Neuronové sítě a výpočetní inteligence

Úvod do předmětu

Miroslav Skrbek ©2010

LS2011/12

Umělá inteligence

Umělá inteligence (UI) je obor informatiky zabývající se tvorbou strojů vykazujících známky inteligentního chování. Definice pojmu „inteligentní chování“ je stále předmětem diskuse, nejčastěji se jako etalon inteligence užívá lidský rozum.

Převzato z: wikipedia http://cs.wikipedia.org/wiki/Um%C4%9Bl%C3%A1_inteligence

Turingův test (Alan Turing 1950): test, ve kterém za oponou je umístěn stroj a člověk. Jiný člověk (před oponou) s oběma komunikuje (verbálně). Pokud nerozenáme stroj od člověka, pak o stroji můžeme hovořit jako o inteligentním stroji.

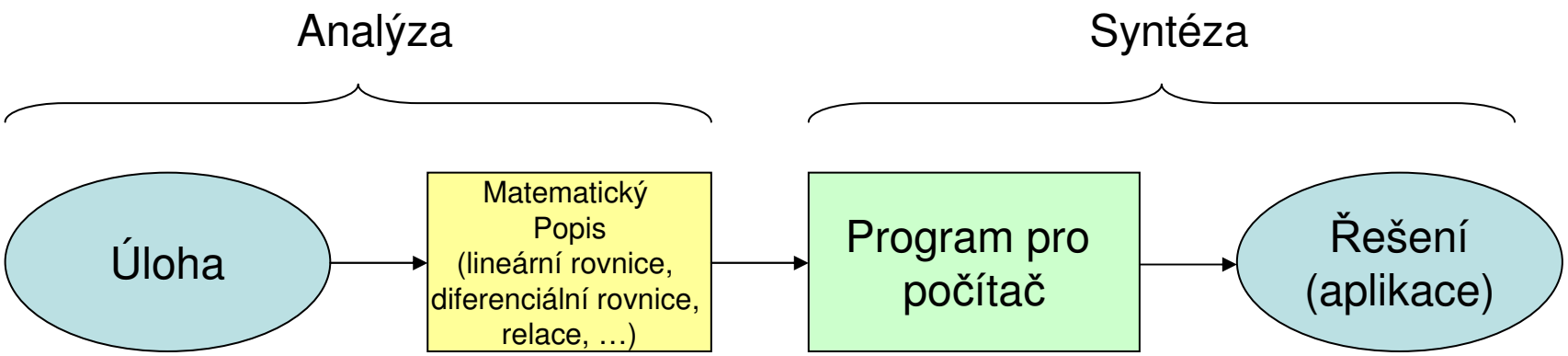
Klasická umělá inteligence je založena na logice, symbolickém počítání, odvozování, řešení problémů – bohužel mnohdy tyto metody vedou k vysoké výpočetní složitosti (prozkoumávání obrovských stavových prostorů).
Programovací jazyky pro AI: prolog, lisp,

Příklad: šachový stroj, expertní systémy

Výpočetní inteligence

- Je to část umělé inteligence, která (na rozdíl od tradiční umělé inteligence založené na symbolickém počítání) využívá především heuristických algoritmů pro řešení problémů
- Výpočetní inteligence zahrnuje
  - Neuronové sítě (neural networks)
  - Fuzzy systémy (fuzzy systems)
  - Evoluční výpočty (evolutionary computation).

