Umelé neurónové siete

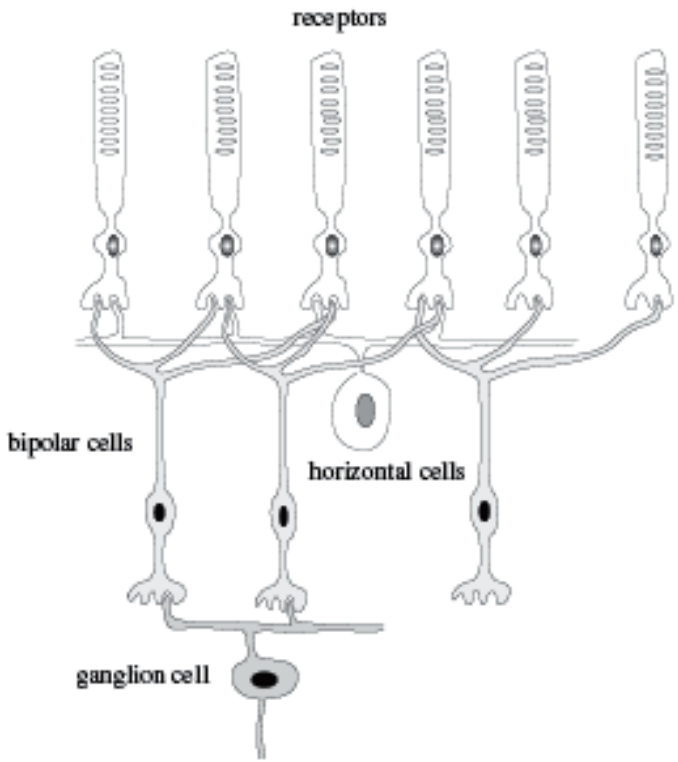
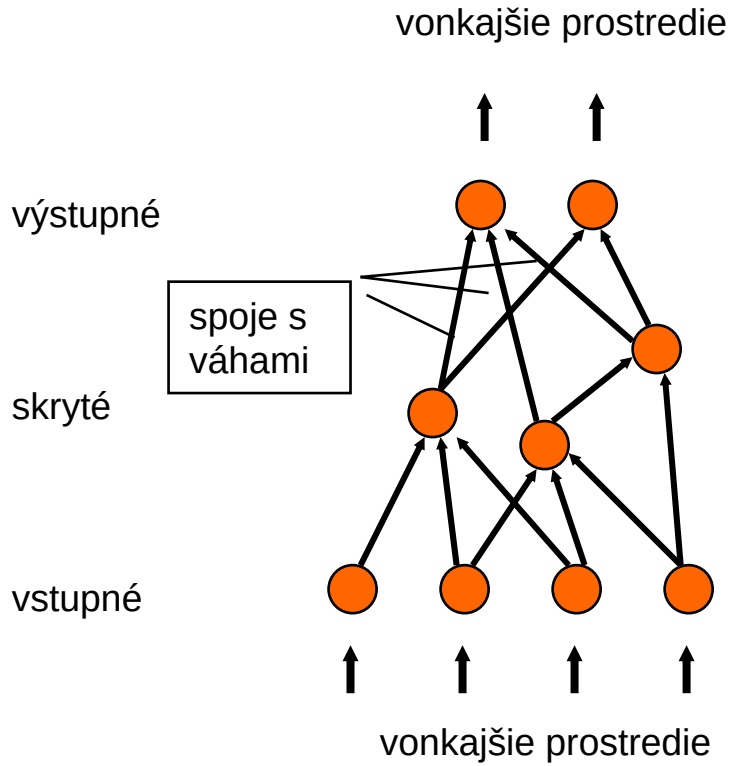


Fig. 3.12. The interconnection pattern in the retina

Umelé neurónové siete

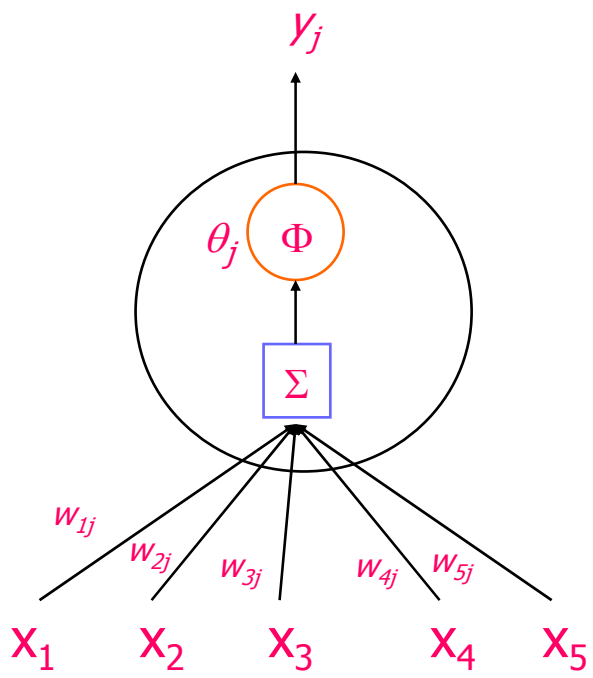
- Súbor jednotiek (**neurónov**) prepojených ohodnotenými spojmi (**synaptické váhy**), ktoré prenášajú signály
- neuróny
  - vstupné, výstupné a skryté
  - pracujú paralelne



