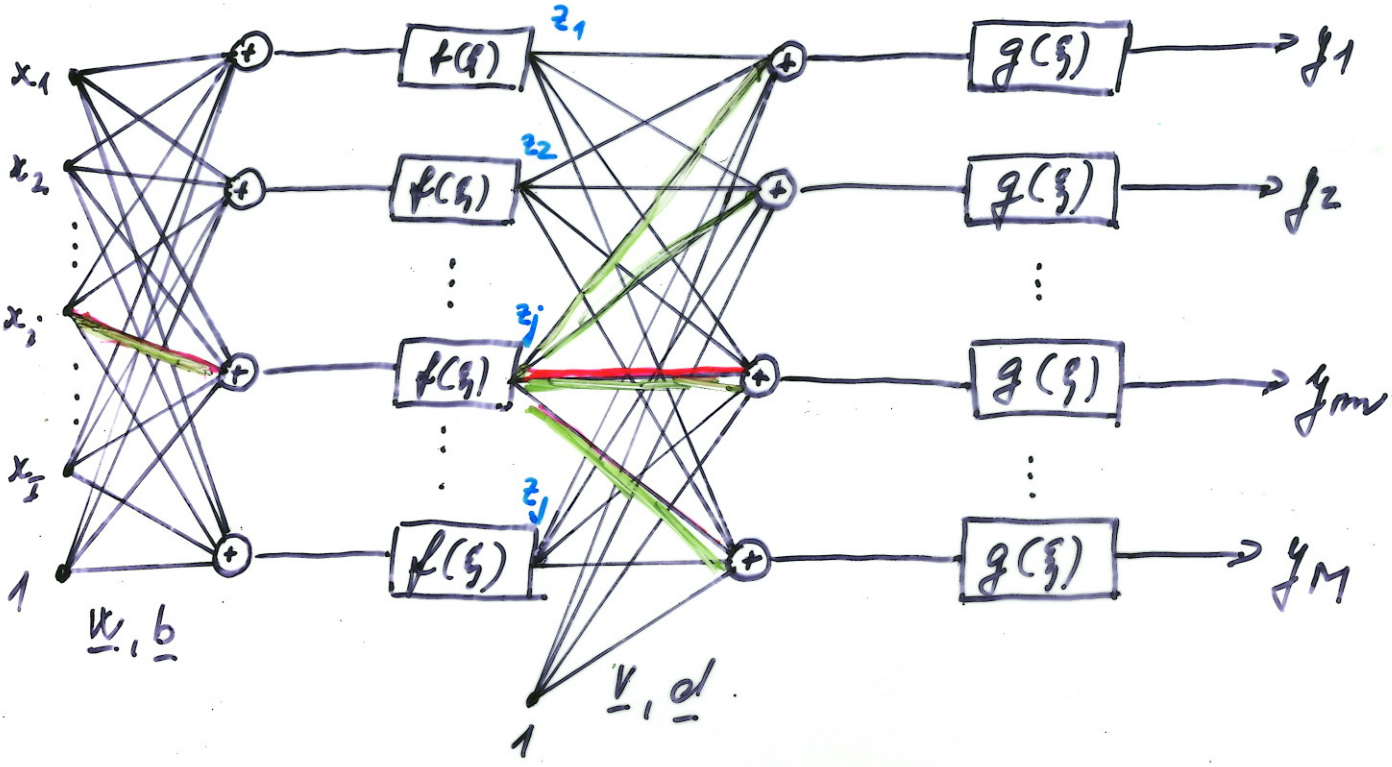
# 3. NELINEÁRNÍ SÍTĚ

- jsou tvořeny jednou nebo více vrstvami neuronů se spojitými aktivačními funkcemi, přičemž alespoň v jedné vrstvě jsou aktivační funkce nelineární.
- schéma dvouvrstvé nelineární sítě s $I$ vstupy, $M$ výstupy a $J$ neurony ve skryté vrstvě




