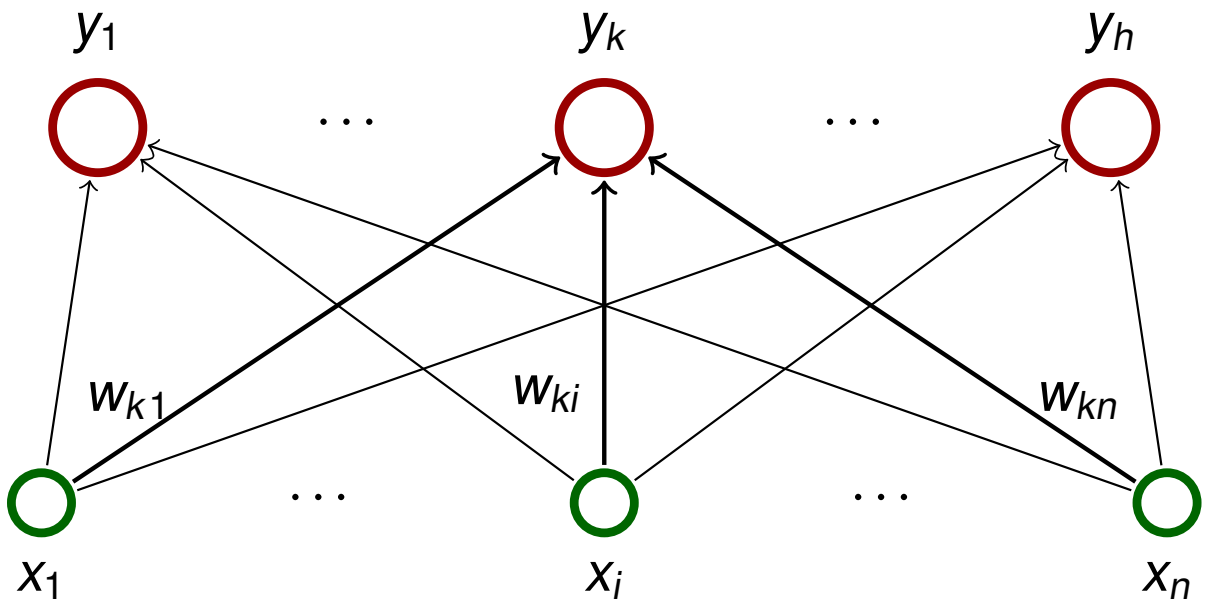
Kohonenova mapa - opakování

Organizační dynamika: Jednovrstvá síť



- Mezi neurony je navíc zavedena **topologická struktura** (tj. neurony tvoří uzly neorientovaného grafu).
- V drtivé většině případů je tato struktura buď jednorozměrná řada jednotek nebo dvojrozměrná mřížka.

Kohonenova mapa

Aktivní dynamika: Pro vstup $\vec{x} \in \mathbb{R}^n$ a $k = 1, \ldots, h$:

$$y_k = \begin{cases} 1 & k = \arg\min_{i=1,\ldots,h} \left\| \vec{x} - \vec{w}_i \right\| \\ 0 & \text{jinak} \end{cases}$$

Adaptivní dynamika: V adaptivním režimu využijeme topologickou strukturu.

- Označme $d(c, k)$ délku nejkratší cesty z neuronu c do neuronu k v topologické struktuře.
- Pro neuron c a dané $s \in \mathbb{N}_0$ definujeme **okolí** neuronu c velikosti s takto: $N_s(c) = \{k \mid d(c, k) \leq s\}$

V kroku t po předložení vstupu $\vec{x}_t$ adaptujeme každé $\vec{w}_k$ takto:

$$\vec{w}_k^{(t)} = \begin{cases} \vec{w}_k^{(t-1)} + \theta \cdot \left(\vec{x}_t - \vec{w}_k^{(t-1)} \right) & k \in N_s(c) \\ \vec{w}_k^{(t-1)} & \text{jinak} \end{cases}$$

kde $c = \arg\min_{i=1,\ldots,h} \left\| \vec{x}_t - \vec{w}_i^{(t-1)} \right\|$ a kde $\theta \in \mathbb{R}$ a $s \in \mathbb{N}_0$ jsou parametry, které se mohou v průběhu učení měnit.

Kohonenova mapa - adaptivní dynamika

Obecnější formulace adaptivní dynamiky:

$$\vec{w}_k^{(t)} = \vec{w}_k^{(t-1)} + \Theta(c,k) \cdot \left(\vec{x} - \vec{w}_k^{(t-1)}\right)$$

kde $c = \arg\min_{i=1,\ldots,h} \left\|\vec{x}_t - \vec{w}_i^{(t-1)}\right\|$. Předchozí případ potom odpovídá

$$\Theta(c,k) = \begin{cases} \theta & k \in N_s(c) \\ 0 & \text{jinak} \end{cases}$$

Obvykle se používá plynulejší přechod mezi nenulovými a nulovými hodnotami, např.

$$\Theta(c,k) = \theta_0 \cdot \exp\left(\frac{-d(c,k)^2}{\sigma^2}\right)$$

kde $\theta_0 \in \mathbb{R}$ určuje maximální míru změny vah a $\sigma \in \mathbb{R}$ je šířka (oba parametry se mohou v průběhu měnit).

LVQ - klasifikace - opakování

Přepodkládejme, že máme náhodně generované vzory tvaru $(\vec{x}_t, d_t)$ kde $\vec{x}_t \in \mathbb{R}^n$ je **vektor vlastností** a $d_t \in \{C_1, \ldots, C_q\}$ určuje jednu z q **tříd**.

Cílem je klasifikovat vstupy do tříd na základě znalosti vlastností, tj. každému $\vec{x}_t$ chceme přiřadit třídu tak, abychom minimalizovali pravděpodobnost chyby.

Př.: Po pásu jede náhodně „naházené“ ovoce dvou druhů: jablka a melouny. Námi sledovaná data budou $(\vec{x}_t, d_t)$ kde

- $\vec{x}_t \in \mathbb{R}^2$, první vlastnost je hmotnost, druhá je průměr
- d_t je buď J nebo M v závislosti na tom, jestli je daný kus jablko nebo meloun

Připouštíme možnost, že se najde jablko a meloun se stejnými mírami.

Cílem je třídit ovoce na základě hmotnosti a průměru.

LVQ - klasifikace - opakování

Využijeme vektorovou kvantizaci (Kohonenovu mapu) takto:

1. Natrénujeme mapu na vektorech vlastností $\vec{x}_t$ kde $t = 1, \ldots, \ell$.
2. Jednotlivé neurony označíme třídami. Třídu v_c neuronu c nalezneme takto:

   Pro každý neuron c a třídu C_i spočítáme četnost vzorů třídy C_i, které jsou reprezentovány neuronem c.

   Toto lze provést jedním průchodem přes vzory

   Neuronu c přiřadíme třídu s maximální četností.
3. Doladíme síť pomocí algoritmu LVQ.

Klasifikaci pomocí natrénované sítě provádíme takto: Na vektor vlastností $\vec{x}$ aplikujeme natrénovanou síť v aktivním režimu. Právě jeden neuron, řekněme c, bude mít výstup 1. Potom vstup zařadíme do třídy v_c.

LVQ1 - opakování

Postupně projdi tréninkové vzory. Pro vzor $(\vec{x}_t, d_t)$ urči nejbližší neuron c

$$c = \arg \min_{i=1,\ldots,h} \left\| \vec{x}_t - \vec{w}_i \right\|$$

Potom uprav váhy neuronu c takto:

$$\vec{w}_c^{(t)} = \begin{cases} \vec{w}_c^{(t-1)} + \alpha(\vec{x}_t - \vec{w}_c^{(t-1)}) & d_t = v_c \\ \vec{w}_c^{(t-1)} - \alpha(\vec{x}_t - \vec{w}_c^{(t-1)}) & d_t \neq v_c \end{cases}$$

Parametr α by měl být od počátku malý (cca 0.01 – 0.02) a postupně klesnout k 0.