Neurón

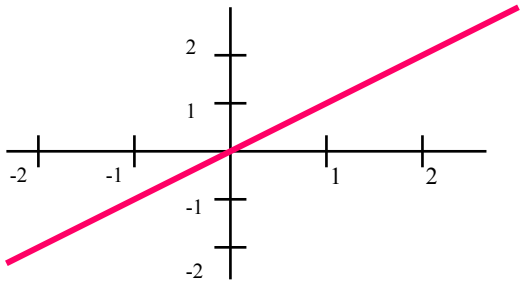
- Realizuje funkciu na váženom súčte jeho vstupov $x_1, x_2, \ldots, x_N$
- w_{ij} je váha spoja z i-teho vstupného neurónu do neurónu y_j
- θ_j je tzv. prah neurónu
- Φ je prenosová funkcia

$$y_j = \Phi\underbrace{\left(\sum_{i=1}^{N} w_{ij}\, x_i - \theta_j\right)}_{\xi_j \text{ potenciál}}$$



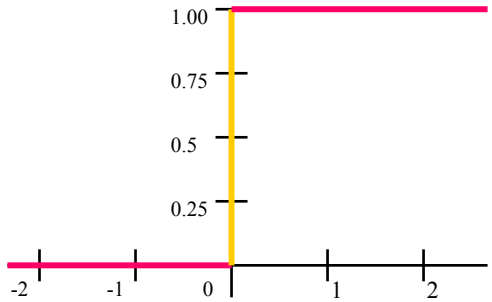
Prenosové funkcie

- λ je strmosť



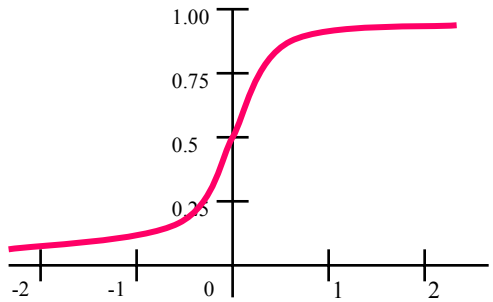
lineárna

$$\Phi(x) = kx$$



skoková

$$\Phi(x) = \begin{cases} 1 & \text{ak } x > 0 \\ 0 & \text{inak} \end{cases}$$



sigmoida

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

bipolárna

$$\Phi(x) = \begin{cases} 1 & \text{ak } x > 0 \\ -1 & \text{inak} \end{cases}$$

$\Phi(x) = \tanh(\lambda x)$ do intervalu $(-1,1)$

Architektúra NS

- Množina neurónov a ich prepojenie – graf
- často vrstvy: vstupná, výstupná, skryté
- 2 typy:
  - **s dopredným šírením (feedforward)**
    - acyklický graf, signály sa šíria od vstupnej vrstvy k výstupnej
  - **rekurentné**
    - spoje aj z vyšších vrstiev do nižších, príp. slučky
- θ môže byť váha spoja z dodatočného neurónu s výstupom -1; typicky sa však používa virtuálny neurón s výstupom 1

$$y_j = \Phi\underbrace{\left(\sum_{i=1}^{N} w_{ij} x_i - \theta_j\right)}_{\xi_j \text{ potenciál}} \quad \Rightarrow \quad y_j = \Phi\underbrace{\left(\sum_{i=1}^{N+1} w_{ij} x_i\right)}_{\xi_j \text{ potenciál}}$$

Učenie NS

- výstup siete závisí na nastavení váh; nastavenie váh sa hľadá **učením** – opakovane sa sieti predkladajú vzory a sieť si podľa nich upravuje váhy
  - **s učiteľom** – podľa chyby na výstupe; s oneskoreným učením (až na základe celkového ohodnotenia siete, bez znalosti presného výstupu)
  - **bez učiteľa** – samo-organizácia; klasifikácia, kompresia dát
- { **spriahnuté** – (on-line) váhy sa upravujú po predložení každého vzoru
  nespriahnuté – (off-line) váhy sa upravujú po predložení všetkých vzorov