Řešení reálných úloh

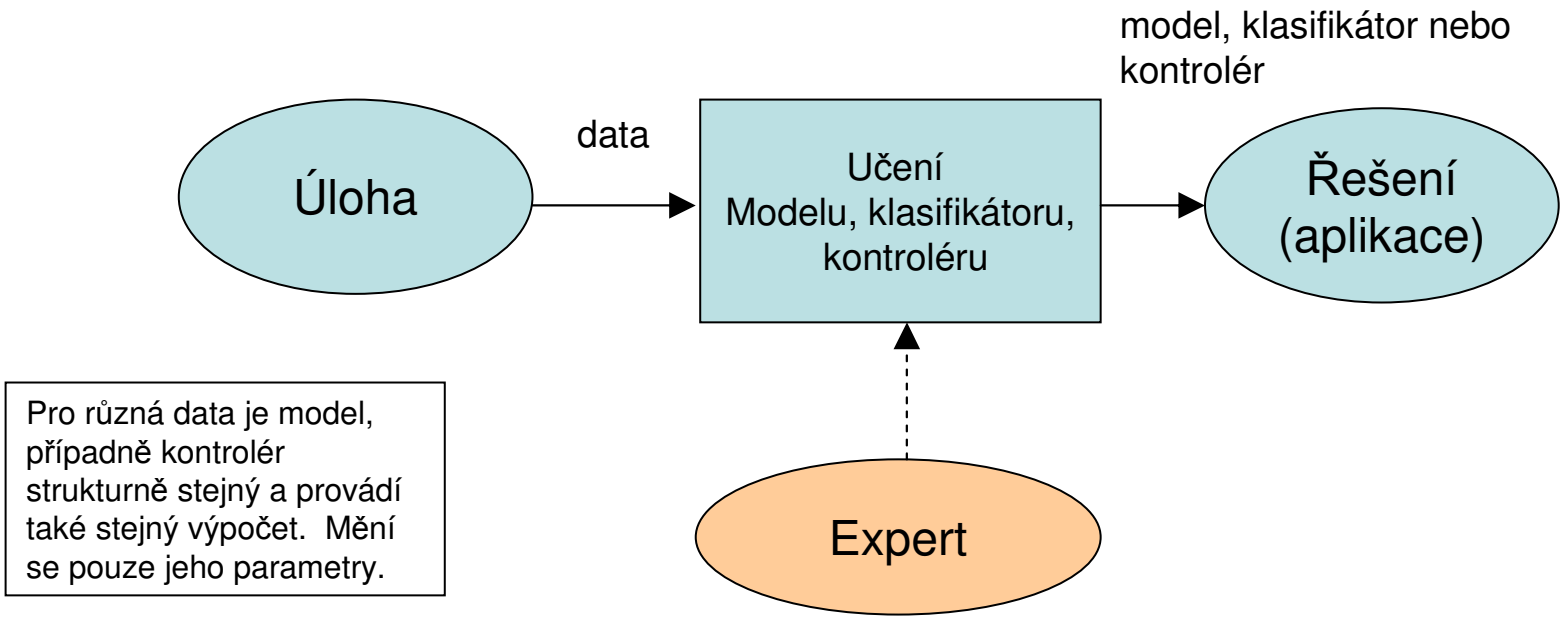


Problém: pro řadu úloh nelze nalézt matematický popis

Příklady: rozpoznej obličej, nakresli umělecký obraz, rozpoznej, co řekl, řiď bezpečně automobil, …

Pokud přeci jen nějaký matematický popis existuje, nemusí být možné získat řešení z důvodu vysoké operační nebo paměťové složitosti algoritmu.

Jiné řešení je učit

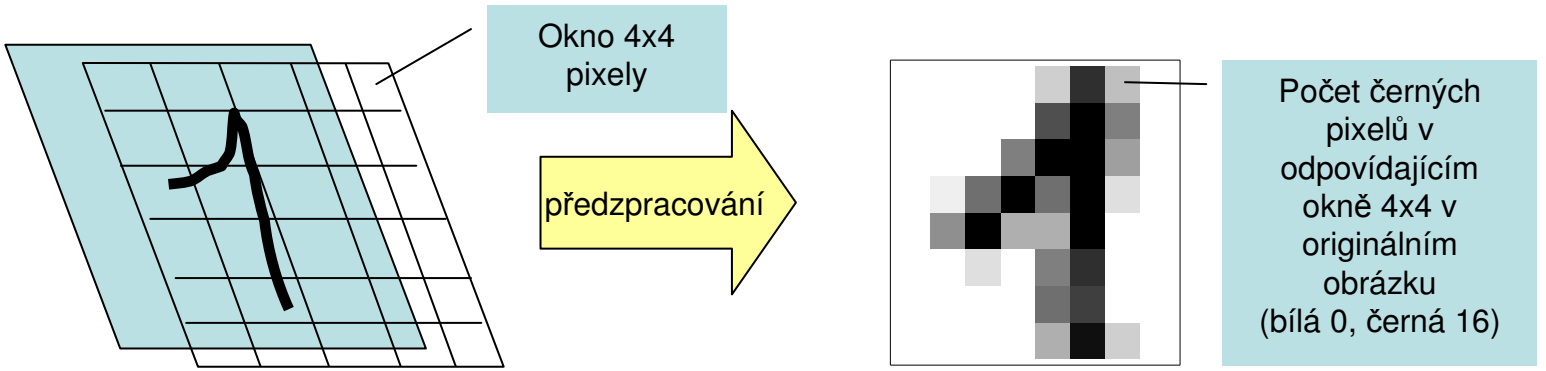


Model (modeluje daná data), kontrolér realizuje funkci řízení (na základě vstupních dat generuje řídící signály)

Příklad kontroléru: programové vybavení pro autonomního robota
Příklad modelu: programové vybavení systému, který určuje závažnost choroby na základě jejích příznaků (dat z laboratorních měření nebo vyšetření)

Příklad

Napiště program, který bude rozpoznávat ručně psané písmo
(praktická aplikace – třídění obálek na poště)



32x32 pixelů
(černobílý obr.)