$\underline{x} = [x_1, x_2, \dots, x_i, \dots, x_I]^T$ ... vstupní vektor sítě

$\underline{z} = [z_1, z_2, \dots, z_j, \dots, z_J]^T$ ... výstupní vektor 1. vrstvy, vstupní vektor 2. vrstvy

$\underline{y} = [y_1, y_2 \dots y_{mv}, \dots, y_M]^T$ ... výstupní vektor sítě

$$\underline{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1I} \\ w_{21} & w_{22} & \dots & w_{2I} \\ \vdots & \vdots & & \vdots \\ w_{J1} & w_{J2} & \dots & w_{JI} \end{bmatrix}$$ ... váhová matice 1. vrstvy (typu $J \times I$)

$\underline{b} = [b_1, b_2, \dots, b_j, \dots, b_J]^T$ ... prahový vektor 1. vrstvy (dimenze $J$)

$$\underline{V} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1J} \\ v_{21} & v_{22} & \cdots & v_{2J} \\ \vdots & \vdots & & \vdots \\ v_{M1} & v_{M2} & \cdots & v_{MJ} \end{bmatrix} \dots \text{ váhová matice 2. vrstvy (typu } M \times J)$$

$\underline{d} = [d_1, d_2, \dots, d_m, \dots d_M]^T$ ... prahový vektor 2. vrstvy (dimenze M)

$f(\xi)$ ... aktivační funkce 1. vrstvy

$g(\xi)$ ... aktivační funkce 2. vrstvy

- pro výstup $j$-tého neuronu 1. vrstvy platí

$$z_j = f\left(\sum_{i=1}^{I} w_{ji} \cdot x_i + b_j\right)$$

- pro výstup $m$-tého neuronu 2. vrstvy platí

$$y_m = g\left(\sum_{j=1}^{J} v_{mj} \cdot z_j + d_m\right) =$$

$$= g\left(\sum_{j=1}^{J} v_{mj} \cdot f\left(\sum_{i=1}^{I} w_{ji} \cdot x_i + b_j\right) + d_m\right)$$

Pokud alespoň 1 z funkcí $f(\xi)$ a $g(\xi)$ je nelineární

$\Rightarrow$ závislost $y$ na $\underline{x}$ je nelineární

# Trénování sítě pomocí učení s učitelem

! předpokládá se, že máme k dispozici trénovací množinu, tj. množinu $P$ dvojic [vstup $x_p$, požadovaný výstup $u_p$]

! chceme nastavit váhy a prahy sítě tak, aby výstup sítě byl pro všechny vstupní vektory z trénovací množiny správný, tzn. aby celková chyba definovaná vztahem

$$E = \sum_{p=1}^{P} \varepsilon_p = \frac{1}{2} \sum_{p=1}^{P} \| \boldsymbol{u}_p - \boldsymbol{y}_p \|^2$$

byla minimální. Přitom

$\boldsymbol{y}_p$ je skutečný výstup sítě pro vstup $\boldsymbol{x}_p$,

$\boldsymbol{u}_p$ je požadovaný výstup sítě pro vstup $\boldsymbol{x}_p$,

$P$ je počet trénovacích dvojic,

$g_p$ je chyba od $p$-té trénovací dvojice.

! analogicky jako u lineární sítě se váhy a prahy mění úměrně gradientu chybové funkce, tzn. chyba se z výstupu sítě šíří zpět do jednotlivých vrstev, kde se podle ní mění váhy a prahy þ **algoritmus backpropagation** (algoritmus zpětného šíření chyby)

# Algoritmus trénování nelineární sítě

*Vstupní podmínky*:

