Vektorová kvantizace

Obecná formulace problému:

- Dána hustota pravděpodobnosti $p(\vec{x})$ na vstupních vektorech $\vec{x} \in \mathbb{R}^n$.

  Tj. předpokládáme, že „něco“ náhodně generuje vstupní vektory podle rozložení s hustotou $p(\vec{x})$.
- Cílem je aproximovat hustotu $p(\vec{x})$ pomocí konečně mnoha **reprezentantů** $\vec{w}_i \in \mathbb{R}^n$ kde $i = 1, \ldots, h$.

  Velmi zhruba: v oblasti s větší hustotou by mělo být více reprezentantů, v oblastech s menší hustotou méně.
- Formálněji: danému vstupnímu vektoru $\vec{x}$ přiřadíme reprezentanta $\vec{w}_c$, který je mu nejblíže:

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

  a poté minimalizujeme chybu

  $$E = \int \left\| \vec{x} - \vec{w}_c \right\|^2 p(\vec{x}) d\vec{x}$$

  Pozor! Zde c závisí na $\vec{x}$.

Vektorová kvantizace

V praxi často máme často k dispozici tréninkovou množinu

$$\mathcal{T} = \{\vec{x}_j \in \mathbb{R}^n \mid j = 1, \ldots, \ell\}$$

z níž jsou vzory rovnoměrně vytahovány. Chyba potom odpovídá

$$E = \frac{1}{\ell} \sum_{j=1}^{\ell} \left\| \vec{x}_j - \vec{w}_c \right\|^2$$

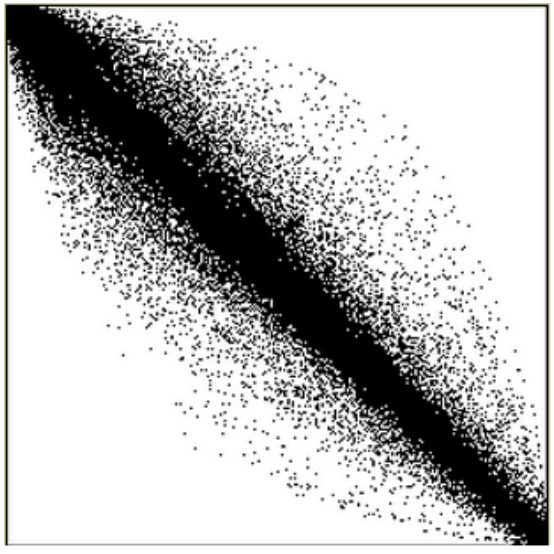
Pokud byla množina $\mathcal{T}$ volena náhodně (dle rozložení $p(\vec{x})$) a ℓ je dost velké, pak

$$\frac{1}{\ell} \sum_{j=1}^{\ell} \left\| \vec{x}_j - \vec{w}_c \right\|^2 \approx \int \left\| \vec{x} - \vec{w}_c \right\|^2 p(\vec{x}) d\vec{x}$$

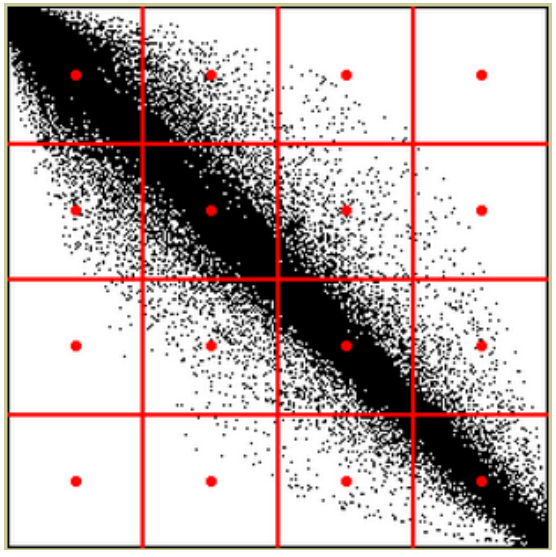
Příklad - komprese obrázků

- každý pixel má 256 stupňů šedi.
- každá dvojice sousedních pixelů je dvourozměrným vektorem z $\{0,\ldots,255\} \times \{0,\ldots,255\}$
- komprese najde malou množinu reprezentantů, kteří budou kódovat stupně šedi *dvojic* pixelů (pro danou dvojici vezmeme nejbližšího reprezentanta)
- obrázek se potom zakóduje jednoduchým průchodem při němž se dvojice pixelů nahradí reprezentanty

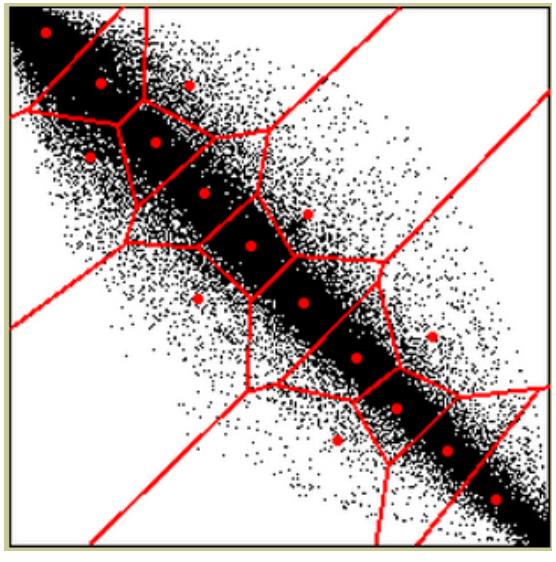
Příklad - komprese obrázků



distribuce dvojic



naivní kvantizace



chytřejší kvantizace

Lloydův algoritmus

Přepodkládáme konečnou tréninkovou množinu:
$\{\vec{x}_j \mid \vec{x}_j \in \mathbb{R}^n \mid j = 1, \ldots, \ell\}$

Algoritmus postupně posouvá reprezentanty k tréninkovým vzorům.
V kroku t vypočte $\vec{w}_1^{(t)}, \ldots, \vec{w}_h^{(t)}$ takto:

- pro každé $k = 1, \ldots, h$ spočítej množinu $\mathcal{T}_k$ vektorů z $\mathcal{T}$, které jsou nejblíže $\vec{w}_k^{(t-1)}$:

$$\mathcal{T}_k = \left\{ \vec{x}_j \in \mathcal{T} \quad | \quad k = \arg \min_{i=1,\ldots,h} \left\{ \left\| \vec{x}_j - \vec{w}_i^{(t-1)} \right\| \right\} \right\}$$

- spočítej $\vec{w}_k^{(t)}$ jako těžiště množiny $\mathcal{T}_k$:

$$\vec{w}_k^{(t)} = \frac{1}{|\mathcal{T}_k|} \sum_{\vec{x} \in \mathcal{T}_k} \vec{x}$$

Výpočet ukončíme např. ve chvíli, kdy je chyba E pro aktuální reprezentanty dostatečně malá.