{0,0,0,0,5,15,1,0,0,0,0,0,14,16,2,0,0,0,1,12,16,16,
2,0,0,1,13,16,12,16,0,0,0,4,13,2,8,13,0,0,0,0,0,0,9
,13,0,0,0,0,0,0,11,16,0,0,0,0,0,0,4,16,4,0,1}

Řešení: rozpoznávat písmena počítač prostě naučíme a nebudeme nic programovat

Zdroj dat: UCI databaze
(http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits)

E. Alpaydin, C. Kaynak, Department of Computer Engineering, Bogazici University, 80815 Istanbul Turkey, alpaydin@boun.edu.tr, July 1998

Zevšeobecnění (generalizace)

- Zevšeobecnění je vlastnost správě reagovat na neučená (ale učeným datům podobná) data
  - V autoškole se učíme řešit křižovatky (kreslené); na základě této učené znalosti jsme schopni projet reálnou křižovatku bez nehody (tedy alespoň většina z nás)
  - V jistém smyslu slova, je možné proložení bodů křivkou chápat jako jistý druh zevšeobecnění (získáme hodnoty mezi zadanými body, které se nebudou příliš lišit od skutečnosti)
    Pozn.: na tomto principu řada adaptivních systémů pracuje, jen se jedná o mnohorozměrný prostor, kde se prokládá hyperrovinami a hyperplochami.
- Učení pozbývá smyslu, pokud učený systém nemá vlastnost zevšeobecnění
  - student se naučí fakta, ale látce nerozumí (neumí znalost použít a nechápe souvislosti)

Co se obvykle učí ?
Model, klasifikátor, kontrolér, agent

Model reprezentuje určitou realitu a je schopen o této realitě vypovídat a dávat předpovědi.

(model chování osoby, počasí, letadla při letu, při nehodě, …)

Klasifikátor zařazuje data do tříd (skupin) na základě nějakého pozorování. Vstupním hodnotám přiřazuje výstupní kategorii.

(příznaky-druh nemoci, vlastnosti-zvíře (rostlina, …), obrázek-písmeno, zvuk-promluva, …)

Kontrolér (regulátor) na základě sensorických vstupů generuje akční (řídící) hodnoty. Může být kompozicí modelů a klasifikátorů.

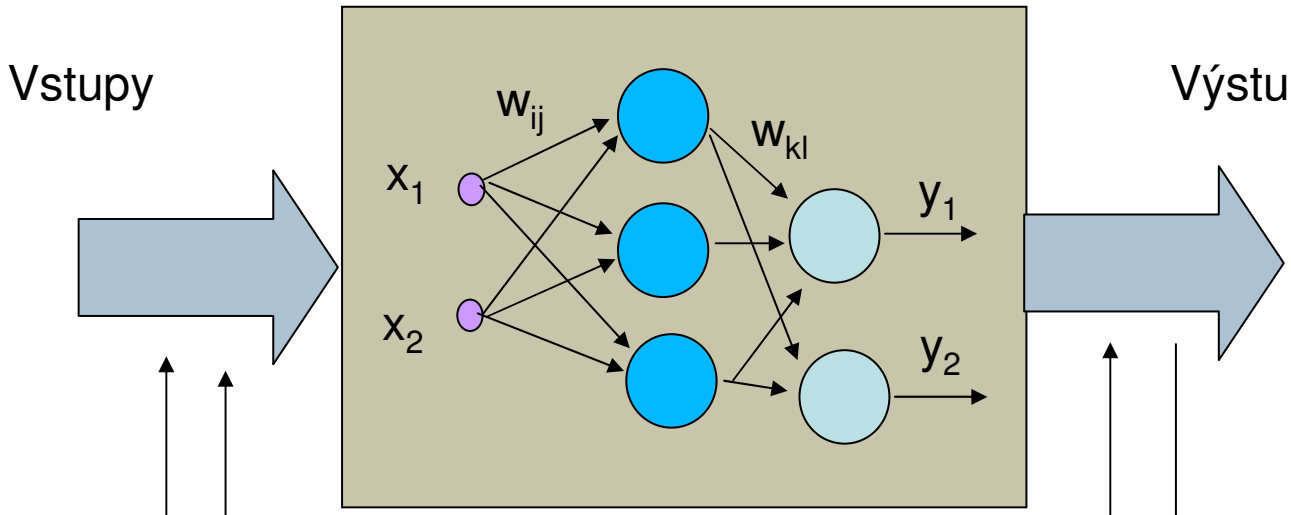
(řídící část robota, regulátor určitého procesu (výrobního, chemického apod., …)

Agent funkční celek (obvykle program nebo kombinace software/hardware), který se umí rozhodovat, hledat řešení, interagovat s okolím a kooperovat s ostatními agenty.