zmena váhy $w_{ij}^{t+1} = w_{ij}^{t} + \Delta w_{ij}^{t+1}$ t čas

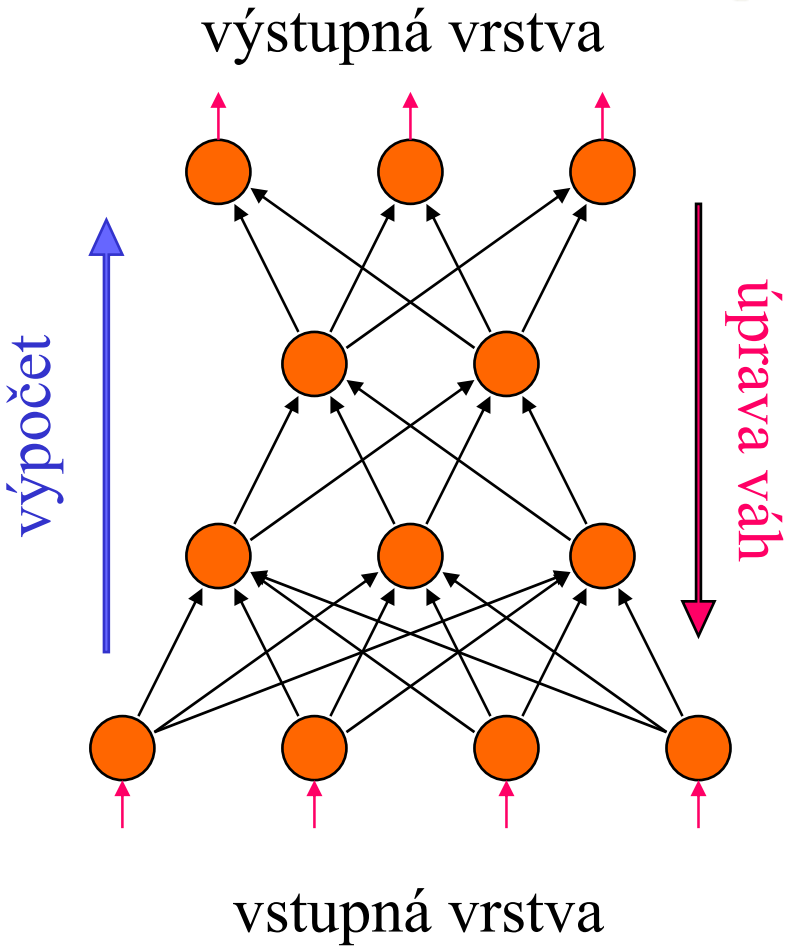
môžu nastať oscilácie, preto

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta\, \Delta w_{ij}^{t+1} \qquad 0 < \eta \leq 1 \qquad \textit{učiaca konštanta}$$

Učenie posilňovaním

- Reinforcement – posilňovanie; posilnenie odmenou za správnu reakciu; odmeňovanie
- vtedy, keď je spätná väzba
  - hrubá – napr. len 1 pre dobre a 0 pre zle
  - riedka – nie vždy, napr. až po dlhej postupnosti akcií
- cieľ – posilniť akcie, ktoré maximalizujú odmenu,
  1. Ktoré to sú?
  2. Ako medzi ne odmenu rozdeliť?
- agent (robot) musí vyskúšať mnoho dvojíc stav-akcia

Vrstevnaté neurónové siete – algoritmus spätného šírenia



- Trénovacia množina

$$\{(x_1, d_1), (x_2, d_2), \ldots, (x_n, d_n)\}$$

- Postup
  1. Predlož vstup x_i na vstupnú vrstvu.
  2. Spočítaj výstup y_i.
  3. porovnaj skutočný výstup y_i s požadovaným d_i.
  4. Rozdiel použi na úpravu váh.

***Trik**: Prahy sa nahradia vstupom z virtuálneho neurónu, ktorý dáva vždy výstup 1. Každá vrstva okrem výstupnej má práve jeden taký neurón.*

Algoritmus spätného šírenia (chyby)

- **Cieľ:** nájsť váhy také, aby pre všetky vstupné vzory z trénovacej množiny skutočný výstup siete bol rovnaký ako požadovaný výstup siete.
- Nie je špecifikovaná skutočná ani požadovaná aktivita na skrytých neurónoch
- Celková chyba siete – chybová funkcia siete – vyjadruje rozdiel medzi požadovaným a skutočným výstupom siete

$$E = \frac{1}{2}\sum_{p}\sum_{j}\left(y_{j,p} - d_{j,p}\right)^2$$

p indexuje vzory z trénovacej množiny

$y_{j,p}$ skutočná aktivita výstupného neurónu j pri vstupe x_p

$d_{j,p}$ požadovaná aktivita výstupného neurónu j pri vstupe x_p

$$d_j = \left(d_{j1}, \ldots, d_{jp}\right)$$

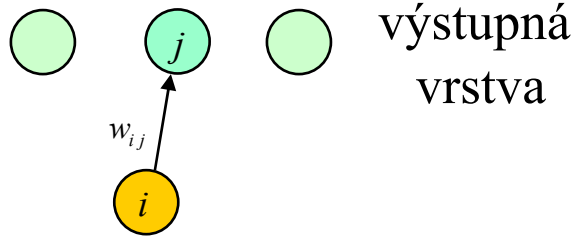
Algoritmus spätného šírenia

- Aktualizácia synaptických váh $w_{ij}(t+1) = w_{ij}(t) + \Delta_E w_{ij}(t)$
- $\Delta_E w_{ij}(t)$ ... prírastok váhy w_{ij} prispievajúci k minimalizácii E
- Pre výstupnú vrstvu platí

$$\Delta_E w_{ij} = -\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} = -\frac{\partial E}{\partial \xi_j}\frac{\partial}{\partial w_{ij}}\sum_k w_{kj} y_k =$$

$$= -\frac{\partial E}{\partial \xi_j} y_i = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j} y_i = -(y_j - d_j) f'(\xi_j) y_i = \delta_j y_i$$

$$\xi_j = \sum_{k=1}^{N} w_{kj} y_k \qquad y_j = f(\xi_j) = \frac{1}{1+e^{-\lambda \xi_j}}$$



$$E = \frac{1}{2}\sum_p \sum_j (y_{j,p} - d_{j,p})^2 \qquad f'(\xi_j) = y_j(1-y_j)\lambda$$