Kohonenovo učení

Nevýhoda Lloydova algoritmu: není online! (mimo jiné ...)

Následující Kohonenův algoritmus je online (připouští obě verze vstupů: náhodně generované i dané tréninkovou množinou).

V kroku t po předložení vstupu $\vec{x}_t$ vypočteme $\vec{w}_k^{(t)}$ takto:

Jestliže je $\vec{w}_k^{(t-1)}$ nejblíže $\vec{x}_t$, tj. $k = \arg\min_i \left\| \vec{x}_t - \vec{w}_i^{(t-1)} \right\|$ **pak**

$$\vec{w}_k^{(t)} = \vec{w}_k^{(t-1)} + \theta \cdot (\vec{x}_t - \vec{w}_k^{(t-1)})$$

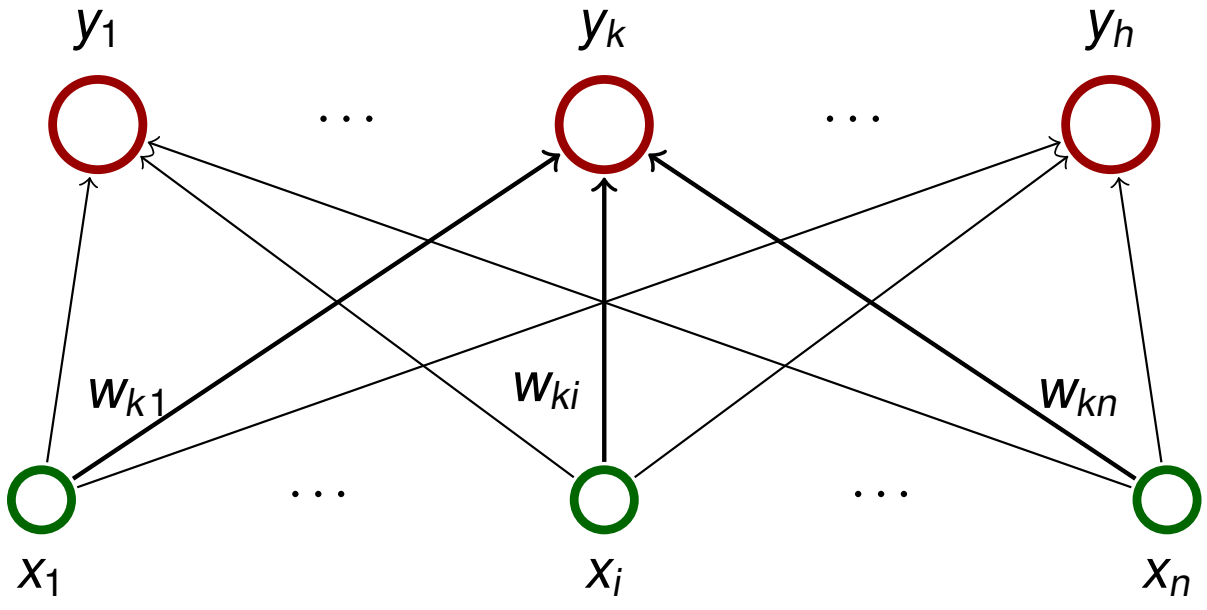
jinak $\vec{w}_k^{(t)} = \vec{w}_k^{(t-1)}$

Parametr $0 < \theta \leq 1$ určuje míru posunu reprezentanta ve směru ke vstupnímu vzoru. Na začátku učení se obvykle volí blízko 1 a postupně se zmenšuje.

Tento algoritmus lze snadno formulovat v jazyce neuronových sítí ...

Kohonenovo učení - neuronová síť

Organizační dynamika: Jednovrstvá síť



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

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

Kohonenovo učení - neuronová síť

Adaptivní dynamika (online algoritmus, Kohonenovo učení):

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

Jestliže $k = \arg\min_{i=1,\ldots,h} \left\| \vec{x}_t - \vec{w}_i^{(t-1)} \right\|$ **pak**

$$\vec{w}_k^{(t)} = \vec{w}_k^{(t-1)} + \theta(t) \cdot (\vec{x} - \vec{w}_k^{(t-1)})$$

jinak $\vec{w}_k^{(t)} = \vec{w}_k^{(t-1)}$

Parametr $0 < \theta \leq 1$ určuje míru změny vah ve směru ke vstupnímu vzoru. Na začátku učení se obvykle volí blízko 1 a postupně se zmenšuje.

Efektivita Kohonenova učení

- Funguje dobře pokud většina vstupů zhruba rovnoměrně pokrývá jednu konvexní oblast.
- V případě dvou a více oddělených shluků nemusí hustota reprezentantů odpovídat hustotě rozložení:
  - Př. dvě oddělené oblasti se stejnou hustotou.
  - Předp., že vzory jsou iniciálně rozmístěny v jedné oblasti.
  - Druhá potom přitáhne jednoho reprezentanta, který poté vždy vyhraje soutěž.
  - Výsledek: jedna oblast bude reprezentována jedním reprezentantem, druhá zbytkem reprezentantů (hustota reprezentantů tedy neodpovídá hustotě pravděpodobnosti).

Částečná zlepšení Kohonenova učení

Tyto nedostatky lze částečně odstranit např.

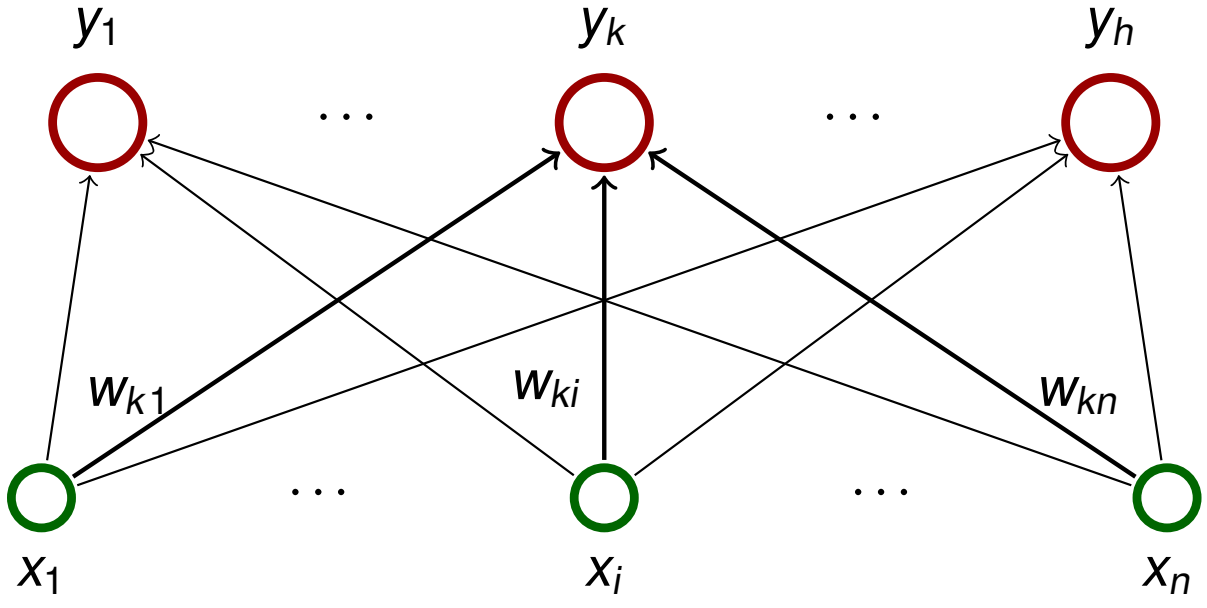
- přidáním šumu k tréninkovým vzorům - obvykle se přidávají náhodně generované vektory k tréninkovým vzorům, na počátku učení náhodné vektory převažují a postupně je jejich počet snižován
- radiální růst
  - začínáme s váhovými vektory blízko $\vec{0}$
  - všechny tréninkové vzory násobíme faktorem β, který je na počátku blízko 0 a postupně roste