Je dáno

- trénovací množina, tj. $P$ dvojic $[\boldsymbol{x}_p,\boldsymbol{u}_p]$, $p$=1,2,...,$P$.
- $\boldsymbol{x}_p$ ... vstupní vektor dimenze $I$
- $\boldsymbol{u}_p$ ... požadovaný výstupní vektor dimenze $M$ pro vstupní vektor $\boldsymbol{x}_p$
- počet neuronů ve skryté vrstvě $J$
- $f(>)$ ... aktivační funkce 1. vrstvy
- $g(>)$ ... aktivační funkce 2. vrstvy

*1. krok - Inicializace*

- Váhové matice $\boldsymbol{W}(1)$ typu $J\times I$ a $\boldsymbol{V}(1)$ typu $M\times J$ a prahové vektory $\boldsymbol{b}(1)$ dimenze $J$ a $d(1)$ dimenze $M$ inicializujeme malými náhodnými čísly.
- Zvolíme $c > 0$, $E_{max}$ $ 0 a $TC_{max} > 0$.
- Stanovíme $k$ =1, $p$ = 1, $q$ = 0 a $E(0) = 0$.

  $E_{max}$ je maximální povolená chyba, na kterou nebo pod kterou se chceme dostat

  $TC_{max}$ je maximální počet trénovacích cyklů, ve kterých chceme síť natrénovat

  $q$ je počet trénovacích cyklů

  $E$ je chyba způsobená nesprávnou činností sítě během jednoho trénovacího cyklu

  $c$ je konstanta učení

## *2. krok - Výpočet výstupu*

$\boldsymbol{x}(k) = \boldsymbol{x}_p$

$\boldsymbol{z}(k) = [f(\boldsymbol{w}_1(k)\textcircled{@}\boldsymbol{x}(k)+b_1(k)), f(\boldsymbol{w}_2(k)\textcircled{@}\boldsymbol{x}(k)+b_2(k)), \ldots, f(\boldsymbol{w}_J(k)\textcircled{@}\boldsymbol{x}(k)+b_J(k))]^T$

$\boldsymbol{y}(k) = [g(\boldsymbol{v}_1(k)\textcircled{@}\boldsymbol{z}(k)+d_1(k)), g(\boldsymbol{v}_2(k)\textcircled{@}\boldsymbol{z}(k)+d_2(k)), \ldots, g(\boldsymbol{v}_M(k)\textcircled{@}\boldsymbol{z}(k)+d_M(k))]^T$

$\boldsymbol{u}(k) = \boldsymbol{u}_p$

$\boldsymbol{w}_j$ ... $j$-tá řádka matice $\boldsymbol{W}$

$\boldsymbol{v}_j$ ... $m$-tá řádka matice $\boldsymbol{V}$

## *3. krok - Výpočet chyby*

$E(k)=E(k\text{-}1)+½(\boldsymbol{u}(k)\text{-}\boldsymbol{y}(k))^T\ (\boldsymbol{u}(k)\text{-}\boldsymbol{y}(k))$

## *4. krok - Modifikace vah a prahů*

$$w_{ji}(k+1) = w_{ji}(k) + c\sum_{m=1}^{M}\left(u_m(k)-y_m(k)\right)\cdot g'\left(\boldsymbol{v}_m(k)\cdot\boldsymbol{z}(k)+d_m(k)\right)\cdot v_{mj}(k)\cdot f'\left(\boldsymbol{w}_j(k)\cdot\boldsymbol{x}(k)+b_j(k)\right)\cdot x_i(k)$$

$$b_j(k+1) = b_j(k) + c\sum_{m=1}^{M}\left(u_m(k)-y_m(k)\right)\cdot g'\left(\boldsymbol{v}_m(k)\cdot\boldsymbol{z}(k)+d_m(k)\right)\cdot v_{mj}(k)\cdot f'\left(\boldsymbol{w}_j(k)\cdot\boldsymbol{x}(k)+b_j(k)\right)$$

$$v_{mj}(k+1) = v_{mj}(k) + c\left(u_m(k)-y_m(k)\right)\cdot g'\left(\boldsymbol{v}_m(k)\cdot\boldsymbol{z}(k)+d_m(k)\right)\cdot z_j(k)$$

$$d_m(k+1) = d_{mj}(k) + c\left(u_m(k)-y_m(k)\right)\cdot g'\left(\boldsymbol{v}_m(k)\cdot\boldsymbol{z}(k)+d_m(k)\right)$$