Algoritmus spätného šírenia

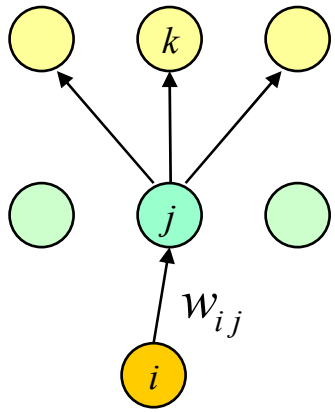
- Pre skryté vrstvy

$$\xi_j = \sum_{k=1}^{N} w_{kj} y_k$$

$$\Delta_E w_{ij} = \frac{\partial E}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} = -\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial \xi_k}{\partial y_j}\frac{\partial y_j}{\partial \xi_j} y_i =$$

$$= -\left(\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial}{\partial y_j}\sum_j w_{jk} y_j\right)\cdot\frac{\partial y_j}{\partial \xi_j}\cdot y_i = -\left(\sum_k \frac{\partial E}{\partial \xi_k} w_{jk}\right)\cdot\frac{\partial y_j}{\partial \xi_j}\cdot y_i =$$

$$= \left(\sum_k \delta_k w_{jk}\right)\cdot f'(\xi_j)\cdot y_i = \delta_j y_i$$



$$E = \frac{1}{2}\sum_p\sum_j \left(y_{j,p} - d_{j,p}\right)^2 \qquad y_j = f(\xi_j) = \frac{1}{1+e^{-\lambda \xi_j}}$$

Algoritmus spätného šírenia

1. **krok:** zvoľ náhodné hodnoty synaptických váh
2. **krok:** predlož nový trénovací vzor $(\vec{x}, \vec{d})$ v tvare

   (*vstup*, *požadovaný výstup*)
3. **krok:** vypočítaj skutočný výstup siete; aktivita neurónu

$$y_j = f(\xi_j) = \frac{1}{1+e^{-\lambda \xi_j}}, \text{ kde } \xi_j = \sum_i y_i w_{ij}$$

4. **krok:** Aktualizuj váhy postupne smerom od výstupnej vrstvy k vstupnej vrstve. Váhy sa adaptujú podľa:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \delta_j y_i + \alpha_m \left(w_{ij}(t) - w_{ij}(t-1) \right)$$

kde pre výstupné neuróny $\delta_j = (d_j - y_j) y_j (1 - y_j) \lambda$

a pre skryté neuróny $\delta_j = \lambda y_j (1 - y_j) \sum_k \delta_k w_{jk}$

5. **krok:** pokračuj **krokom 2**

Označenie

- $w_{ij}(t)$ ... váha z neurónu i do neurónu j v čase t
- α ... parameter učenia $\in<0,1>$
- α_m ... moment učenia $\in<0,1>$
- k ... označuje neuróny z vrstvy nad neurónom j
- ξ_j ... celkový vážený vstup neurónu j
- λ ... strmosť prenosovej funkcie

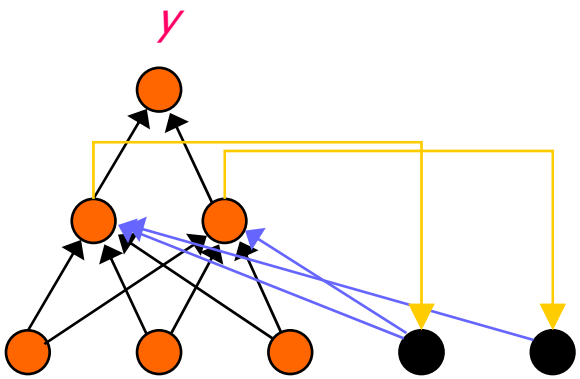
- Je to teda gradientná metóda, ktorá minimalizuje strednú kvadratickú odchýlku medzi požadovanou a skutočnou aktivitou všetkých výstupných neurónov systému

Rekurentné neurónové siete

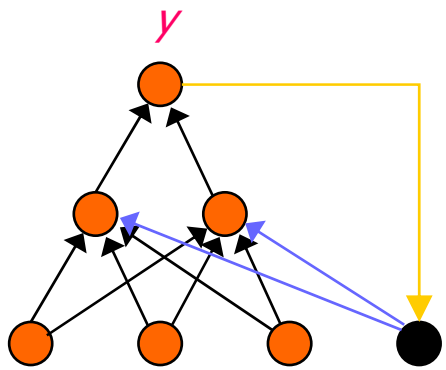
- detekcia časových závislostí v časovom rade
- okienko
  - 2 13 22 5 32 14 7 81 2 28 24
- Neurónové siete s časovým posunom (time delay neural network)
  - aké veľké okienko? nárast počtu váh
- inak: rekurentné siete – spätné spoje,
  - s diskrétnym časom, alebo
  - so spojitým časom (rekurentné signály a vstupy sú kontinuálne integrované)