Hranice mezi třídami vytvořená pomocí LVQ1 je poměrně dobrou aproximací *bayesovské rozhodovací hranice*.

Klasická aplikace - fonetický psací stroj

Zdroj: The Self-Organizing Map. T. Kohonen, IEEE, 1990

Cílem je automaticky zapisovat diktovaný text (v originále buď Finština nebo Japonština)

Zvuk byl nejprve předzpracován:

- 5.3 kHz low-pass filter
- digitalizace (12-bit, 13.02-kHz sampling rate)
- spektrální rozklad (Fourierova transformace každých 9.83 ms)
- spektrum „zkombinováno“ do 15 komponent od 200 Hz po 5 kHz -> 15 dimenzionální vstupy pro Kohonenovu mapu

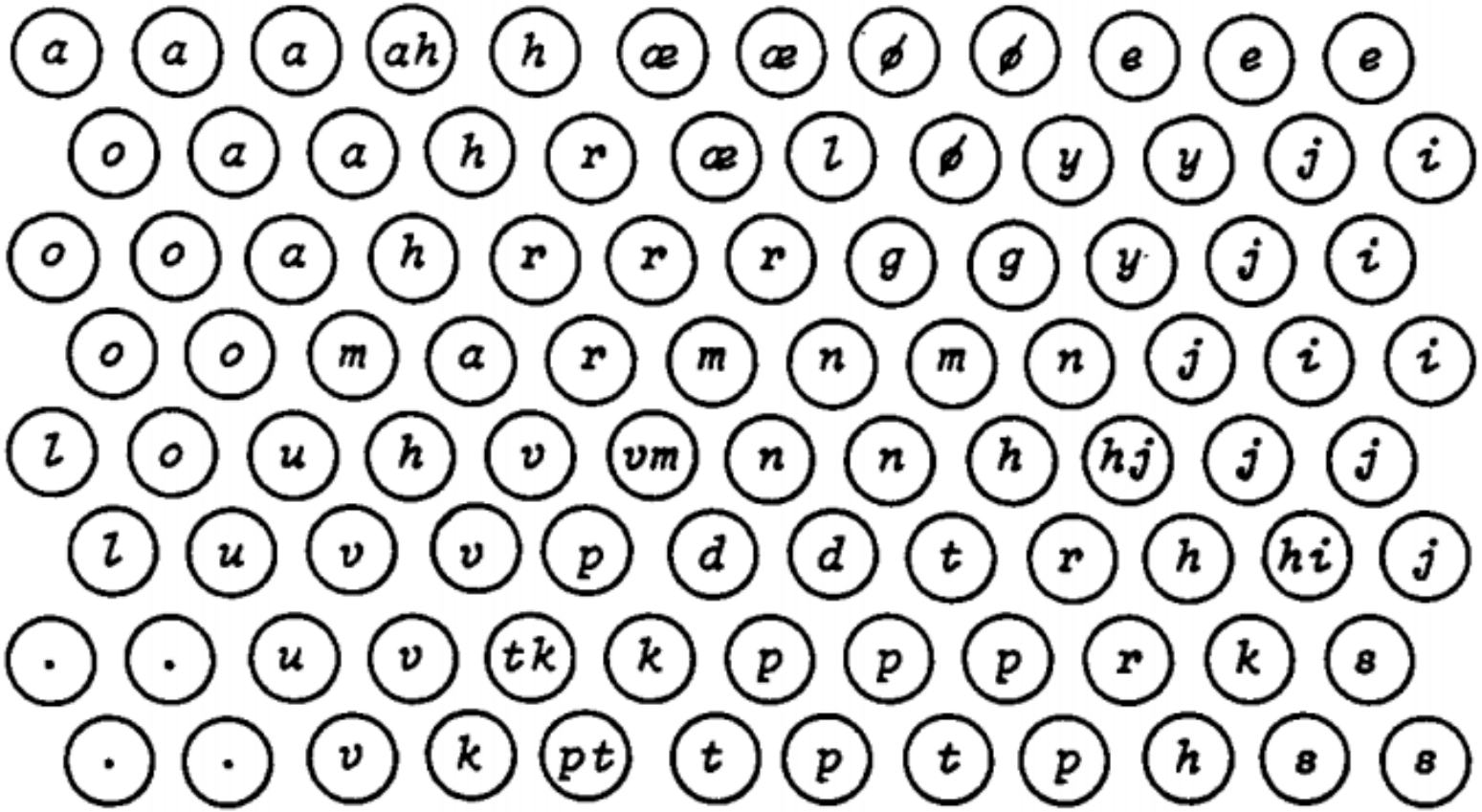
  Byla také provedena normalizace na nulové střední hodnoty komponent a konstantní délku vektorů.

Klasická aplikace - fonetický psací stroj

Vytvořena Kohonenova mapa 8×12 jednotek, hexagonální

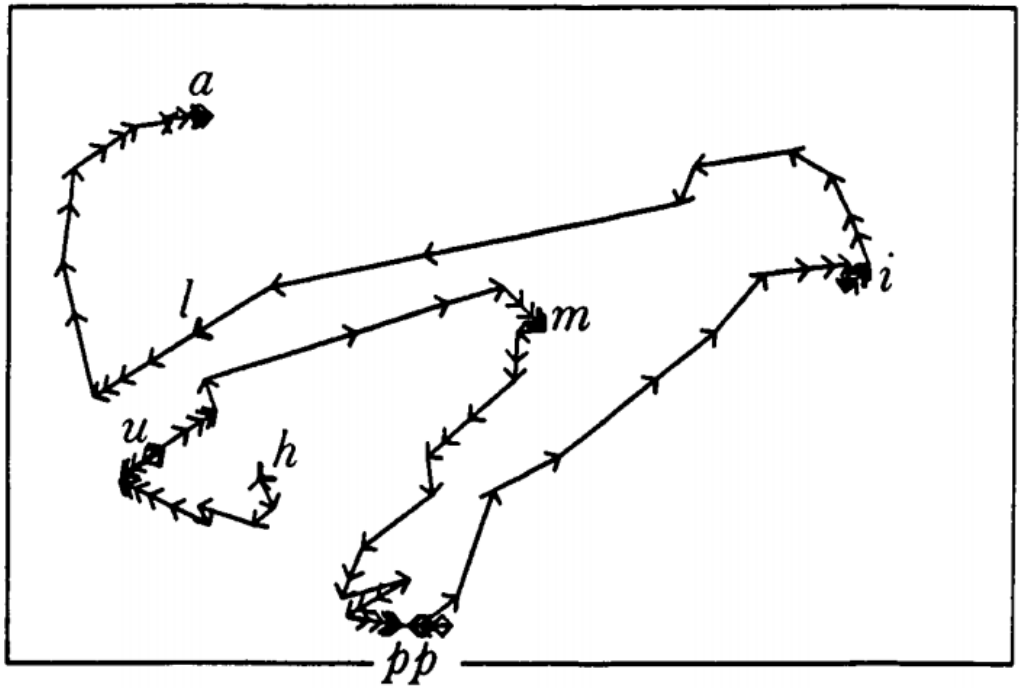
Trénována na toku 15 dim vektorů v jejich přirozeném pořadí (pouze samoorganizace)

Poté přiřazeny fonémy jednotlivým neuronům, doladěno pomocí LVQ.



Klasická aplikace - fonetický psací stroj

Mapu lze vyžít k vizualizaci mluveného slova (zde *Humpilla*):



Může být využito k rozpoznávání mluvených slov s použitím klasifikátoru natrénovaného na mnoha překladech.