$\forall i;\ i=1,\ldots,I$

$\forall j;\ j=1,\ldots,J$

$\forall m;\ m=1,\ldots,M$

## *5. krok - Konec trénovacího cyklu*

Je-li $p < P$ potom $p = p + 1$

$k = k + 1$

jdi na krok 2)

jinak $q = q + 1$

jdi na krok 6)

*6. krok - Konec trénování*

$E_c(q) = E(k)$

Je-li $E_c(q) \# E_{max}$

potom KONEC (síť se podařilo natrénovat)

jinak je-li $q > TC_{max}$ potom KONEC (síť se nepodařilo natrénovat)

jinak $E(0) = 0$

$k = k + 1$

$p = 1$

jdi na krok 2)

$E_c$ ... chyba na konci trénovacího cyklu

**Poznámky k trénování nelineární sítě:**

! Pro konkrétní tvar aktivačních funkcí lze odvodit konkrétní vztahy pro modifikaci vah a prahů

< pro $h(\xi) = \frac{2}{1+e^{-\lambda\xi}} - 1$ je $h'(\xi) = \frac{\lambda}{2}(1-h^2)$

< pro $h(\xi) = \frac{1}{1+e^{-\lambda\xi}}$ je $h'(\xi) = \lambda h(1-h)$

< pro $h(\xi) = \lambda\xi$ je $h'(\xi) = \lambda$

! Speciálním případem nelineárních sítí jsou lineární sítě a sítě s binární bipolární aktivační funkcí

! Podobně jako u sítě s binární bipolární aktivační funkcí a u lineární sítě lze pro výpočet celkové chyby $E_c$ využít též vztah

$$E = \frac{1}{2PM}\sum_{p=1}^{P}\|\boldsymbol{u}_p - \boldsymbol{y}_p\|^2 = \frac{1}{2PM}\sum_{p=1}^{P}\sum_{m=1}^{M}(u_{pm} - y_{pm})^2 ,$$

kde $M$ je počet výstupů sítě, $P$ je počet trénovacích dvojic v trénovací množině a $u_{pm}$ resp. $y_{pm}$ je $m$-tá složka vektoru $\boldsymbol{u}_p$, resp. $\boldsymbol{y}_p$. Takto definovaná chyba umožňuje lépe posoudit činnost sítí s různým počtem výstupů a různě velkou trénovací množinou.

! *Volba konstanty učení*: Malá hodnota konstanty učení

zpomaluje proces učení, pro velkou hodnotu hrozí nebezpečí nestability procesu učení.

! *Vliv strmosti aktivační funkce*: Má na proces učení stejný vliv jako konstanta učení → konstantu učení je třeba volit s ohledem na strmost aktivační funkce.

! Algoritmus může skončit v nevyhovujících "mělkých" lokálních minimech chybové funkce (chybová funkce je závislost chyby na parametrech, tj. váhách a prahách sítě). Možné řešení:

< vyzkoušet různé inicializace vah a prahů
< zvýšit počet neuronů ve skrytých vrstvách, popř. počet skrytých vrstev (pouze částečné řešení)
< momentová metoda

! V průběhu trénovaní je třeba sledovat vývoj celkové chyby $E_c$, váhových matic a prahových vektorů v závislosti na počtu trénovacích cyklů a podle toho případně upravit hodnotu strmosti aktivačních funkcí 8 a konstanty učení *c*, nebo maximální počet trénovacích cyklů $TC_{max}$.

! Volba počtu vstupů, výstupů, skrytých vrstev a neuronů v nich, volba aktivačních funkcí v jednotlivých vrstvách a jejich strmosti 8 a volba konstanty učení *c* závisí na typu řešené úlohy.