Rekurentné siete

- Elman



- Jordan

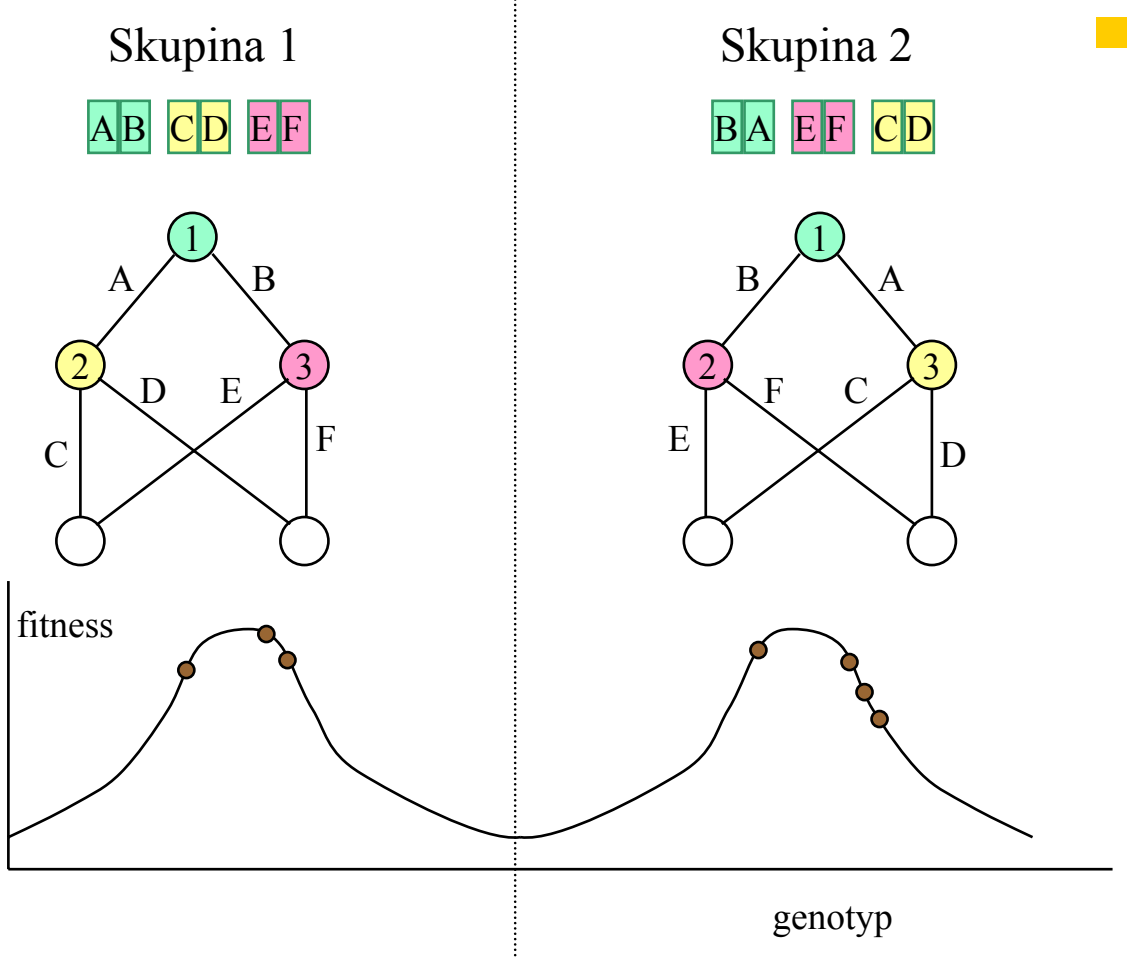


- **Elman** – pamäťové bunky uchovávajú stav skrytých neurónov z predchádzajúceho kroku
- **Jordan** – pamäťová bunka počíta svoj stav z výstupu siete a svojho stavu v predchádzajúcom kroku

Evolučný vývoj a NS

- Prečo NS?
  - relatívne hladký prehľadávaný priestor
  - rôzne úrovne evolučnej granularity:
    - fylogenetická (evolučná) – iba GA
    - vývojová (dozrievanie) – sieť „dorastie"
    - ontogenetická (celoživotné vzdelávanie) – sieť sa učí
  - môžu byť kombinované
  - NS poskytujú priamočiare zobrazenie senzory $\rightarrow$ motory
  - analógové vstupy na analógové alebo diskrétne výstupy
  - robustnosť voči šumu
- biologicky motivovaný model adaptívneho chovania