    (váhy se musí postupně vzdalovat od $\vec{0}$)
- lokální paměť: Cílem je, aby pravděpodobnost vítězství každého z h neuronů (reprezentantů) byla zhruba $1/h$.

  Každý neuron si udržuje přehled o tom kolikrát vyhrál v soutěži. Pokud je četnost výher příliš vysoká, na nějakou dobu vypadne ze soutěže.

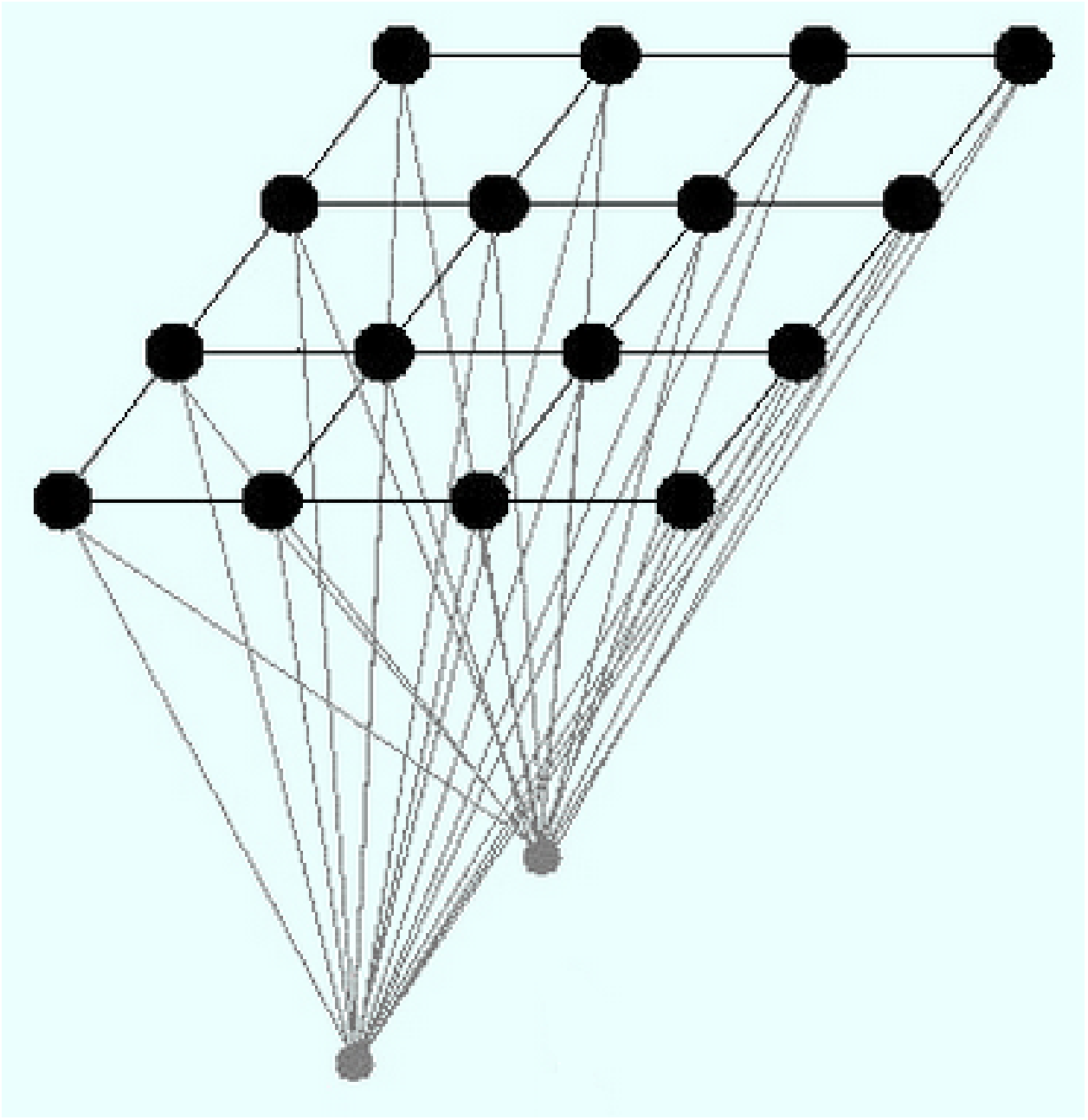
Kohonenova mapa

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 - ilustrace