## *Momentová metoda algoritmu backpropagation*

- jejím smyslem je zabránit tomu, aby se proces trénování zastavil v mělkém lokálním minimu
- Princip: pro změnu vah platí

$$w(t+1)=w(t)+\alpha\,\Delta w(t-1)-(1-\alpha)c\,\frac{\partial\varepsilon}{\partial w}, \qquad (*)$$

  kde $\Delta w(t-1)$ je minulá změna váhy, tj. $\Delta w(t-1)=w(t)-w(t-1)$, a $\alpha$ je tzv. momentová konstanta, pro kterou platí $\alpha \doteq 0{,}1$,
- analogický vztah platí i pro změnu prahů
- při inicializaci je třeba nastavit $\Delta w(0) = 0$ a $\Delta b(0) = 0$
- Momentová metoda se používá zejména při dávkovém trénování. Při sekvenčním trénování je třeba opatrnosti.

# Použití nelineárních sítí

## *Aproximátory funkcí*

! Jednotlivé vstupy nelineárních sítí představují nezávisle proměnné aproximované funkce, výstupy představují závisle proměnné ö **počet vstupů** je roven počtu nezávisle proměnných, **počet výstupů** je roven počtu závisle proměnných.

! Skryté vrstvy mají neurony s nějakou **sigmoidální aktivační funkcí**, ve výstupní vrstvě jsou obvykle neurony s **lineární aktivační funkcí**.

***Klasifikátory***

*Poznámka*: Úkolem klasifikace je zařazování předmětů do skupin (tzv. **tříd**) podle společných vlastností. Každý předmět je reprezentován svým **obrazem**, tj. vektorem charakteristických **příznaků**.

! Vstupem sítě jsou jednotlivé klasifikované obrazy (tj. vektory příznaků), tzn. síť má **tolik vstupů, kolik je příznaků**. Výstupem sítě je informace o zařazení vstupního obrazu do určité třídy, tzn počet **výstupů sítě** je **určen počtem tříd**, do kterých se klasifikuje. Je-li $R$ počet tříd, potom počet výstupů sítě je obvykle roven buď $R$ nebo $\log_2 R$. Obecně platí, že počet výstupů sítě může být jakékoli celé číslo z intervalu $\langle \log_2 R, R \rangle$ (většinou se volí horní mez).

! ve všech vrstvách se nejčastěji používá některá **sigmoidální aktivační funkce**, která se po natrénování ve výstupní vrstvě někdy nahradí odpovídající binární aktivační funkcí.

***Asociativní paměti***

*Poznámka*: Asociativní paměť je paměť adresovaná obsahem, adresou je klíčová hodnota ukládaná s informací.

! Vstupem sítě je vstupní obraz (tzv. klíč), kterým se přistupuje do paměti, výstupem sítě je příslušný asociovaný obraz ö **počet vstupů** sítě je **roven dimenzi vstupního obrazu**, **počet výstupů** je **roven dimenzi** asociobaného **výstupního obrazu**.

! Obvykle se využívají neurony se **sigmoidální aktivační funkcí**.

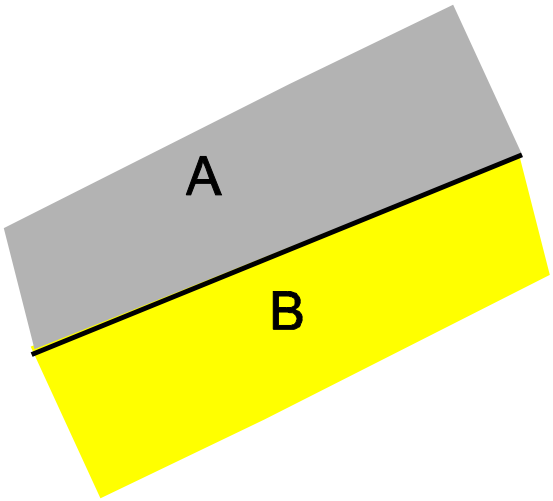
*Poznámka*: zapamatovaná informace je v síti uložena v hodnotách vah a prahů ö uložená informace je rozprostřena po celé síti ö asociativní paměť je schopna vykonávat svou funkci dostatečně správně i při částečném poškození (například při odstranění části neuronů skryté vrstvy). U pamětí adresovaných adresou se obsah z poškozené části ztrácí úplně.