Evolučný vývoj NS
Problémy s učením



- súperiace skupiny
  - veľmi odlišné genotypy zodpovedajú sieťam s veľmi podobným chovaním ⇒ viacej maxím; kríženie vedie na slabých jedincov
  - v praxi to väčšinou nevadí, ale často sa volí nižšia pravdepodobnosť kríženia

Evolúcia váh a parametrov učenia

- Prečo
  - A) GA skúšajú celú populáciu sietí, nie len jednu sieť
  - B) bez obmedzení na architektúru, prenosovú funkciu, …
  - C) nepotrebujeme detailne poznať výstup siete pre každý vzor
- Kódovanie váh:
  - reťazec reálnych čísel
  - reťazec binárnych čísel – ako čísla s pevnou desatinnou čiarkou so zadanou presnosťou
    - Varianta: **dynamické kódovanie**
      - čísla v kóde na začiatku učenia najvýznamnejšie bity,
      - neskôr (keď je nájdené uchádzajúce riešenie) tie isté čísla kódujú menej významné bity
- **príklad:** *analýza sonarových signálov – GA lepšie než BP*
- iná stratégia: kombinácia evolúcie a učenia s učiteľom
  - BP je citlivé na počiatočné nastavenie váh
  - GA môže nájsť počiatočné nastavenie váh – fitness=chyba siete po doučení algoritmom BP (Darwinistická evolúcia)
- gén môže kódovať i iné parametre učenia

Evolúcia architektúry

- Priestor možných architektúr je nekonečný a nie je derivovateľný (nedá sa prehľadávať gradientnými metódami)
- kódovanie:
  - typicky „**nepriame kódovanie**“: kódujú sa iba niektoré charakteristiky siete – počet neurónov, pravdepodobnosť ich spojenia, typ prenosovej funkcie, …; váhy sa doladia učením
  - „priame kódovanie“
- **príklad 1:** *gen. kód obsahuje plán budovania siete – postupnosť segmentov zodpovedajúcich vrstvám; segment*
    - *vlastnosti uzlov (počet neurónov, prenosová funkcia, geometrické rozloženie)*
    - *vlastnosti vychádzajúcich spojov (hustota spojov, parametre učenia)*
  - *dekódovaná sieť sa učí BP*
  - *vo fitness sa dá penalizovať počet spojov alebo počet cyklov učenia*

Evolúcia architektúry

- **Príklad 2:** *genotyp kóduje prepisovacie pravidlá, výsledkom prepisovania je binárna matica spojov medzi neurónmi, váhy sa doučujú BP*

$$\varepsilon \to \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$A \to \begin{bmatrix} a & d \\ a & a \end{bmatrix} \quad B \to \begin{bmatrix} c & b \\ b & a \end{bmatrix} \quad C \to \begin{bmatrix} b & a \\ a & c \end{bmatrix} \quad D \to \begin{bmatrix} a & b \\ a & d \end{bmatrix}$$

$$a \to \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad b \to \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad c \to \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad d \to \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\varepsilon \Rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Rightarrow \begin{bmatrix} a & d & c & b \\ a & a & b & a \\ b & a & a & b \\ a & c & a & d \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Evolúcia architektúry

- **Príklad 3:** *„dozrievanie" – sieť rastie v dobe, keď sa robot pohybuje vo svete*
  - *NS je 2D pletivo, genotyp kóduje*
    - *pozíciu neurónu v pletive*
    - *smer rastu axónov*
    - *dĺžku axónov, …*
  - *axón, ktorý končí tam, kde nie sú žiadne neuróny, odumiera, inak vytvára synaptické spoje*
  - *rast závisí aj na aktivite robota*

Evolúcia pravidiel učenia

- **Pravidlo učenia (Heb):** *funkcia presynaptickej aktivity x_j, postsynaptickej aktivity y_i a aktuálnej váhy w_{ij}*

$$\Delta w_{ij} = \Phi\left(x_j, y_i, w_{ij}\right)$$

- predpíše sa ako lineárna kombinácia súčinov premenných

$$\Delta w_{ij} = a_1(x_j x_j) + a_2(x_j y_i) + a_3(x_j w_{ij}) + a_4(y_i y_i) + a_5(y_i w_{ij}) + a_6(w_{ij} w_{ij})$$

- a_n buď diskrétna hodnota z {-1,0,1} alebo spojitá hodnota z intervalu <-1,1>; genotyp obsahuje hodnoty konštánt
- fitness: váhy sa nastavia náhodne a sieť sa trénuje dekódovaným pravidlom učenia a jej fitness je jej výkon

GENETICKÉ PROGRAMOVANIE

- Genetický reťazec nekóduje riešenie problému, ale program riešiaci problém
- založené na LISPovskej reprezentácii programov
  - ♦ (*f*,*a*,*b*) … *f* funkcia, *a*,*b* argumenty
  - ♦ (+,2,5) zodpovedá výrazu 2+5
  - ♦ (+,2,(∗,3,7)) … 2+(3∗7)