Kohonenova mapa - bio odbočka

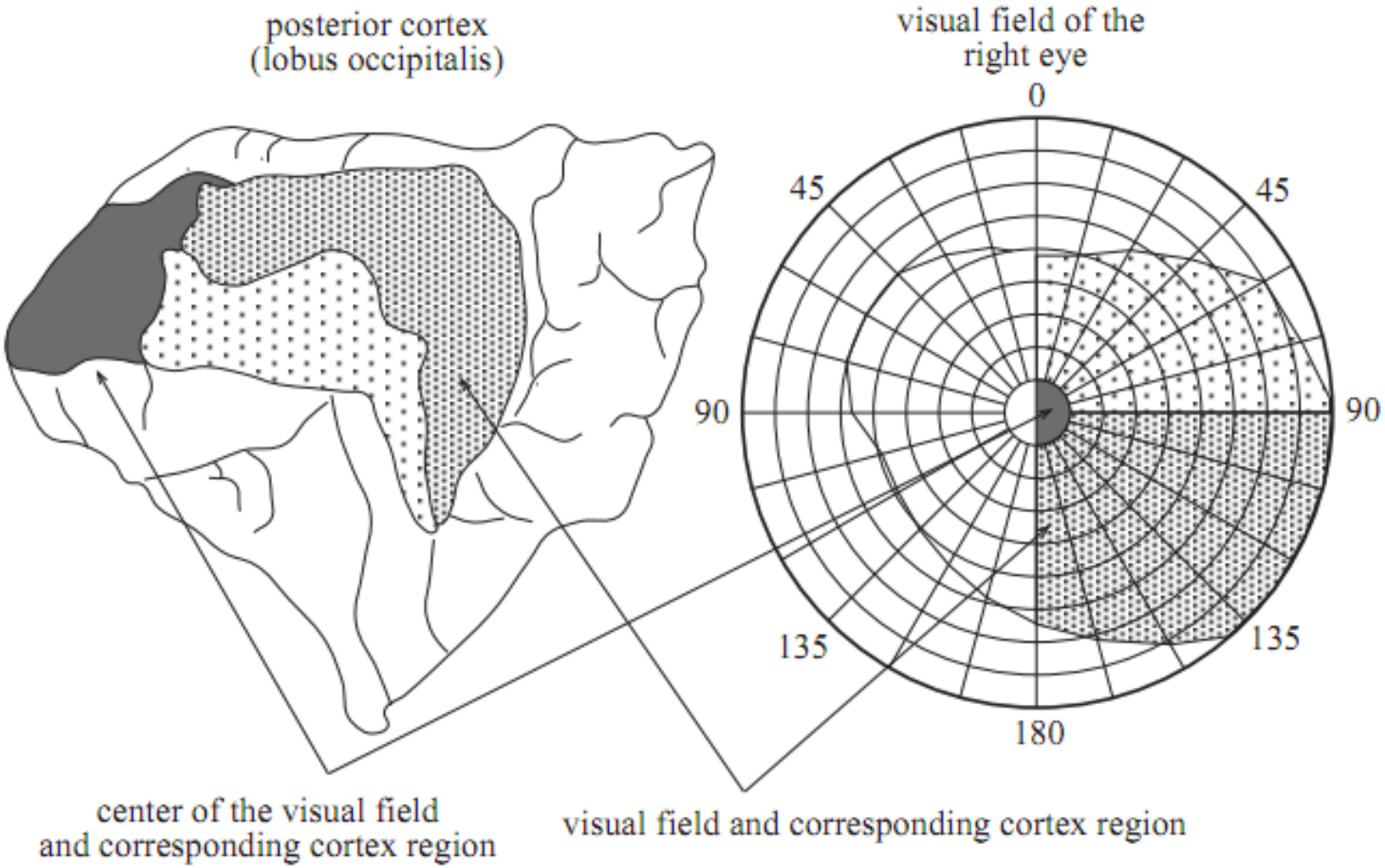


Fig. 15.2. Mapping of the visual field on the cortex

Zdroj obrázku: Neural Networks - A Systematic Introduction, Raul Rojas, Springer, 1996

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

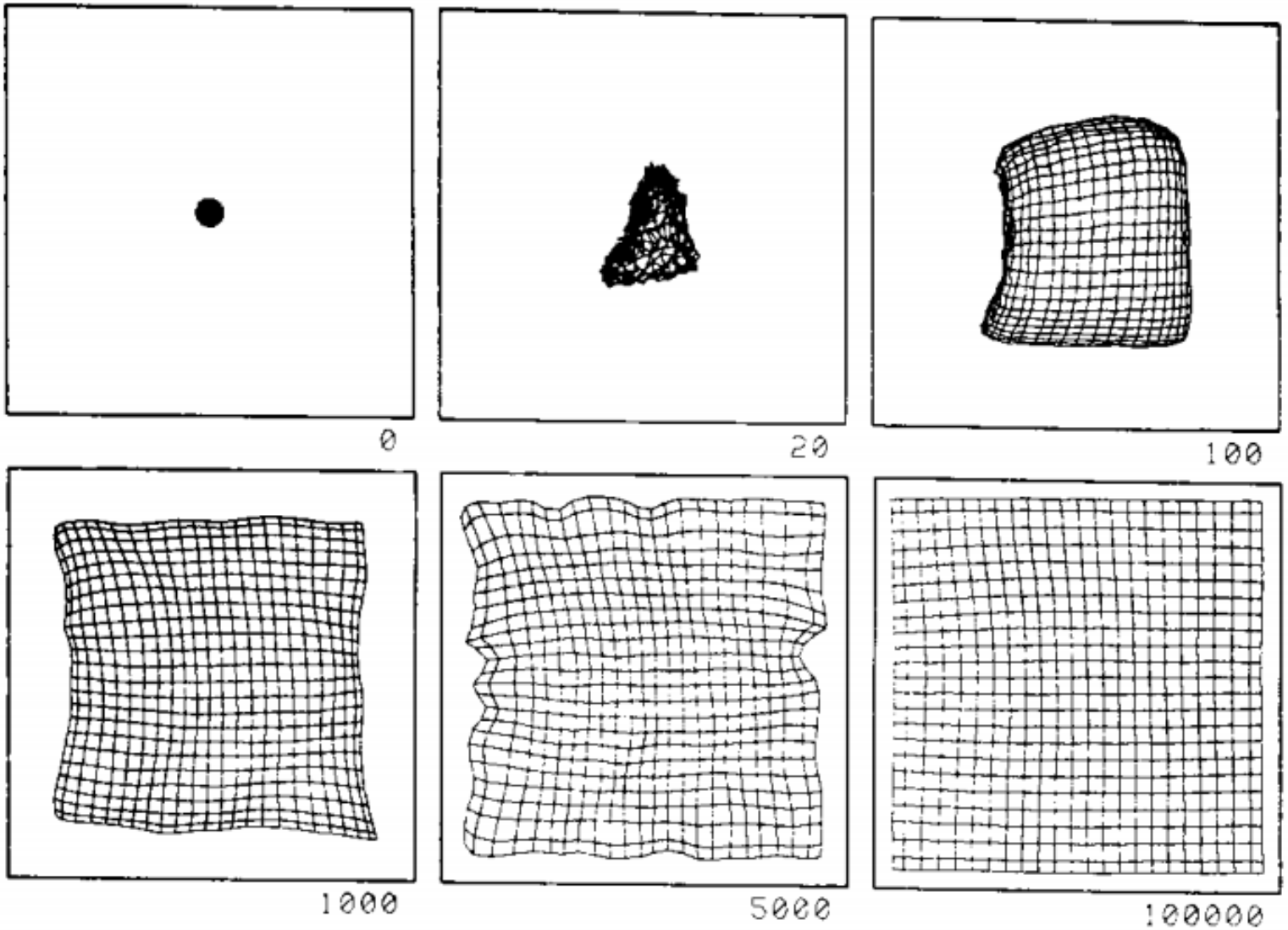
$$\Theta(c,k) = \begin{cases} \theta & k \in N_s(c) \\ 0 & \text{jinak} \end{cases}$$

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).