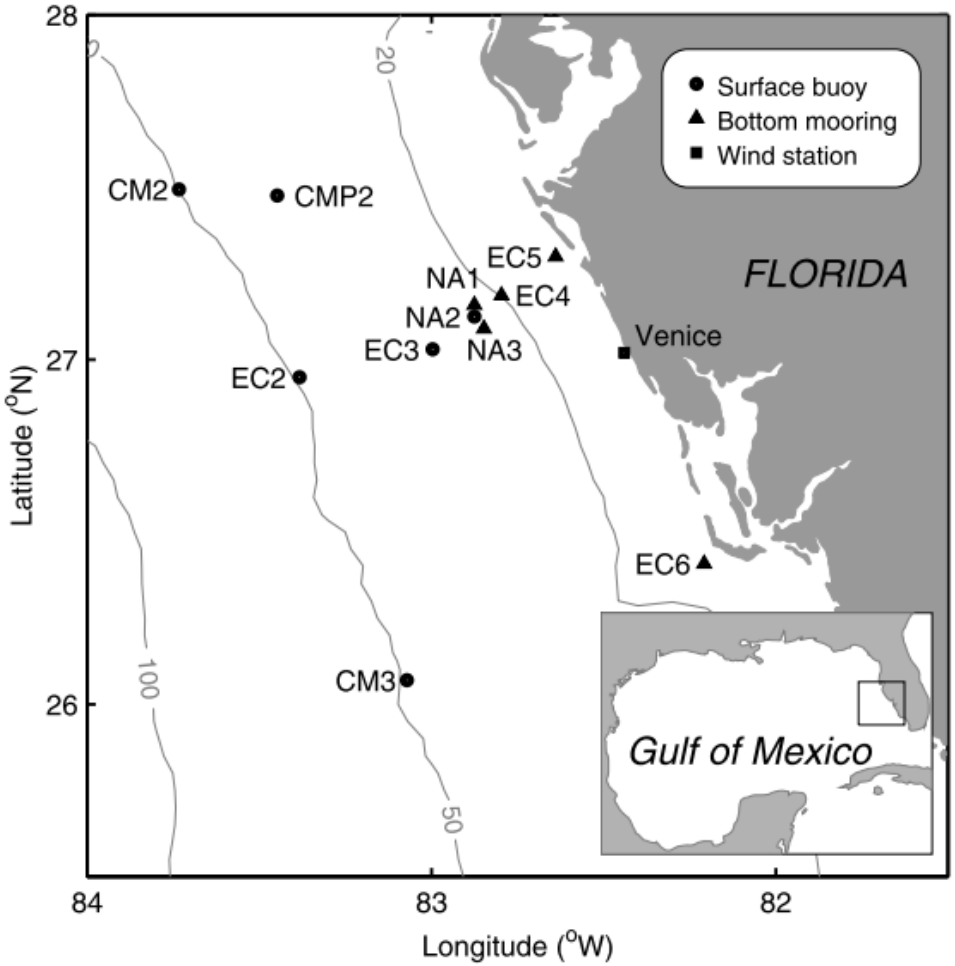
Table 2 Speech Recognition Experiments with Error Percentages for Independent Test Data

	Parametric Bayes	kNN	LVQ1	LVQ2	LVQ3
Test 1	12.1	12.0	10.2	9.8	9.6
Test 2	13.8	12.1	13.2	12.0	11.5

Oceánografická data

Zdroj: Patterns of ocean current variability on the West Florida Shelf using the self-organizing map. Y. Liu a R. H. Weisberg, JOURNAL OF GEOPHYSICAL RESEARCH, 2005

Zkoumá se vývoj proudění vod v oceánu kolem pobřeží Floridy



Oceánografická data

- 11 měřicích stanic, 3 hloubky (hladina, dno, mezi)
- data: 2D vektory rychlosti (a směru) proudění
- měřeno po hodinách, 25585 hodin

Celkově tedy 25585 řádků 66 dimenzionálních položek.

Kohonenova mapa:

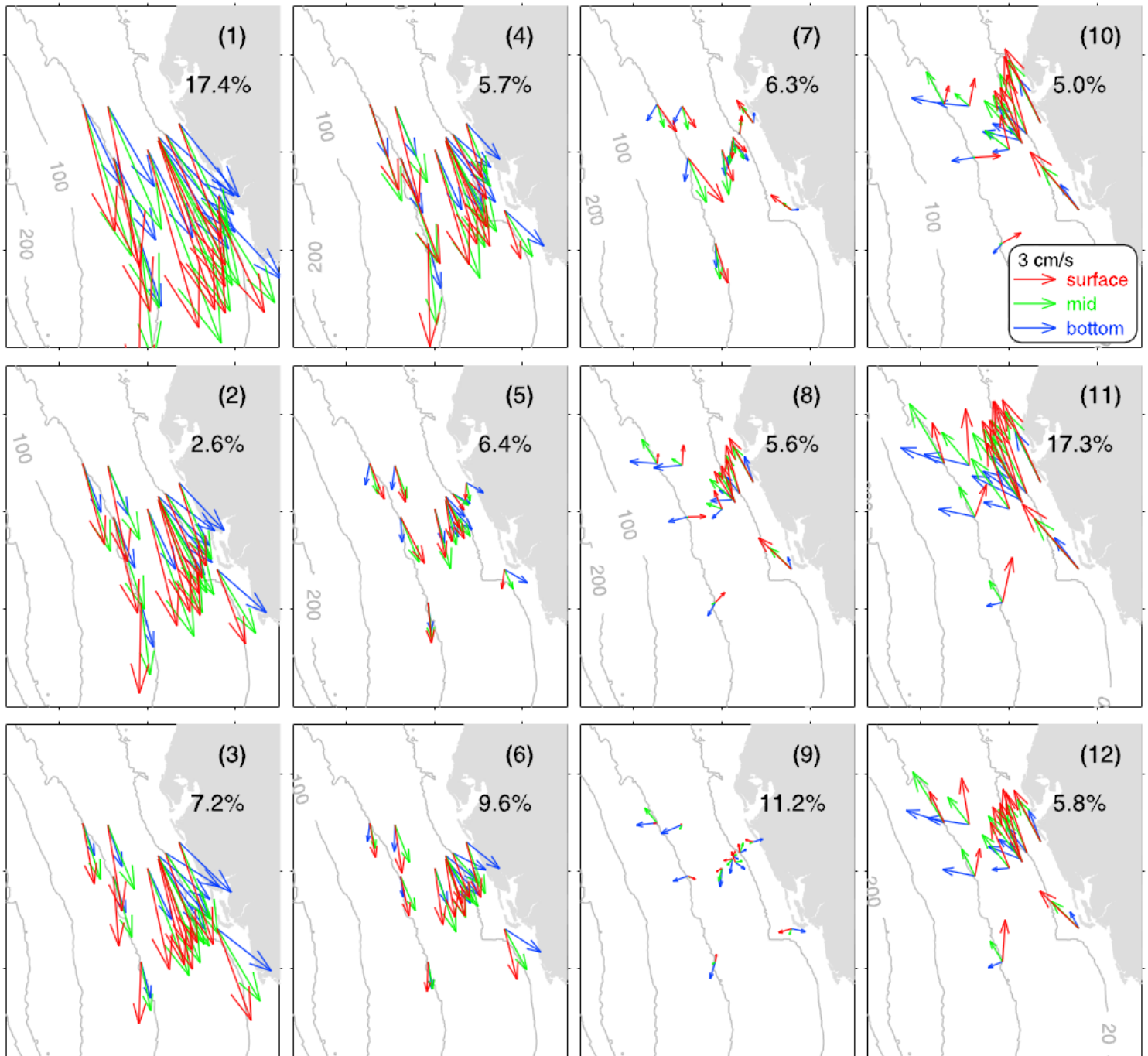
- mřížka 3×4
- okolí dána Gaussovou funkcí