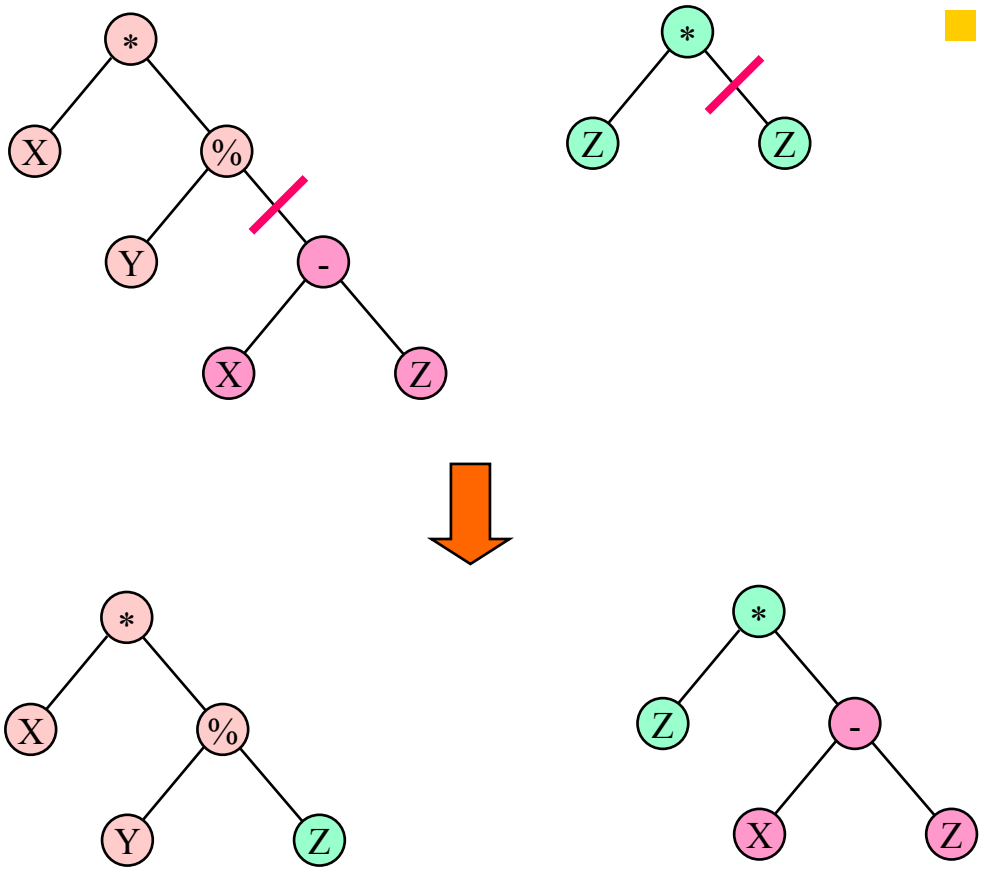
1. zvolí sa množina funkcií F a terminálov T. Napr.

   F={+,-,∗,%,IFLTE}, T={X,Y,Z,ℜ}, kde % je bezpečné delenie (vracia 1 pri deleni nulou), IFLTE je „if-less-than-or-equal", ℜ je náhodné číslo.

   Funkcie musia byť uzavreté, tj. všade definované

2. Náhodne sa vygeneruje počiatočná populácia programov
3. Ďalej ako GA – selekcia, kríženie, mutácia.

Genetické programovanie



- **Kríženie**: krížením 2 programov má vzniknúť zase funkčný (syntakticky správny) program – zámena podstromov medzi náhodne zvolenými uzlami
  - dvoch rodičov alebo
  - samotného jedinca

Genetické programovanie

- **Mutácia**: náhodne sa zvolí uzol a
  - buď sa zruší, alebo
  - na jeho mieste sa náhodne vytvorí nový podstrom
- mutácie sa nepoužívajú často, miesto toho sa pracuje s veľkými populáciami a veľkou pravdepodobnosťou kríženia
- **hypotéza**: *náhodné programy vyjadrené v LISPe majú vyššiu pravdepodobnosť, že sú správne, než iné reprezentácie*

- vylepšenie: *automaticky definované funkcie* – funkcia nahradzujúca celý podstrom

GRAMATICKÁ EVOLÚCIA

- Evolúcia programu v ľubovoľnom programovacom jazyku
- Používa lineárny genóm (GP stromy)
- Genóm = postupnosť kodónov (obyčajne 8 bitov)
- Zobrazenie genotyp → fenotyp konštruuje odvodenie programu podľa gramatiky:
  - Ak *X* je najľavejší neterminál v reťazci, tak číslo pravidla použitého na prepísanie *X* sa spočíta ako (*kodón* mod *počet pravidiel pre X*)
  - Kodóny sa aplikujú jeden za druhým, po poslednom kodóne sa pokračuje znova od prvého
  - Ak po dopredu zadanom počte „cyklov" cez celý kodón reťazec stále obsahuje neterminály, tak jedinec je označený ako nepoužiteľný a dostáva minimálnu možnú hodnotu fitness

Príklad gramatickej evolúcie

Gramatika

```
<expr> ::= <expr><op><expr> (0)
         | <var>            (1)

<op> ::= + (0)
       | - (1)
       | * (2)

<var> ::= x (0)
        | y (1)
```