Všechny tři výše uvedé aplikace lze matematicky formulovat jako aproximaci funkce (tj. optimalizační úlohu)

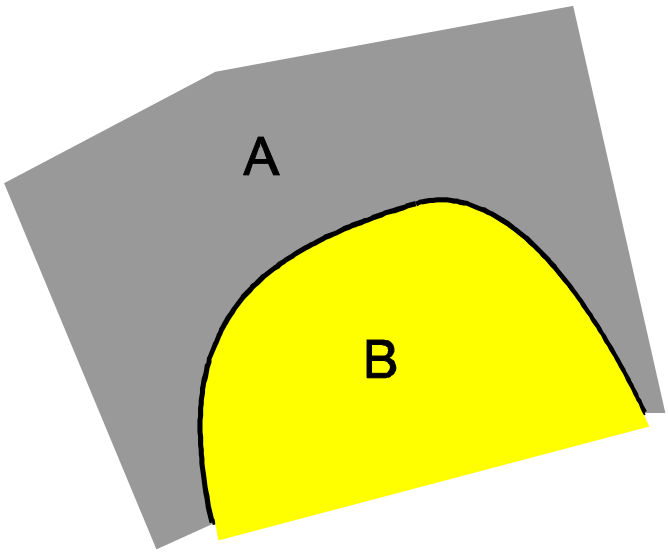
$$y = f(x, \textit{parametry}).$$

! pro aproximaci libovolné funkce postačují **2 až 3 vrstvy** neuronů:

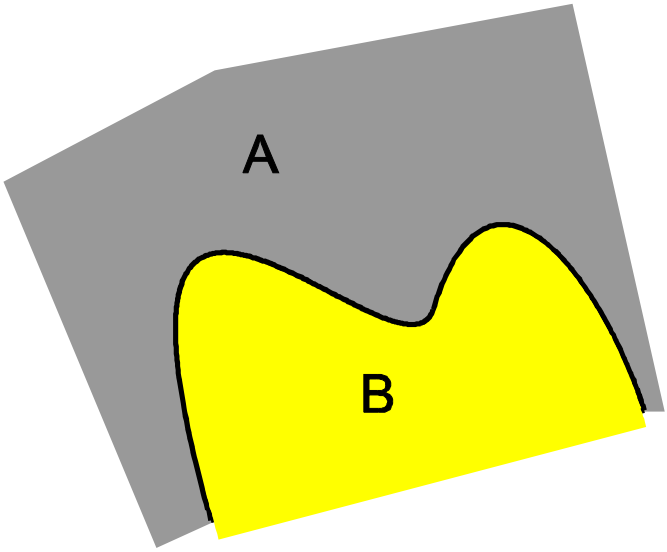
< jedna vrstva rozdělí vstupní prostor nadrovinou (přímkou)




< dvě vrstvy jsou schopny oddělit konvexní oblast (konvexní oblast je taková oblast, kde libovolné dva body této oblasti lze spojit úsečkou, která celá leží v této oblasti)



< tři vrstvy jsou schopny oddělit i nekonvexní oblasti




! počet skrytých vrstev a počet neuronů v nich má vliv na schopnost sítě zobecňovat (generalizovat) vstupní body z trénovací množiny. Velké množství vrstev a neuronů může vést k **přetrénování** (overfitting) sítě, malé množství k **nedotrénování** sítě (underfitting).