$$\Theta(c,k) = \theta_0 \cdot \exp\left(\frac{-d(c,k)^2}{\sigma^2}\right)$$

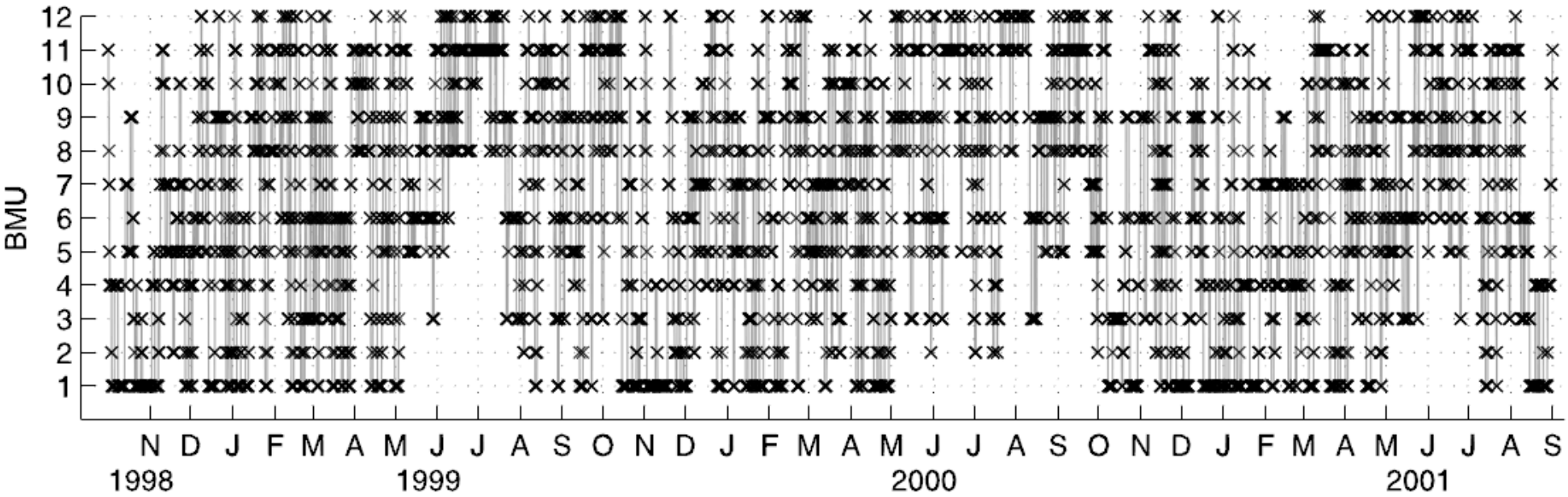
se zmenšující se šířkou

(navíc je tam lineárně se zmenšující rychlost učení, kterou se násobí změna polohy neuronů)

Oceánografická data - mapa

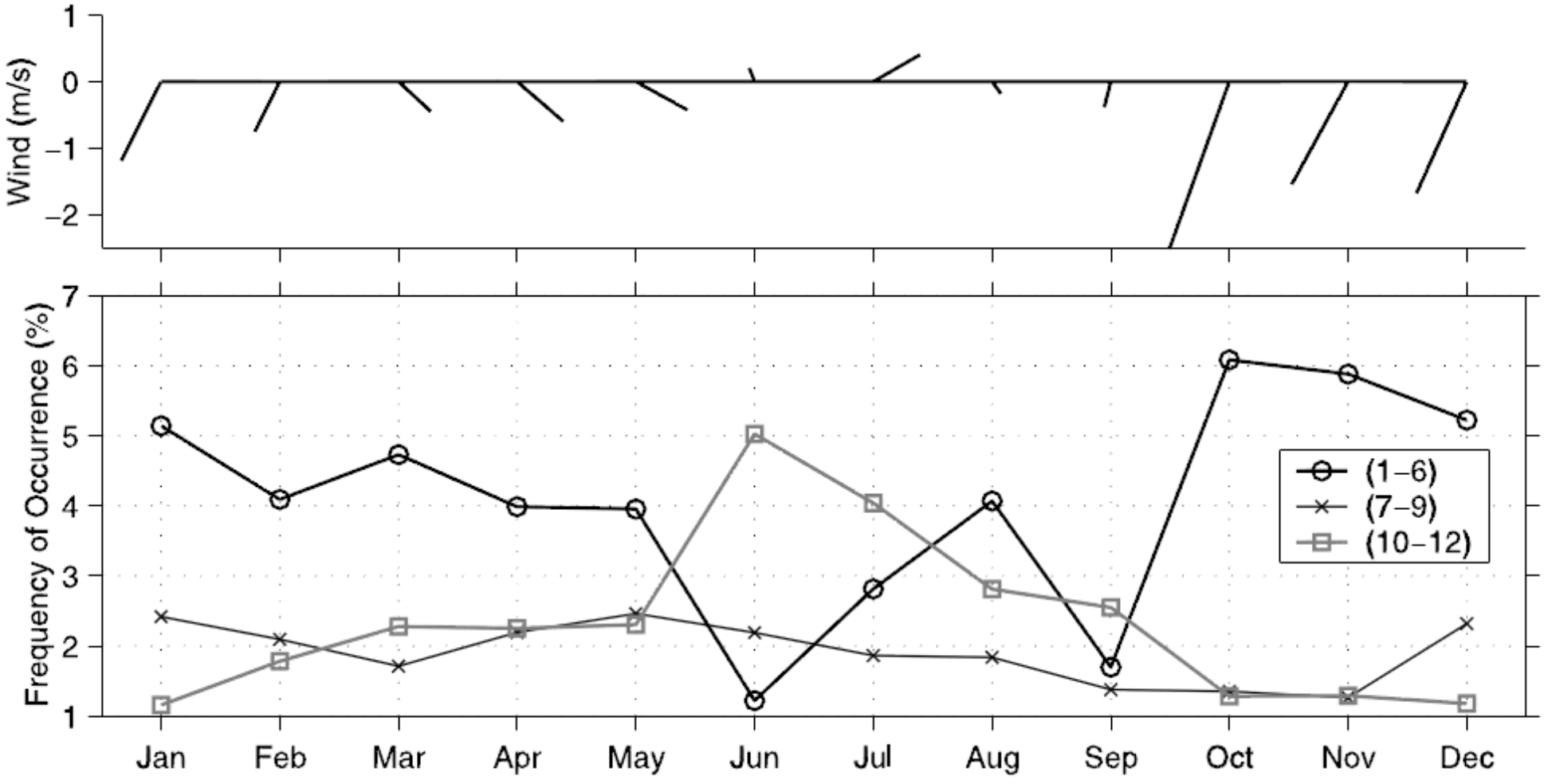


Oceánografická data



- křížky označují „vítězné“ neurony (po hodinách)
- ovlivněno lokálními fluktuacemi
- pozorovatelný trend:
  - v zimě neurony 1-6 (jiho-východ)
  - v létě neurony 10-12 (severo-západ)

Oceánografická data



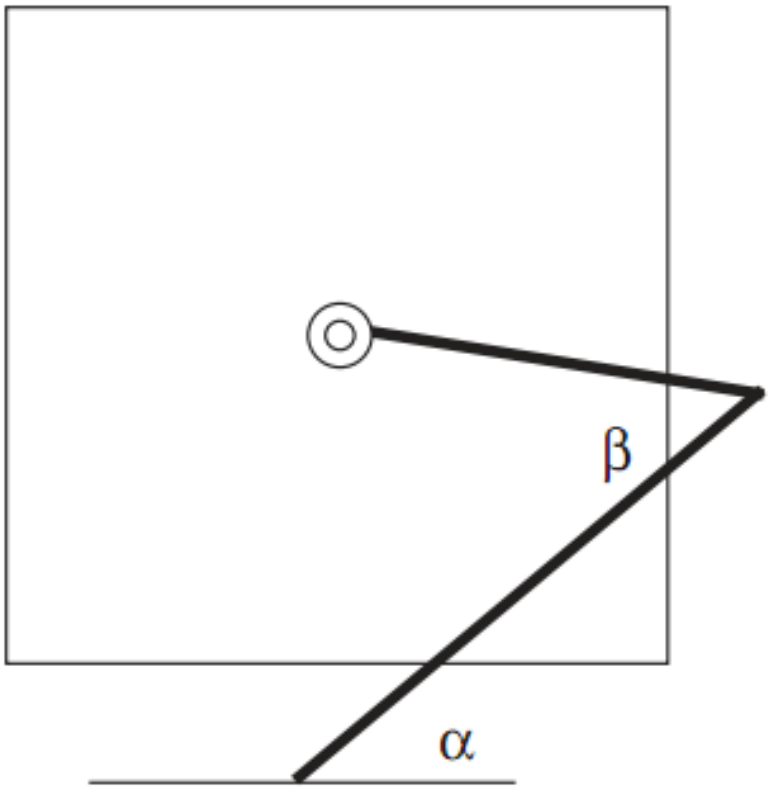
Srovnání směru větru a proudění vody (znatelná korelace)

