\partial w_{\ell ji}}\}$ nám udává směr největšího růstu chybové funkce. My však potřebujeme tuto odchylku minimalizovat, proto se vydáme opačným směrem (znaménko "-"). Dále zavedeme diskrétní časové okamžiky $t+1$ pro nově upravené váhy a t pro staré hodnoty. Potom můžeme učící proces napsat:

$$w_{\ell ji}(t+1) = w_{\ell ji}(t) - \mu \frac{\partial E(W)}{\partial w_{\ell ji}}\Big|_{W(t)} \quad , \tag{13}$$

kde P je počet vektorů trénovací množiny a μ učící konstanta taková, že $\mu > 0$. Tato konstanta určuje rychlost a kvalitu učení celého procesu. Jak je patrné z obrázku 8, může při učení dojít k oscilacím. Tyto oscilace ovlivňuje právě tato konstanta (μ je příliš veliké).



Obrázek 8: Vlevo oscilace v učebním procesu a vpravo problém lokálního minima.

Dalším velmi významným problémem této metody je, že gradientní vektor ∇E může "uváznout" v nějakém lokálním minimu a dále se již nepodaří tuto funkci minimalizovat (tento problém lze velmi omezeně také vyřešit zvětšením μ). Tento jev je znázorněn na obrázku 8.

Vraťme se však zpět ke vztahu (13). Tento vztah můžeme rovněž zapsat maticově:

$$-\mathbf{W}(t+1) = \mathbf{W}(t) - \mu \nabla \mathbf{E} \tag{14}$$

Je vidět, že vztah (14) je rekurzivní, a proto potřebujeme počáteční $\mathbf{W}(0)$. Počáteční váhy obvykle inicializujeme malými náhodnými hodnotami, obvykle z intervalu $w_{\ell ij} \in < -1, 1 >$. Vhodnou volbou $\mathbf{W}(0)$ můžeme také zabránit uváznutí.

Největším problémem metody zpětného šíření je výpočet gradientu ∇E. Tato operace má velkou časovou náročnost $\mathcal{O}(N_W^2)$, kde N_W je počet váhových parametrů neuronu. To je způsobeno tím, že každý výpočet E má složitost $\mathcal{O}(N_W)$. Tento výpočet však musíme opakovat pro každou $w_{\ell ji}$ daného neuronu. Výhoda algoritmu zpětného šíření je, že snižuje časovou náročnost výpočtu ∇E na $\mathcal{O}(N_W)$. Metodu zpětného šíření nebudeme dále rozebírat. Více podrobností naleznete v [3].

3.7.4 Ofset (Bias)

Pro některé aplikace nestačí popis neuronu, jak bylo uvedeno ve vztahu (8). Proto se do neuronu zavádí další vstup vážený vahou $w_{\ell k0}$, tzv. ofset (*bias*).

Pro výstup neuronu potom platí

$$z_{\ell k} = f\left(w_{\ell k0} + \sum_{j=1}^{N_{\ell-1}} w_{\ell kj} z_{\ell-1,j}\right). \tag{15}$$

Abychom nemuseli příliš měnit již jednou uvedené vztahy, budeme se na offset dívat jako na další vážený vstup, který je trvale připojen na úroveň 1. To provedeme tak, že přidáme jeden neuron $z_{\ell-1,0}$ do předchozí vrstvy $(\ell-1)$. Ten se bude chovat jako senzor, který rozvádí hodnotu 1 na všechny neurony ve vrstvě ℓ. Pak už jen zavedeme modifikace výstupních vektorů neuronů v prostoru N+1

$$\widetilde{\mathbf{z}}_\ell^T = \begin{pmatrix} 1 & z_{\ell 1} & \cdots & z_{\ell N_\ell} \end{pmatrix} \tag{16}$$

a váhových matic

$$\widetilde{\mathbf{W}}_\ell = \begin{pmatrix} w_{\ell 10} & w_{\ell 11} & \cdots & w_{\ell 1 N_{\ell-1}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{\ell N_\ell 0} & w_{\ell N_\ell 1} & \cdots & w_{\ell N_\ell N_{\ell-1}} \end{pmatrix}. \tag{17}$$

3.8 Pseudoinverze

Pseudoinverze se využívá v případech, kdy není možné vypočítat inverzní matici z důvodů její singularity.