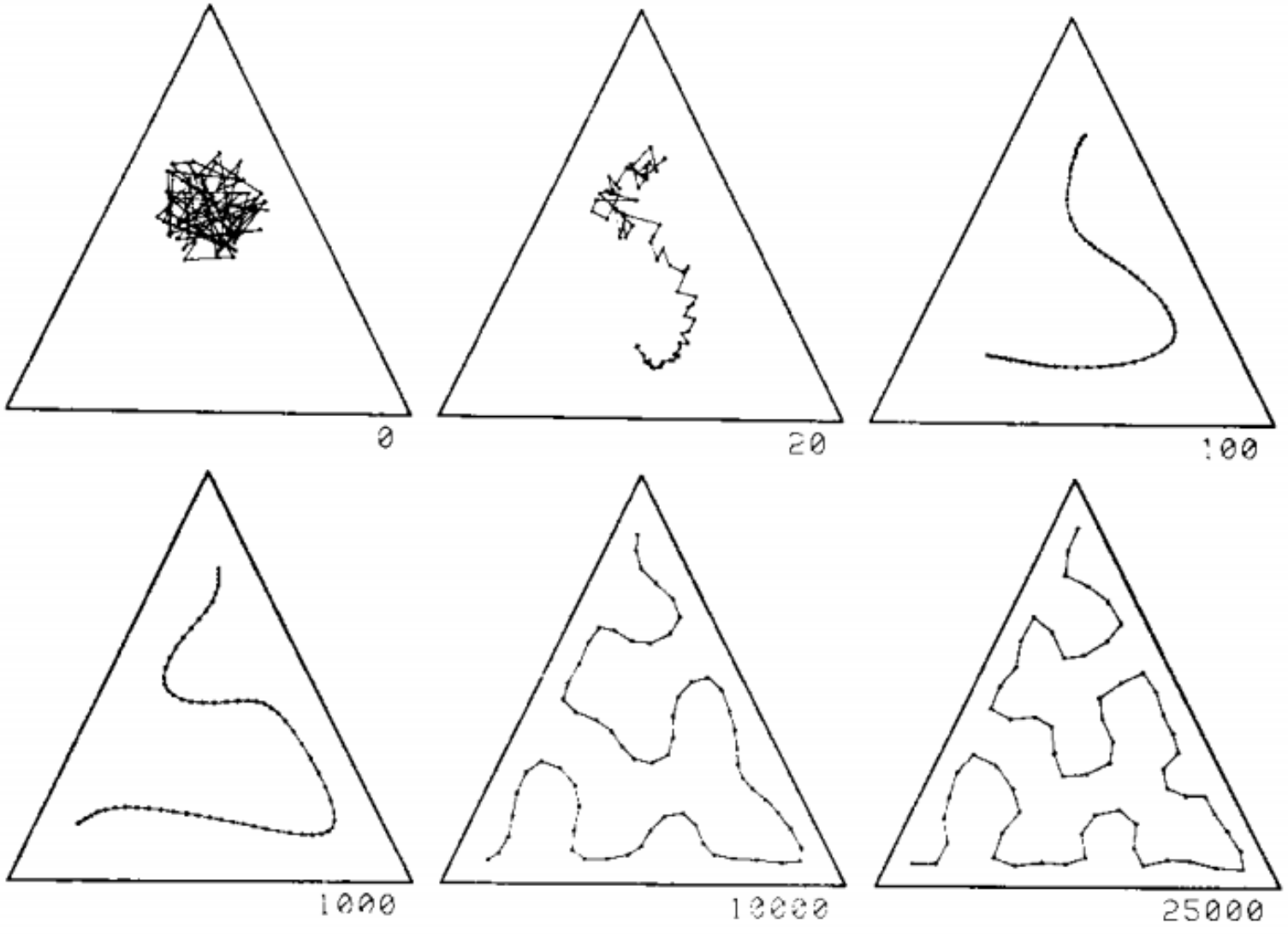
Příklad 1



Vstupy jsou uniformně rozloženy ve čtverci.

Zdroj obrázku: Neural Networks - A Systematic Introduction, Raul Rojas, Springer, 1996

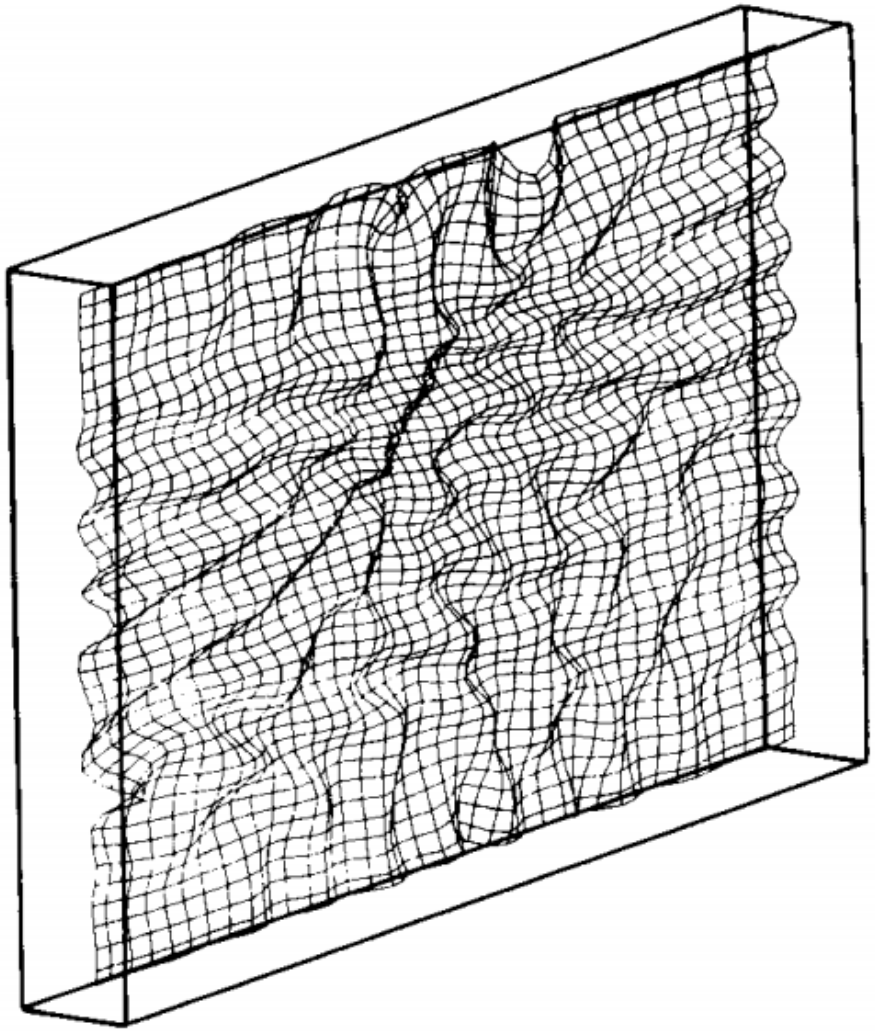
Příklad 2



Vstupy jsou uniformně rozloženy v trojúhelníku.