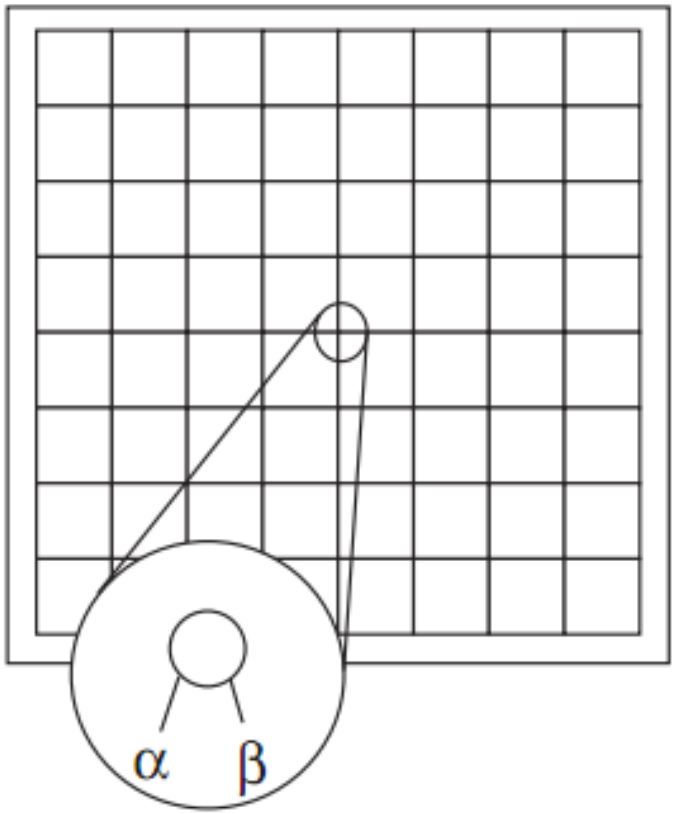
- vítr: směr čáry = směr větru ; délka čáry = intenzita
- proudění vody v procentech úspěšnosti skupin neuronů

Robotická ruka

Zdroj: Neural Networks - A Systematic Introduction, R. Rojas, Springer, 1996

Cílem je řídit robotickou ruku s jedním kloubem na čtvercové pracovní ploše:





Pozice ruky je dána dvěma úhly α a β.

Robotická ruka - idea

Topologie Kohonenovy mapy zde odpovídá *pracovní ploše*

Váhy neuronů odpovídají příslušným úhlům α a β.

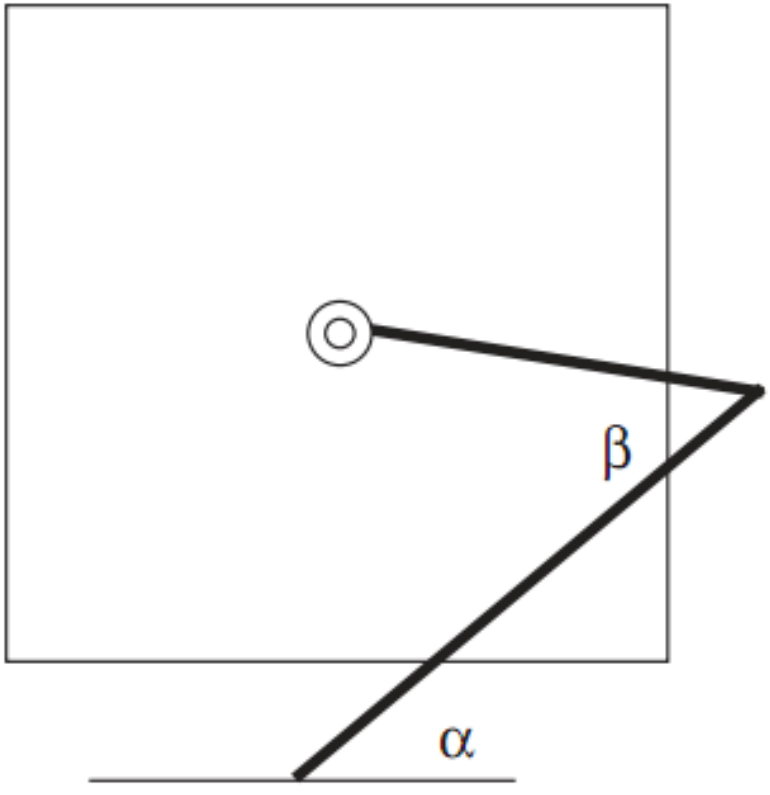
Adaptace:

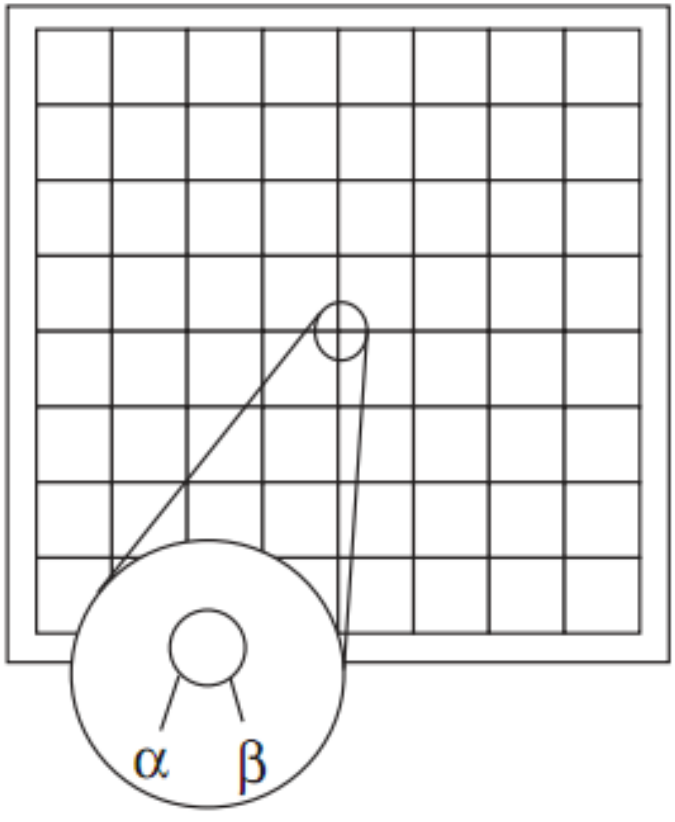
- ruka je nastavena na náhodné místo na pracovní ploše, tj. ukazuje na vybraný neuron v topologické struktuře
- váhy tohoto neuronu (a neuronů v jeho okolí) jsou potom adaptovány podle aktuálních úhlů ruky

Ruka se tedy naučí mapovat souřadnice na pracovní ploše (topologie) na úhlové souřadnice.

Robotická ruka - idea

Cestu z bodu *A* do bodu *B* na pracovní ploše lze snadno namapovat na cestu v úhlovém prostoru:





Chybějící body v úhlovém prostoru se dostanou interpolací.

Analýza pohádek bratří Grimmů

Zdroj: Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map. T. Kohonen, T. Honkela a V. Pulkki, ICANN, 1995

Cílem je vizualizovat syntaktické a sémantické kategorie slov v pohádkách (v závislosti na kontextu).