3.8.1 Zobecněná diskriminační funkce

Aby bylo možno pochopit princip pseudoinverze, potřebujeme zavést notaci zobecněné diskriminační funkce, která byla uvedena výše. Připomeňme jen, že $\varphi : X \to X$ je vektorová funkce, provádějící zobrazení mezi prostory stejné dimenze. Pak pomocí vztahu (2) můžeme pro prostor N+1 psát:

$$y_k(\widetilde{\mathbf{x}}) = \widetilde{\mathbf{w}}_k^T \widetilde{\varphi}(\mathbf{x}), \tag{18}$$

kde $\widetilde{\varphi}_0(\mathbf{x}) \equiv 1$. A maticově můžeme psát:

$$\mathbf{y}(\mathbf{x}) = W\widetilde{\varphi}(\mathbf{x}). \tag{19}$$

3.8.2 Odvození

Odvození pseudoinverze vychází z minimalizování sumy kvadratické chybové funkce (*sum-of-squares-error function*) se zobecněnou diskriminační funkcí. Pro $E(W)$ z definice 3.1 pro jednovrstvé neuronové sítě platí:

$$E(W) = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{K}\left(\widetilde{\mathbf{w}}_k^T \widetilde{\varphi}(\mathbf{x_p}) - \mathbf{t}_{kp}\right)^2, \tag{20}$$

kde

$$\mathbf{y}_k^T = \begin{pmatrix} \widetilde{\mathbf{w}}_k^T \widetilde{\varphi}(\mathbf{x}_1) & \cdots & \widetilde{\mathbf{w}}_k^T \widetilde{\varphi}(\mathbf{x}_p) \end{pmatrix}, \tag{21}$$

$$\varphi_i^T = \begin{pmatrix} \widetilde{\varphi}_i(\mathbf{x}_1) & \cdots & \widetilde{\varphi}_i(\mathbf{x}_P) \end{pmatrix}. \tag{22}$$

Chybovou funkci E můžeme tedy psát:

$$\begin{aligned} E &= \frac{1}{2}\sum_{k=1}^{K}\sum_{p=1}^{P}\left(\{\mathbf{y}_k\}_p - t_{kp}\right)^2 = \frac{1}{2}\sum_{k=1}^{K}\|\mathbf{y}_k - \mathbf{t}_k\|^2 = \\ &= \frac{1}{2}\sum_{k=1}^{K}\left(\mathbf{y}_k - \mathbf{t}_k\right)^T\left(\mathbf{y}_k - \mathbf{t}_k\right) \end{aligned} \tag{23}$$

a z toho

$$\frac{\partial E}{\partial w_{ki}} = 0 \Leftrightarrow \varphi_i^T(\mathbf{y}_k - \mathbf{t}_k) = 0. \tag{24}$$

Nyní můžeme přistoupit k vlastnímu odvození pseudoinverze. Zavedeme matici $\mathbf{\Phi}$ rozměrů $P \times (N+1)$

$$\mathbf{\Phi} = \begin{pmatrix} \widetilde{\varphi}_0(\mathbf{x}_1) & \cdots & \widetilde{\varphi}_N(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \widetilde{\varphi}_0(\mathbf{x}_P) & \cdots & \widetilde{\varphi}_N(\mathbf{x}_P) \end{pmatrix} \tag{25}$$

a matici $\mathbf{T}$ rozměrů $P \times K$, kde sloupce tvoří vektory $\mathbf{t}_k$

$$\mathbf{T} = \begin{pmatrix} t_{11} & \cdots & t_{K1} \\ \vdots & \ddots & \vdots \\ t_{1P} & \cdots & t_{KP} \end{pmatrix}. \tag{26}$$

Využitím vztahu (21) můžeme podmínku (24) zapsat pomocí matic následovně:

$$\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{W}^T - \mathbf{T}) = \mathbf{0}, \tag{27}$$

$$\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{W}^T - \mathbf{\Phi}^T\mathbf{T} = \mathbf{0}, \tag{28}$$

$$\mathbf{W}^T = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{T} = \mathbf{\Phi}^\dagger\mathbf{T}. \tag{29}$$