Zdroj obrázku: Neural Networks - A Systematic Introduction, Raul Rojas, Springer, 1996

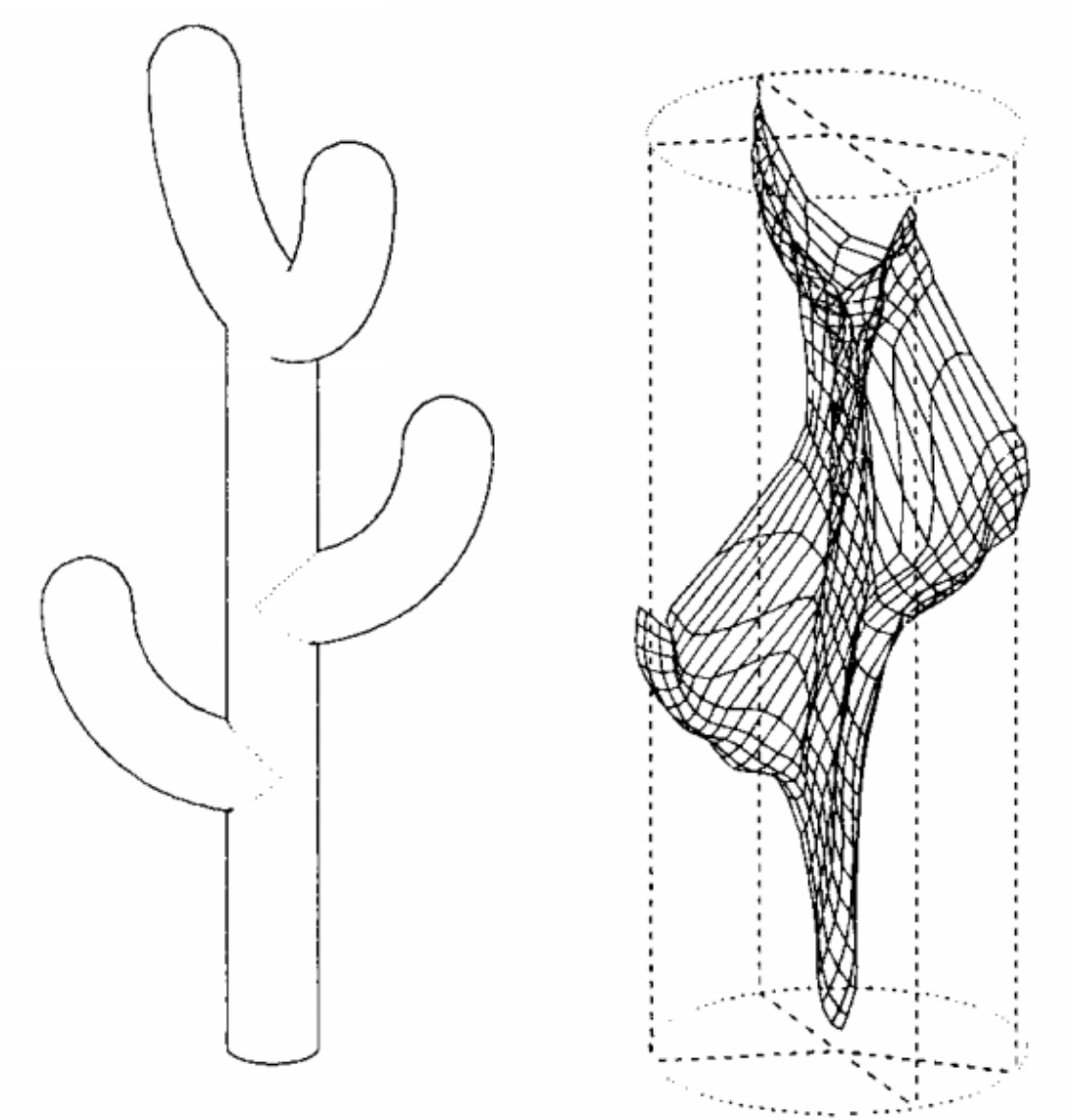
Příklad 3



Vstupy jsou uniformně rozloženy v kvádru.

Zdroj obrázku: Neural Networks - A Systematic Introduction, Raul Rojas, Springer, 1996

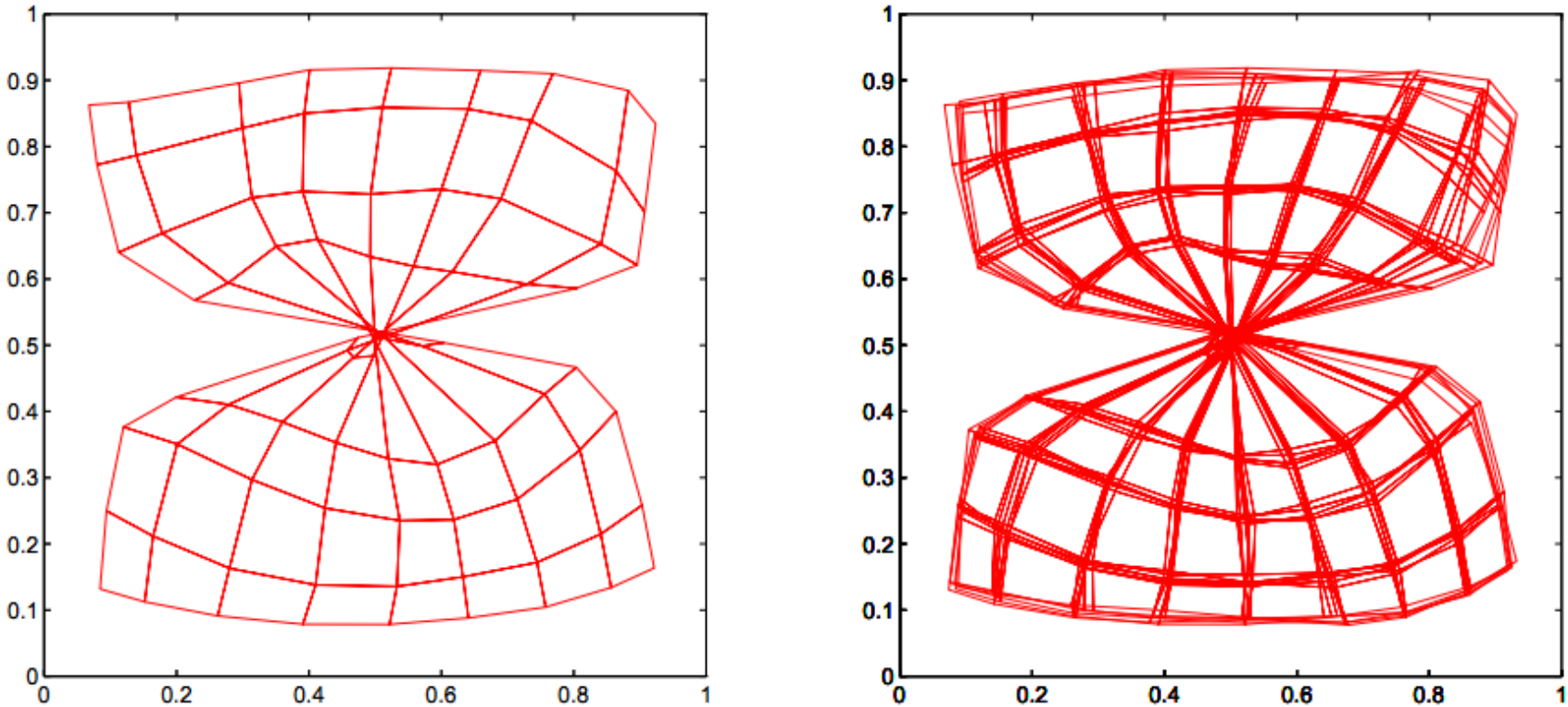
Příklad 4



Vstupy jsou uniformně rozloženy v kaktusu.