Vstup: Pohádky bratří Grimmů (srozumitelně kódované pomocí proudu 270-dimenzionálních vektorů)

- uvažují se trojice slov (předchůdce, klíč, následník)
- každá položka v trojici se zakóduje náhodně vybraným 90-dimenzionálním reálným vektorem (trojice má tedy dimenzi 270)

Síť: Kohonenova mapa, 42×36 neuronů, váhy neuronů jsou tvaru $w = (w_p, w_k, w_n)$ kde $w_p, w_k, w_n \in \mathbb{R}^{90}$.

Analýza pohádek bratří Grimmů

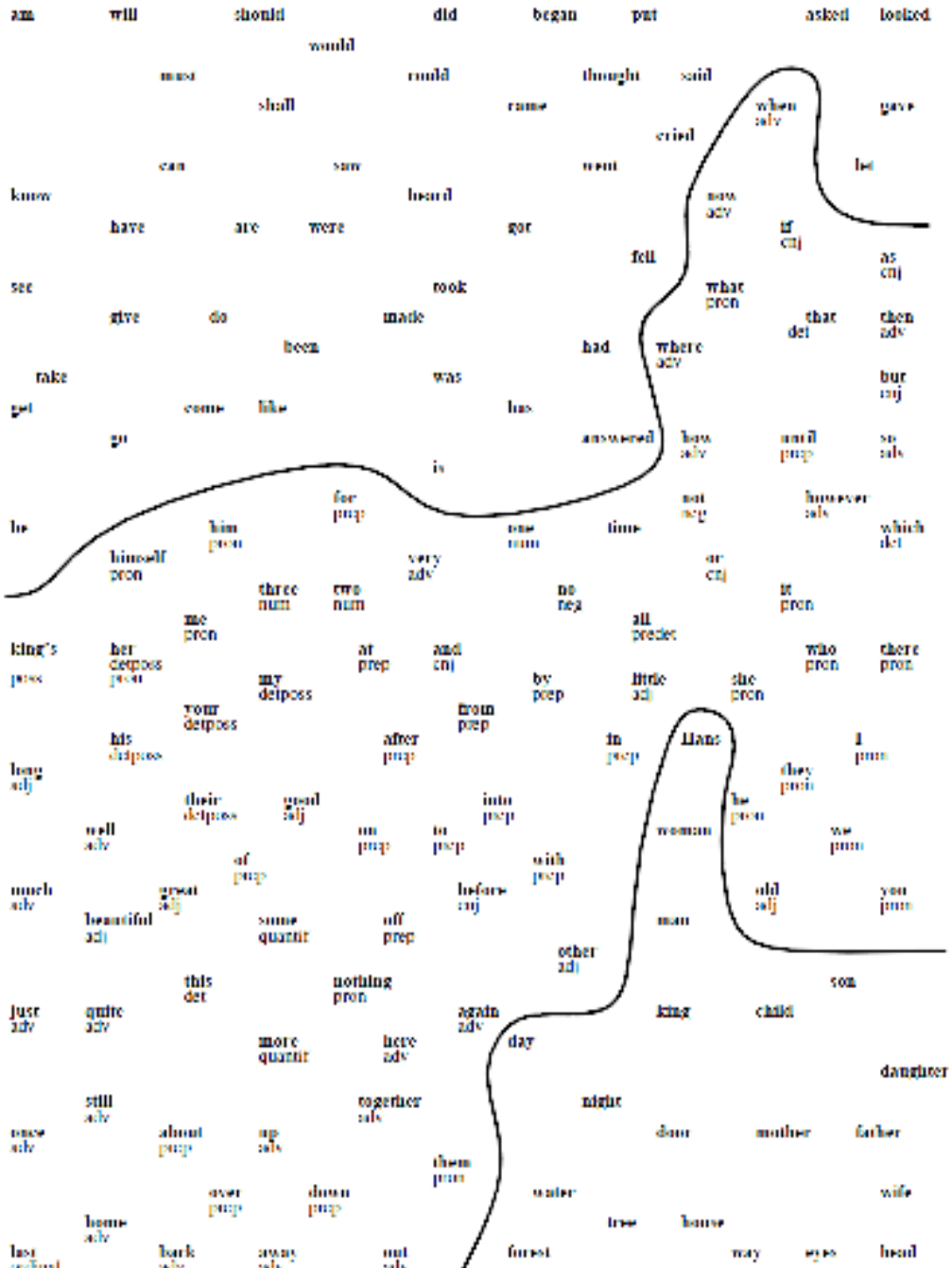
Adaptace:
Síť je inicializována takto: nejprve je identifikován dvojrozměrný podprostor prostoru $\mathbb{R}^{270}$ jehož osy jsou ve směru největší variability dat. V tomto podprostoru tvoří váhy neuronů pravidelnou mřížku se středem ve střední hodnotě dat (rozloha mřížky je odvozena z variability dat)

Síť je natrénována na trojicích po sobě jdoucích slov z pohádek
Pozn. Trojice byly agregovány pomocí průměrování předchůdců a následníků apod.

Hrubé učení: 600 000 iterací; Dolaďování: 400 000

Nakonec je vybráno 150 nejčastěji použitých slov, kterými jsou označeny neurony:
slovem u je označen ten neuron, pro jehož váhy $w = (w_p, w_k, w_n)$ platí, že w_k je nejblíže kódu slova u.

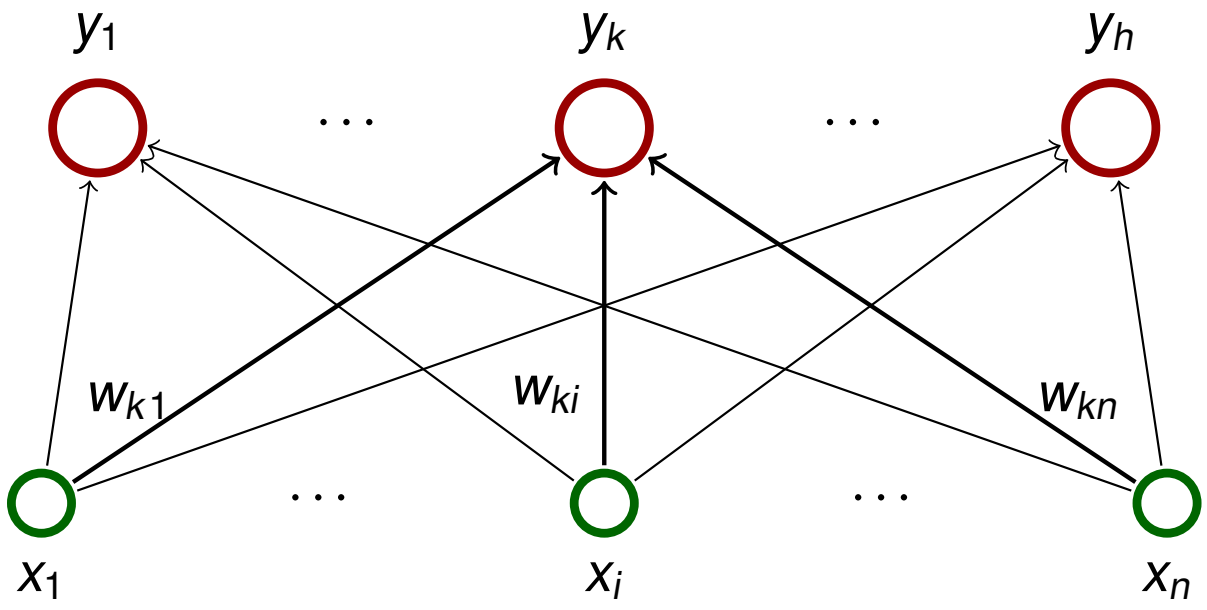
Analýza pohádek bratří Grimmů - výsledek



Sítě s nervovým plynem :-)

(Neural gas networks)

Organizační dynamika: Jednovrstvá síť



Aktivní dynamika: Pro vstup $\vec{x} \in \mathbb{R}^n$ a $k = 1, \ldots, h$:

$$y_k = \begin{cases} 1 & k = \arg\min_{i=1,\ldots,h} \left\| \vec{x} - \vec{w}_i \right\| \\ 0 & \text{jinak} \end{cases}$$

Neural gas

Adaptivní dynamika:
Inicializace: náhodně vygeneruj váhy neuronů.