Matice $\mathbf{\Phi}^\dagger$ se nazývá pseudoinverze matice $\mathbf{\Phi}$ (není obecně čtvercová). Matice $\mathbf{\Phi}^\dagger$ a je definována:

$$\boxed{\mathbf{\Phi}^\dagger = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T} \quad . \tag{30}$$

Pokud je $\mathbf{\Phi}^T\mathbf{\Phi}$ singulární, definuje se matice pseudoinverze jako:

$$\mathbf{\Phi}^\dagger = \lim_{\varepsilon\to 0}(\mathbf{\Phi}^T\mathbf{\Phi} + \varepsilon\mathbf{I})^{-1}\mathbf{\Phi}^T. \tag{31}$$

4 Radial Basis Function

RBF *Radial Basis Function* jsou další z mnoha aplikací neuronových sítí. Používá se např. pro interpolaci funkcí a klasifikaci.

4.1 Interpolace

Pro NN síť s jedním výstupem mějme množinu trénovacích dat $\{\mathbf{x}_{p=\overline{1,P}}\}$, které odpovídá množina požadovaných výstupů $\{\mathbf{t}_{p=\overline{1,P}}\}$. Dále mějme funkci $h : X \to Y$, která se snaží co nejpřesněji zobrazit trénovací množinu na množinu výstupu:

$$h(\mathbf{x}_p) = \mathbf{t}_p, \quad p = \overline{1,P}. \tag{32}$$

Předpokládejme, že funkci $h(x)$ můžeme nalézt jako lineární kombinaci množiny P bázových funkcí tvaru $\varphi(\|\mathbf{x} - \mathbf{x}_p\|)$:

$$h(\mathbf{x}) = \sum_{p=1}^{P} w_p \varphi(\|\mathbf{x} - \mathbf{x}_p\|). \tag{33}$$

Zavedením symetrické matice

$$\mathbf{\Phi} = \begin{pmatrix} \varphi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \cdots & \varphi(\|\mathbf{x}_P - \mathbf{x}_1\|) \\ \vdots & \ddots & \vdots \\ \varphi(\|\mathbf{x}_1 - \mathbf{x}_P\|) & \cdots & \varphi(\|\mathbf{x}_P - \mathbf{x}_P\|) \end{pmatrix} \tag{34}$$

můžeme vztah (32) přepsat do tvaru

$$\mathbf{w}^T \mathbf{\Phi} = \mathbf{t}^T, \tag{35}$$

kde $\mathbf{w}^T = (w_1, \ldots, w_p)$ a $\mathbf{t}^T = (t_1, \ldots, t_p)$.

Pokud je matice $\mathbf{\Phi}$ regulární, můžeme váhový vektor $\mathbf{w}$ nalézt inverzí této matice jako

$$\mathbf{w}^T = \mathbf{t}^T \mathbf{\Phi}^{-1}. \tag{36}$$

Pro sítě s více výstupy můžeme (32) a (33) rozšířit na tvar

$$\begin{aligned} h_k(\mathbf{x}_p) &= \mathbf{t}_{kp}, \quad p = \overline{1,P}, k = \overline{1,K}, & (37) \\ h_k(\mathbf{x}) &= \sum_{p=1}^{P} w_{kp} \varphi(\|\mathbf{x} - \mathbf{x}_p\|), \quad k = \overline{1,K}. & (38) \end{aligned}$$