Zdroj obrázku: Neural Networks - A Systematic Introduction, Raul Rojas, Springer, 1996

Příklad - defekt



Překroucená síť - topologický defekt.

Zdroj obrázku: Neural Networks - A Systematic Introduction, Raul Rojas, Springer, 1996

Kohonenova mapa - parametry - pohled praktika

Iniciální váhy (prý) nejsou příliš důležité, pokud jsou různé.

Učení se obvykle sestává ze dvou fází:

hrubé učení:

- obvykle cca 1000 kroků
- rychlost učení θ by měla začít kolem 0.1 a postupně klesat k 0.01
- okolí každého neuronu (dané parametrem *s* nebo případně šířkou σ) by mělo zpočátku obsahovat většinu neuronů a postupně se zmenšovat tak, aby ke konci této fáze obsahovalo nejvýše několik neuronů

dolaďování:

- počet kroků je obvykle volen jako 500 násobek počtu neuronů v síti
- θ by se měla držet kolem 0.01 jinak hrozí topologický defekt (zmačkaná síť)
- okolí každého neuronů by mělo obsahovat jen pár dalších neuronů (nebo jen ten jeden)

Kohonenova mapa - pohled teoretika

Předpokládáme náhodně generované vstupy s hustotou pravděpodobnosti $p(\vec{x})$. Platí $\int p(\vec{x})d\vec{x} = 1$.

Definujeme hustotu bodů $m(\vec{x}) = \#(\vec{x})/d\vec{x}$ kde $\#(\vec{x})$ je počet neuronů jejichž váhové vektory leží v malém okolí $d\vec{x}$ bodu $\vec{x}$ ve *vstupním prostoru*. Platí

$$\int m(\vec{x})d\vec{x} = h$$

kde h je počet neuronů (reprezentantů).

Pokud je dimenze mapy rovna jedné (tj. mapa je řetězcem neuronů) pak po natrénování

$$m(\vec{x}) \propto p^{2/3}(\vec{x})$$

Kohonenova mapa - pohled teoretika

- Konvergenci k uspořádanému stavu se podařilo dokázat pouze pro jednorozměrnou topologii a různá rozložení vstupů (první výsledky byly pro uniformní rozložení) a fixní okolí velikosti 1 a konstantní rychlost učení

  Lze nalézt velice jednoduché protipříklady pokud tyto podmínky nejsou splněny.
- Ve vícerozměrném případě žádné obecné výsledky neexistují, konvergence do uspořádaného stavu je ovlivněna mnoha faktory:
  - iniciální rozložení neuronů
  - velikost okolí
  - rychlost učení
- Jakou dimenzi topologie zvolit? Drtivá většina aplikací je založena na jedné nebo dvoudimenzionální mapě. Experimenty prokázaly, že ideální dimenze topologie odpovídá (Haussdorfově) dimenzi dat.

LVQ - klasifikace pomocí Kohonenovy mapy

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 pomeranče. 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 P v závislosti na tom, jestli je daný kus jablko nebo pomeranč

Připouštíme možnost, že se najdou dvě jablka se stejnými mírami.

Cílem je třídit ovoce na základě hmotnosti a průměru.

LVQ - klasifikace pomocí Kohonenovy mapy

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 (viz. dále)

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

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*.

Co to je??

Bayesovský klasifikátor

Mějme pouze dvě třídy C_0 a C_1 (např. J a P).

Nechť $P(C_i)$ je pravděpodobnost, že náhodně vybraný vstup s vektorem vlastností $\vec{x}$ padne do třídy C_i.
(např. $P(C_0)$ je pravděpodobnost, že další kus na pásu je jablko)

Nechť $P(C_i \mid \vec{x})$ je pravděpodobnost, že je vstup z třídy C_i za přepodkladu, že má vlastnost $\vec{x}$.
(např. $P(C_0 \mid (a,b))$ vyjadřuje pravděpodobnost, že ovoce s hmotností a a průměrem b je jablko)

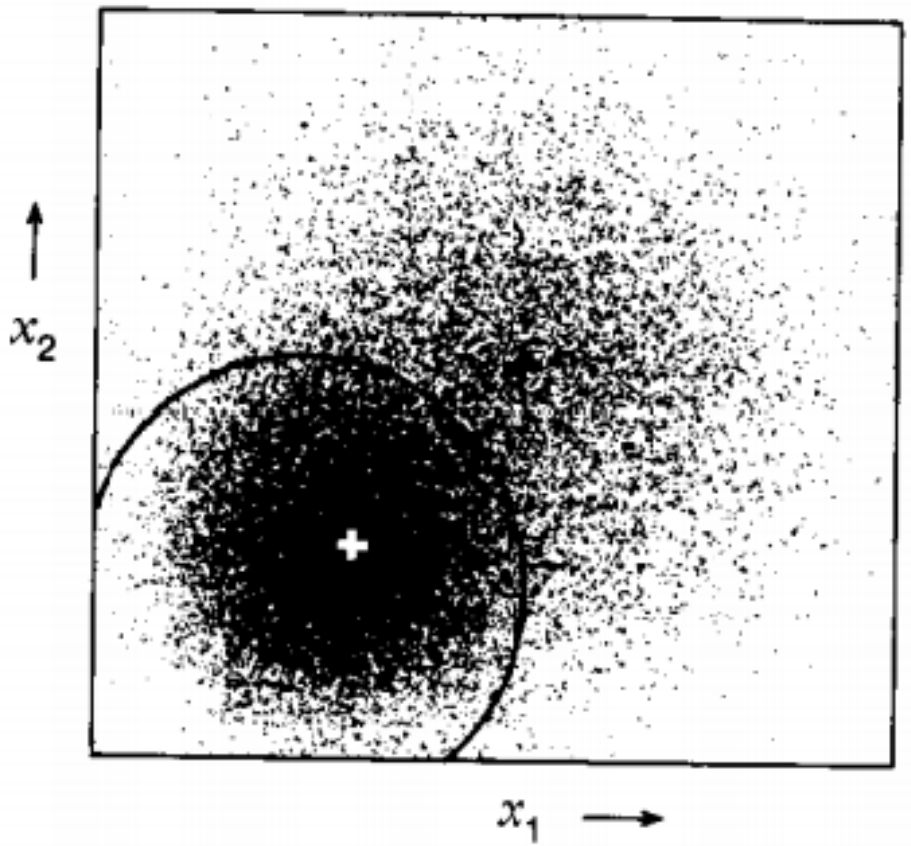
Bayesovský klasifikátor přiřadí vstupu s vektorem vlastností $\vec{x}$ třídu C_i, která splňuje $P(C_i \mid \vec{x}) \geq P(C_{1-i} \mid \vec{x})$. Označme R_0 všechna $\vec{x}$ která splňují $P(C_0 \mid \vec{x}) \geq P(C_1 \mid \vec{x})$ a $R_1 = \mathbb{R}^n \setminus R_0$.