Genotyp → fenotyp

```
154:  <expr>
```

Genotyp

154	37	211	92	254	79	143

EVOLÚCIA ROBOTOV

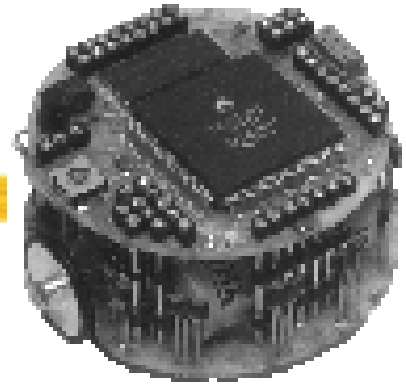
- Okruhy problémov:
  - mechanická odolnosť: nárazy do stien vo veľkej rýchlosti
  - prívod energie: batérie nestačia
  - analýza: vyvinuté riadiace systémy môžu byť veľmi zložité; na ich popis je možné využiť metódy z neuroetológie
  - časová náročnosť: hodiny, dni i týždne
  - návrh fitness funkcie

a) evolúcia fyzických robotov – Khepera

b) simulovaná evolúcia – minimalistická simulácia

c) priestor fitness

Khepera



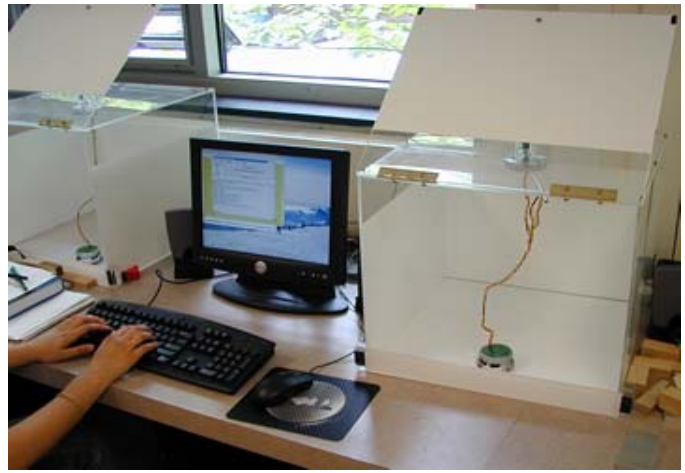
- 1993:
- www: www.k-team.com
- simulátor: http://diwww.epfl.ch/lami/team/michel/khep-sim/
- parametre:
  - priemer 55 mm
  - hmotnosť 70 g
  - dve samostatne poháňané kolá s inkrementálnymi enkodérmi (600 pulzov/otáčka)
  - 8 aktívnych infračervených senzorov;
    - 2 režimy
      - „pasívny" - meria infračervené svetlo z okolia; pozor na zdroje tepla
      - „aktívny" - vysiela infra svetlo a meria odrazené svetlo – detekuje biely papier na 5 cm; distančný senzor – detektor prekážok
    - režimy sa prepínajú softwarovo a veľmi rýchle – je možné snímať oboje naraz

Khepera

- Parametere – pokračovanie
  - akumulátory – 30-40 min. samostatnej činnosti
  - CPU Motorola MC68331, 128 kB EEPROM, 256 kB statickej RAM
  - A/D prevodník, RS232
- miniaturizácia:
  - pracovný stôl 0,9 x 1,8 m je pre Kheperu priestor ekvivalentný tenisovému ihrisku pre robota s priemerom 55 cm
  - na malom priestore sa dá jednoduchšie (technicky) sledovať
  - náraz 55 mm Khepery pri rýchlosti 55 mm/s verzus náraz 1 m robota pri rýchlosti 1 m/s
- modularita: možnosti rozšírenia
  - nástavce: lineárne videnie – 64-pixelový riadok, uhol 36°
  - chápadlo s 2 stupňami voľnosti
  - rozširujúca doska

Khepera

- Pripojenie rotačnými kontaktmi
  - spojenie s počítačom
  - dlhodobá práca



- 2 režimy
  - interaktívny – riadenie z počítača – častejšie
  - autonómny – riadenie na robote
- Analýza vyvinutého riadiaceho systému nie je možná oddelene od jeho okolia, preto je užitočné mať možnosť presne merať polohu robota

Ďalšie roboty

- **Koala**: príbuzný robot
  - rozmery: d x š x v 32 x 32 x 20 cm, 3 kg
  - 2 motory, 6 kôl, 16 infračervených senzorov
  - softwarovo kompatibilná s Kheperou
- **Koala II**: novšia verzia

- Khepera II
  - Motorola 68331, 25MHz
  - priemer 70 mm
- Hemison

Ďalšie roboty – pokračovanie

- Khepera III (Linux)

- e-Puck

Ďalšie roboty – pokračovanie

- **KiloBot**
  - http://www.k-team.com/mobile-robotics-products/kilobot/introduction
  - Priemer 33 mm, výška 34 cm i s nohami
  - Vibračné motory
  - Komunikácia so susednými robotmi do vzdialenosti 7 cm (odrazom infračerveného svetla od podlahy