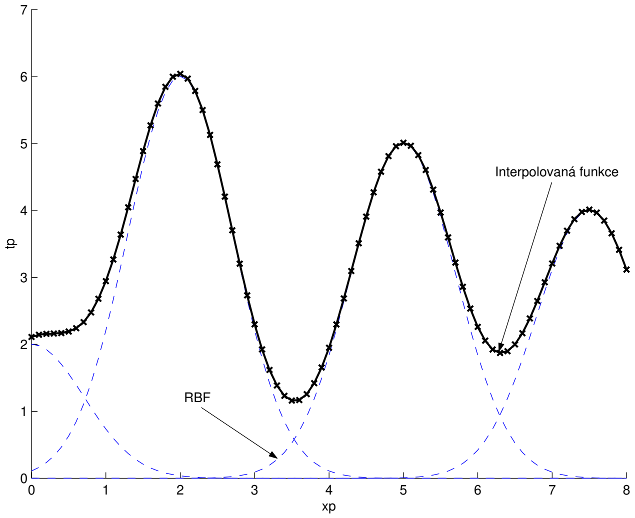
Když sestavíme vektor $\mathbf{h} = (h_1, \ldots, h_K)$, pak podobně jako v (32) dostaneme $\mathbf{h}(\mathbf{x}_p) = \mathbf{t}_p$. Sestavením matice $\mathbf{W}$, kde $\mathbf{w}$ tvoří její řádky a matice $\mathbf{T}$, kde $\mathbf{t}$ tvoří řádky, dostaneme $\mathbf{W\Phi} = \mathbf{T}$. Matici vah pak můžeme určit jako $\mathbf{W} = \mathbf{T\Phi}^{-1}$ v případě, že $\mathbf{\Phi}$ je regulární.

V případě, kdy $\mathbf{\Phi}$ není regulární matice, vypočteme $\mathbf{\Phi}^{-1}$ pomocí pseudoinverze, viz kapitola 3.8. K tomu může dojít nařiklad v případě, že zvolíme méně bázových funkcí než je skutečných trénovacích dat.

Jako bázové funkce se nejčastěji používají:

- Gaussián: $\varphi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right)$,
- lineární funkce: $\varphi(x) = x$,
- kubická funkce: $\varphi(x) = x^3$,
- multi-kvadratická funkce: $\varphi(x) = \sqrt{x^2 + \sigma^2}$.

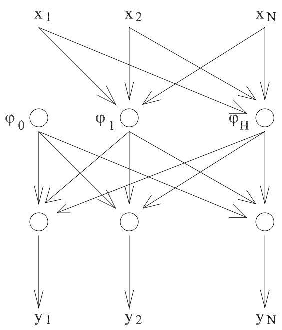
  Příklad 3:
  Interpolace pomocí RBF. Jako bázová funkce skryté vrstvy je zvolen Gaussián.



Obrázek 9: Interpolace pomocí RBF

4.2 Neuronové sítě RBF

RBF síť je tvořena neuronovou sítí o dvou vrstvách. Vstupní vrstva přivádí signál s váhou 1 na skrytou vrstvu. Neurony v této skryté vrstvě mají jako aktivační funkci nějakou bázovou funkci. Aktivační funkce výstupní vrstvy je lineární funkce (identita). Neuronová síť RBF je znázorněna na obrázku 10.



Obrázek 10: Možná struktura RBF

4.3 Učení sítí RBF

Velikosti vah a pozice středů bázových funkcí získáváme učením. To probíhá ve dvou fázích:

1. Učení první vrstvy :

- metodami shlukové analýzy,
- editací k-NN (např. určení středů bázových funkcí),
- EM algoritmus.

2. Učení druhé vrstvy :

- metodou nejmenších čtverců,
- gradientními algoritmy,
- perceptronovým algoritmem,
- minimalizace kvadratické funkce.

Reference

[1] R. O. Duda, P. E. Hart, G. E. Stork: *Pattern Classification (Chapt. 5)*,
http://cmp.felk.cvut.cz/cmp/courses/recognition/resources/DHS/DHSChap5.pdf

[2] J. C. Bourges: *A Toutorial on Support Vector Machines for Pattern Recognition*,
http://cmp.felk.cvut.cz/cmp/courses/recognition/resources/svmBourges.ps.gz

[3] R. M. Hristev: *The ANN Book*,
http://cmp.felk.cvut.cz/cmp/courses/recognition/resources/nnBook.ps.gz

[4] B. Kröse, P. van der Smagt: *An Introduction to Nural Networks*,
http://cmp.felk.cvut.cz/cmp/courses/recognition/resources/ann_Book.ps.gz