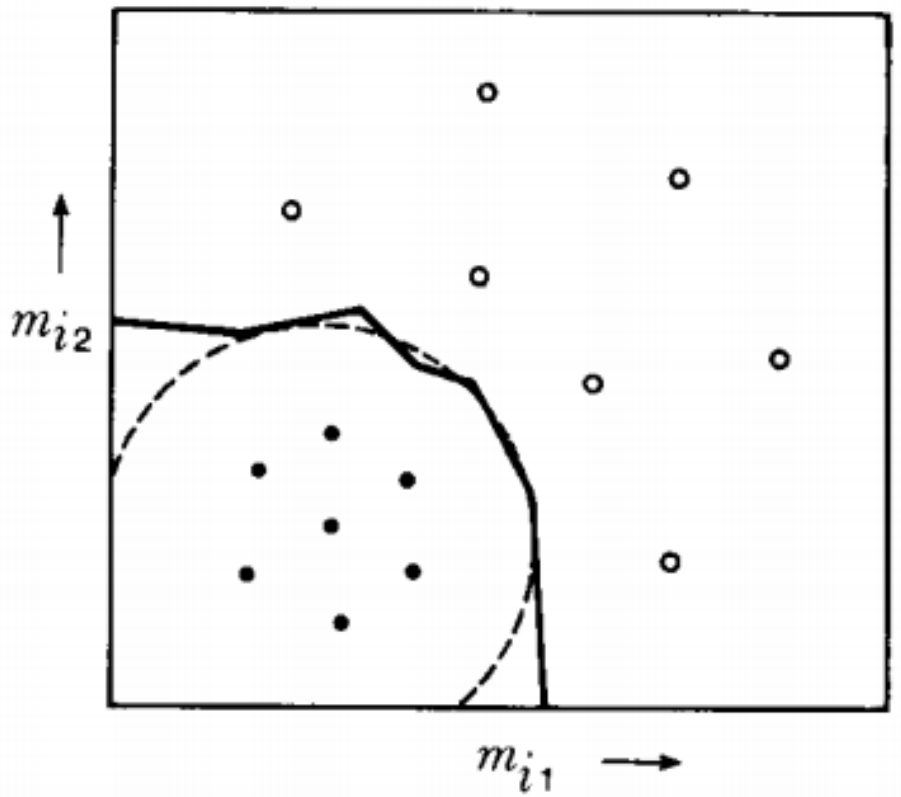
Bayesovský klasifikátor minimalizuje pravděpodobnost chyby:

$$P(\vec{x} \in R_0 \wedge C_1) + P(\vec{x} \in R_1 \wedge C_0)$$

Bayesovská hranice je hranicí mezi množinami R_0 a R_1.

Bayesovská hranice a LVQ1 - příklad





Zdroj obrázku: The Self-Organizing Map, Teuvo Kohonen, IEEE, 1990

LVQ2

Aproximaci bayesovské hranice je možné zlepšit následujícím způsobem:

Pro vzor $(\vec{x}_t, d_t)$ určíme dva nejbližší neurony, řekněme i a j s aktuálními váhami $\vec{w}_i^{(t-1)}$ a $\vec{w}_j^{(t-1)}$.

Jestliže jsou splněny následující dvě podmínky

- $\vec{x}_t$ se nachází v **okně** okolo nadroviny umístěné ve středu spojnice $\vec{w}_i^{(t-1)}$ a $\vec{w}_j^{(t-1)}$ a
- jeden z neuronů i a j klasifikuje správně a druhý špatně

pak aktualizujeme váhy takto (bez újmy na obecnosti předpokládejme, že neuron i klasifikuje správně):

$$\vec{w}_i^{(t)} = \vec{w}_i^{(t-1)} + \alpha(\vec{x}_t - \vec{w}_i^{(t-1)})$$

$$\vec{w}_j^{(t)} = \vec{w}_j^{(t-1)} - \alpha(\vec{x}_t - \vec{w}_j^{(t-1)})$$

LVQ2

Okno je obvykle definováno takto: Řekneme, že $\vec{x}_t$ se nachází v okně relativní šířky q mezi $\vec{w}_i^{(t-1)}$ a $\vec{w}_j^{(t-1)}$ pokud

$$\min\left\{\frac{d_i}{d_j}\,,\ \frac{d_j}{d_i}\right\} > \frac{1-q}{1+q}$$

kde $d_i = \left\|\vec{x}_t - \vec{w}_i^{(t-1)}\right\|$ a $d_j = \left\|\vec{x}_t - \vec{w}_j^{(t-1)}\right\|$.

Obvykle se volí q mezi 0.1 až 0.3.

LVQ2 zlepšuje pozici rozhodovací hranice tím, že ji posunuje k bayesovské hranici, ale jen do určitého počtu kroků. Později se neurony začínají vzdalovat (typicky se používá kolem 10 000 iterací).

LVQ3

LVQ3 se snaží odstranit defekt LVQ2 tím, že dobře klasifikující dvojici nejbližších neuronů posunuje směrem k vzoru (LVQ2 v takovém případě nedělá nic):

Nejprve „LVQ2 část“:

$$\vec{w}_i^{(t)} = \vec{w}_i^{(t-1)} + \alpha(\vec{x}_t - \vec{w}_i^{(t-1)})$$

$$\vec{w}_j^{(t)} = \vec{w}_j^{(t-1)} - \alpha(\vec{x}_t - \vec{w}_j^{(t-1)})$$

pokud i a j je pár výstupních neuronů nejblíže k $\vec{x}_t$, $v_i = d_t$, $v_j \neq d_t$ a $\vec{x}_t$ patří do okna mezi $\vec{w}_i^{(t)}$ a $\vec{w}_j^{(t)}$.

Navíc:

$$\vec{w}_r^{(t)} = \vec{w}_r^{(t-1)} + \varepsilon\alpha(\vec{x}_t - \vec{w}_r^{(t-1)})$$

pokud $r \in \{i, j\}$ a $v_i = v_j = d_t$.

ε se doporučuje nastavit dle šířky okna mezi 0.1 až 0.5 a nechat konstantní.