V kroku t po předložení $\vec{x}_t$ proveď:

- uspořádej neurony od nejbližšího k nejvzdálenějšímu, tj. vypočti $i_1, i_2, \ldots i_h$ t.ž.

$$\left\|\vec{x}_t - \vec{w}_{i_1}^{(t-1)}\right\| < \left\|\vec{x}_t - \vec{w}_{i_2}^{(t-1)}\right\| < \cdots < \left\|\vec{x}_t - \vec{w}_{i_h}^{(t-1)}\right\|$$

  Dále pro každý neuron j urči počet k_j ostatních neuronů, které jsou blíže k $\vec{x}_t$ než neuron j, tj.

$$k_j = \left|\left\{ i \quad | \quad \left\|\vec{x}_t - \vec{w}_i^{(t-1)}\right\| < \left\|\vec{x}_t - \vec{w}_j^{(t-1)}\right\| \right\}\right|$$

- adaptuj váhy všech neuronů j takto:

$$\vec{w}_j^{(t-1)} + \varepsilon \cdot e^{-k_j/\lambda} \cdot \left(\vec{x}_t - \vec{w}_j^{(t-1)}\right)$$

  Zde $\varepsilon, \lambda \in \mathbb{R}^+$ jsou parametry učení.

Neural gas s Hebbovským učením

Přidáme topologickou strukturu: neorientovaný graf G

G se bude konstruovat v průběhu učení (neovlivňuje učení), jeho smyslem je zvýraznit sousednost neuronů (podobně jako v naučené Kohonenově mapě)

Adaptivní dynamika je stejná jako pro neural gas net, pouze přidáme manipulaci s grafem G

- iniciálně nemá G žádné hrany, v průběhu výpočtu budou mít hrany přiřazen **věk** $t_{ij} \in \mathbb{N}_0$
- v každém kroku:
  - přidáme hranu mezi nejbližším a druhým nejbližším neuronem, tj. $\{i_1, i_2\}$, ke grafu G a vynulujeme věk této hrany: $t_{i_1 i_2} := 0$
  - zvýšíme věk ostatním hranám
  - vyhodíme staré hrany (tj. ty co mají věk vyšší než dané T)

Pozn.: z důvodu vypočetní náročnosti se aktualizace věku a vyhazování hran obvykle dělá pouze pro hrany vedoucí z neuronu i_1.

RBF sítě

Organizační dynamika:

- Dvouvrstvá síť (výstupní neuron má bias)
- vstupy: $\vec{x} = (x_1, \ldots, x_n)$
- m skrytých neuronů

  Jejich hodnoty značíme $\phi_0, \phi_1, \ldots, \phi_m$ kde $\phi_0 = 1$ je formální jednotkový vstup pro výstupní neuron (viz. aktivní dynamika)

  Občas budu psát $\phi_0(\vec{x}), \phi_1(\vec{x}), \ldots, \phi_m(\vec{x})$ abych zdůraznil, že se jedná o hodnoty skrytých neuronů pro vstup sítě $\vec{x}$.
- jeden výstup: y
  (pro jednoduchost; lze zobecnit na libovolný počet)

Parametry sítě:

- Výstupní neuron je jako ADALINE: má váhový vektor $\vec{w} \in \mathbb{R}^{m+1}$
- Každý skrytý neuron má tzv. střed $\vec{c}_i \in \mathbb{R}^n$ (odpovídá vahám ve standardní vícevrstvé síti) a šířku (odpovídá strmosti ve standardní síti, tj. je to parametr aktivační funkce)

RBF sítě

Aktivní dynamika: Vnitřní potenciály skrytých neuronů:

$$\xi_j = \left\| \vec{x} - \vec{c}_j \right\|$$

Zde $\vec{c}_j$ je **střed** skrytého neuronu j.

Aktivační funkce skrytých neuronů $\varphi_0, \varphi_1, \ldots, \varphi_m$ jsou obecně libovolné diferencovatelné (a $\varphi_0 \equiv 1$). Budeme uvažovat:

$$\varphi_j(\xi_j) = \exp\left(-\frac{\xi_j}{2\sigma_j^2}\right) \qquad \left(= \phi_j(\vec{x}) \right)$$

Zde σ_j je **šířka** skrytého neuronu j.

Vnitřní potenciál výstupního neuronu:

$$\xi = \sum_{j=0}^{m} w_j \cdot \phi_j$$

Zde w_j jsou reálné **váhy**.

Aktivační funkce výstupu je identita: $y = \xi$.

RBF sítě

Funkce sítě:

$$y(\vec{x}) = w_0 + \sum_{j=1}^{m} w_j \cdot \exp\left(-\frac{\left\|\vec{x} - \vec{c}_j\right\|}{2\sigma_j^2}\right)$$

Tvrzení: Pro každou spojitou funkci $f : \mathbb{R}^n \to \mathbb{R}$ a každé $\varepsilon > 0$ existuje RBF síť taková, že její funkce y splňuje:

$$|f(\vec{x}) - y(\vec{x})| < \varepsilon \qquad \forall \vec{x} \in \mathbb{R}^n$$

RBF sítě

Adaptivní dynamika: Dána množina **tréninkových vzorů**

$$\mathcal{T} = \left\{\left(\vec{x}_1, d_1\right), \left(\vec{x}_2, d_2\right), \ldots, \left(\vec{x}_p, d_p\right)\right\}$$

Zde $\vec{x}_k = (x_{k1} \ldots, x_{kn}) \in \mathbb{R}^n$ je vstup k-tého vzoru a $d_k \in \mathbb{R}$ je očekávaný výstup.

Chybová funkce:

$$E = \frac{1}{2} \sum_{k=1}^{p} \left(y(\vec{x}_k) - d_k\right)^2$$

Cílem je nalézt následující parametry tak, aby byla chyba E minimalizována:

- váhy w_j výstupního neuronu
- středy $\vec{c}_j$ skrytých neuronů
- šířky σ_j skrytých neuronů

RBF - dvoufázový trénink

Velkou výhodou RBF sítí je, že jejich trénink lze rozdělit do dvou (téměř) nezávislých fází:

- trénink skrytých neuronů, tj.
  - umístění středů $\vec{c}_j$
  - nastavení velikosti šířek σ_j
- trénink vah w_j výstupního neuronu

RBF - středy a šířky

Rozmístění středů:
Pozorování: Máme velké množství dat (vstupů), pro malou část z nich máme předepsán požadovaný výstup (tréninková množina). Je dobré umístit středy neuronů do oblastí, které obsahují hodně datových bodů.

Středy lze proto umístit pomocí učení bez učitele - Kohonenovo učení (tj. k-means clustering), mapy, plyny apod.

Toto lze provádět na mnohem větší množině dat, pro které nepotřebujeme předepsaný výstup

Často se ovšem používá i jednoduché rovnoměrné rozmístění středů (pokud víme, že data jsou více méně pravidelně rozmístěna) nebo umístění středů do pozice náhodně vybraných vstupů.

Nastavení šířek: Aktivační funkce by neměly být ani příliš strmé ani příliš ploché. Údajně dobrou heuristikou je jednotná šířka $\sigma = D_{\max}/\sqrt{2m}$ kde $D_{\max}$ je maximální (Euklidovská) vzdálenost středů a m je počet skrytých neuronů.

RBF - váhy

Pro fixní hodnoty parametrů skrytých neuronů (tj. fixní středy a šířky) se jedná o ADALINE. Můžeme proto použít příslušné metody k minimalizaci chyby E.

Lze také nalézt analytické řešení: Pokud fixujeme všechny středy a šířky, chyba E je funkcí $\vec{w}$ (píšeme $E(\vec{w})$). Gradient funkce $E(\vec{w})$ je roven

$$(\vec{w} \cdot \Phi - \vec{d}) \cdot \Phi^T$$

kde $\vec{d} = (d_1, \ldots, d_p)$, $\vec{w} = (w_0, \ldots, w_m)$ a Φ je matice $(m+1) \times p$ jejíž i-tý sloupec obsahuje hodnoty skrytých neuronů pro i-tý vstup, tj. pro každé $j = 1, \ldots, m$ a $i = 1, \ldots, p$ máme

$$\Phi_{ji} = \phi_j(\vec{x}_i) = \exp\left(-\frac{\left\|\vec{x}_i - \vec{c}_j\right\|}{2\sigma_j^2}\right)$$

Vektor $\vec{w}^T = \vec{d} \cdot \Phi^+$ kde $\Phi^+ = \Phi^T \cdot (\Phi \cdot \Phi^T)^{-1}$ potom řeší $(\vec{w} \cdot \Phi - \vec{d}) \cdot \Phi^T = 0$ a tedy minimalizuje $E(\vec{w})$.

Sítě typu RBF - gradient

Učení lze také provádět pomocí standardního gradientního sestupu. Gradient lze snadno spočítat přímým derivováním:

$$\frac{\delta E}{\delta \sigma_j} = \sum_{k=0}^{p} (y(\vec{x}) - d_k) \cdot \phi_k$$

$$\frac{\delta E}{\delta \sigma_j} = \sum_{k=0}^{p} (y(\vec{x}) - d_k) \cdot w_j \cdot \exp\left(-\frac{\left\| \vec{x}_k - \vec{c}_j \right\|^2}{2\sigma_j^2} \right) \cdot \frac{\left\| \vec{x}_k - \vec{c}_j \right\|^2}{\sigma_j^3}$$

$$\frac{\delta E}{\delta c_{ji}} = \sum_{k=0}^{p} (y(\vec{x}) - d_k) \cdot w_j \cdot \exp\left(-\frac{\left\| \vec{x}_k - \vec{c}_j \right\|^2}{2\sigma_j^2} \right) \cdot \frac{(x_{ki} - c_{ji})^2}{\sigma_j^2}$$

Rychlost tohoto učení je ovšem srovnatelná s rychlostí zpětné propagace pro standardní (sigmoidální) sítě.

RBF sítě - regularizace

Naučená síť by měla dobře generalizovat. Neměla by příliš „opisovat“ tréninkové vzory (tj. neměla by být přetrénovaná).

Intuice: Funkce sítě, která příliš opisuje vzory se hodně kroutí - omezíme kroucení.

Definujeme novou chybovou fci

$$E' = E + \gamma \frac{1}{2} \sum_{k=1}^{p} \sum_{i=1}^{n} \left(\frac{\delta y}{\delta x_i^2}(\vec{x}_k) \right)^2$$

Zde $\gamma > 0$ je míra vlivu regularizace.

Váhy, které minimalizují $E'(\vec{w})$, jsou řešením: $\vec{w} \cdot M = \vec{d} \cdot \Phi^T$ kde

$$M_{ij} = \sum_{k=1}^{p} \left(\phi_i(\vec{x}_k)\phi_j(\vec{x}_k) + \gamma \sum_{\ell=1}^{n} \left(\frac{\delta \phi_i}{\delta x_\ell^2}(\vec{x}_k) \frac{\delta \phi_j}{\delta x_\ell^2}(\vec{x}_k) \right) \right)$$

(Pozn. pro $\gamma = 0$ dostaneme $M = \Phi \cdot \Phi^T$, tj. minimalizaci $E(\vec{w})$)

RBF sítě - výhody

Srovnejme s vícevrstvou sítí se sigmoidálními jednotkami (dále budu značit MP (multi-layer perceptron))

- Teoreticky lze aproximovat libovolnou spojitou fci (stejně jako MP)
- Dvoufázové učení: jednotlivé fáze jsou řádově rychlejší než učení MP (tj. zpětná propagace)
- RBF sítě jsou vhodné pro klasifikaci (více než MP) díky jejich lokálnímu charakteru. Vektory, které jsou daleko od tréninkových vzorů, dostanou malou hodnotu (toto nemusí platit pro MP).
- jsou vhodné pro aplikace s měnícími se daty (rychlé učení umožňuje on-line adaptaci na nová data - téměř nemožné s MP)
- snadná regularizace (lineární)

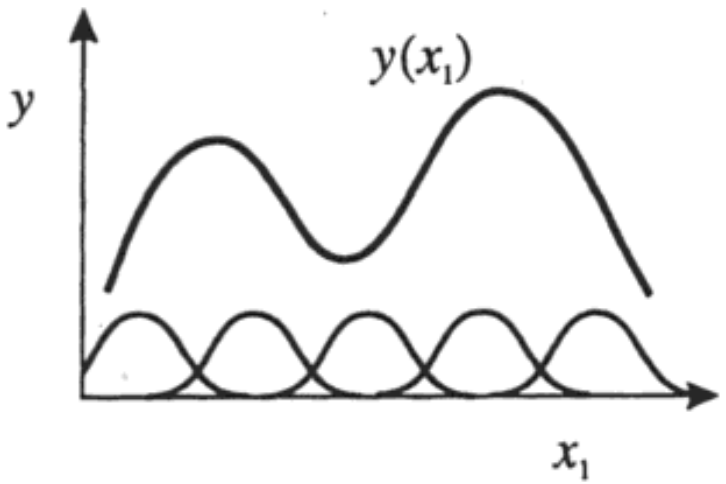
RBF sítě - nevýhody

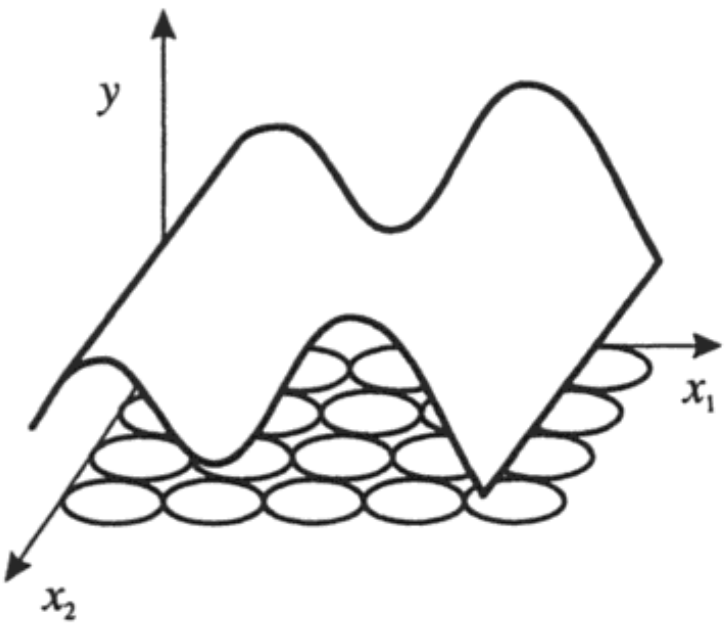
Opět srovnejme s MP.

- Teoretické výsledky o aproximaci nefungují v praxi (odhady na potřebný počet neuronů jsou nadsazené podobně jako u MP)
- Dvoufázové učení může být deformované pokud požadované funkční hodnoty nerespektují rozmístění vstupů (např. pokud je požadovaná funkce konstantní v oblasti s mnoha vzory, ale velmi variabilní v oblasti s málo vzory)
- RBF potřebují mnohem více dat i neuronů pro stejnou přesnost jako MP.
  - MP aproximují funkci globálně: každý vzor adaptuje většinu neuronů a informace je tedy distribuována po síti
  - RBF aproximují lokálně: pouze několik málo neuronů reaguje na daný vstup

  Z toho také plynou lepší extrapolační vlastnosti MP.
- RBF sítě mají větší problém s prokletím dimenzionality (exponenciální nárust jednotek s počtem dimenzí). Nejsou schopny odhalit nízkou variabilitu předepsané funkce v daném směru.

RBF sítě - nevýhody





kvadratický nárust počtu neuronů;
RBF neumí poznat, že je funkce (v podstatě) konstantní v x_2 (MP to umí)