(nyní ve výzkumu, kooperace různých spotřebičů v inteligentní domácnosti, agenti vyhledávající neobvyklé aktivity v rozsáhlých síťových nebo ekonomických prostředích, poskytující určité služby, atp.)

Co znamená učit ?

Model/kontrolér $\vec{y} = f(\vec{x})$



Vstup		Výstup	
x1	x2	y1	y2
0,1	0,3	0,2	0,4
0,8	0,2	0,8	0,1
0,4	0,4	0,8	0,8

Učení modelu

Vstup		Výstup	
x1	x2	y1	y2
0,2	0,1	???	???
0,1	0,8	???	???
0,3	0,9	???	???

Použití (aplikace) modelu

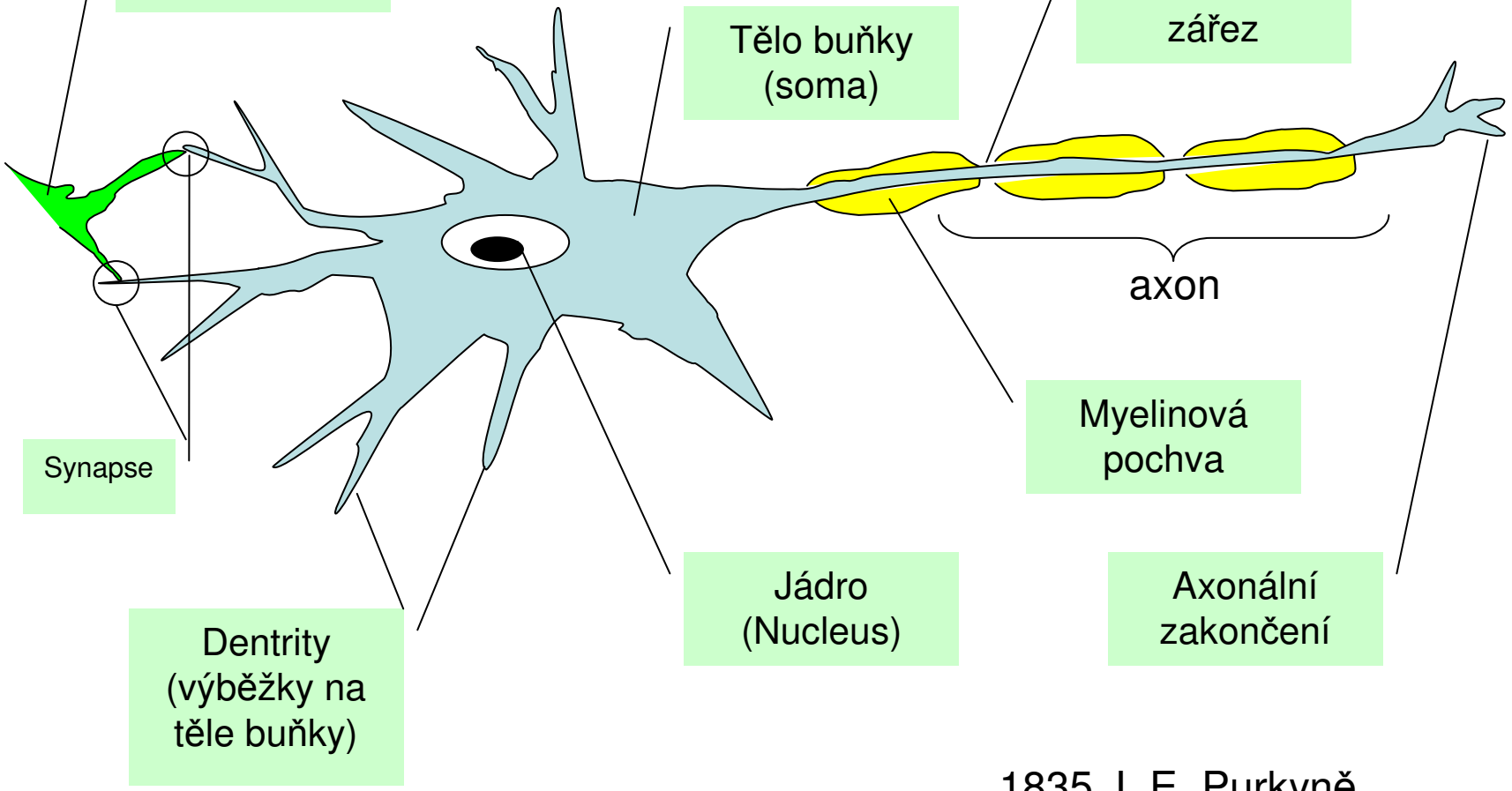
Neuronové sítě a výpočetní inteligence

Neuronové sítě

Miroslav Skrbek ©2010

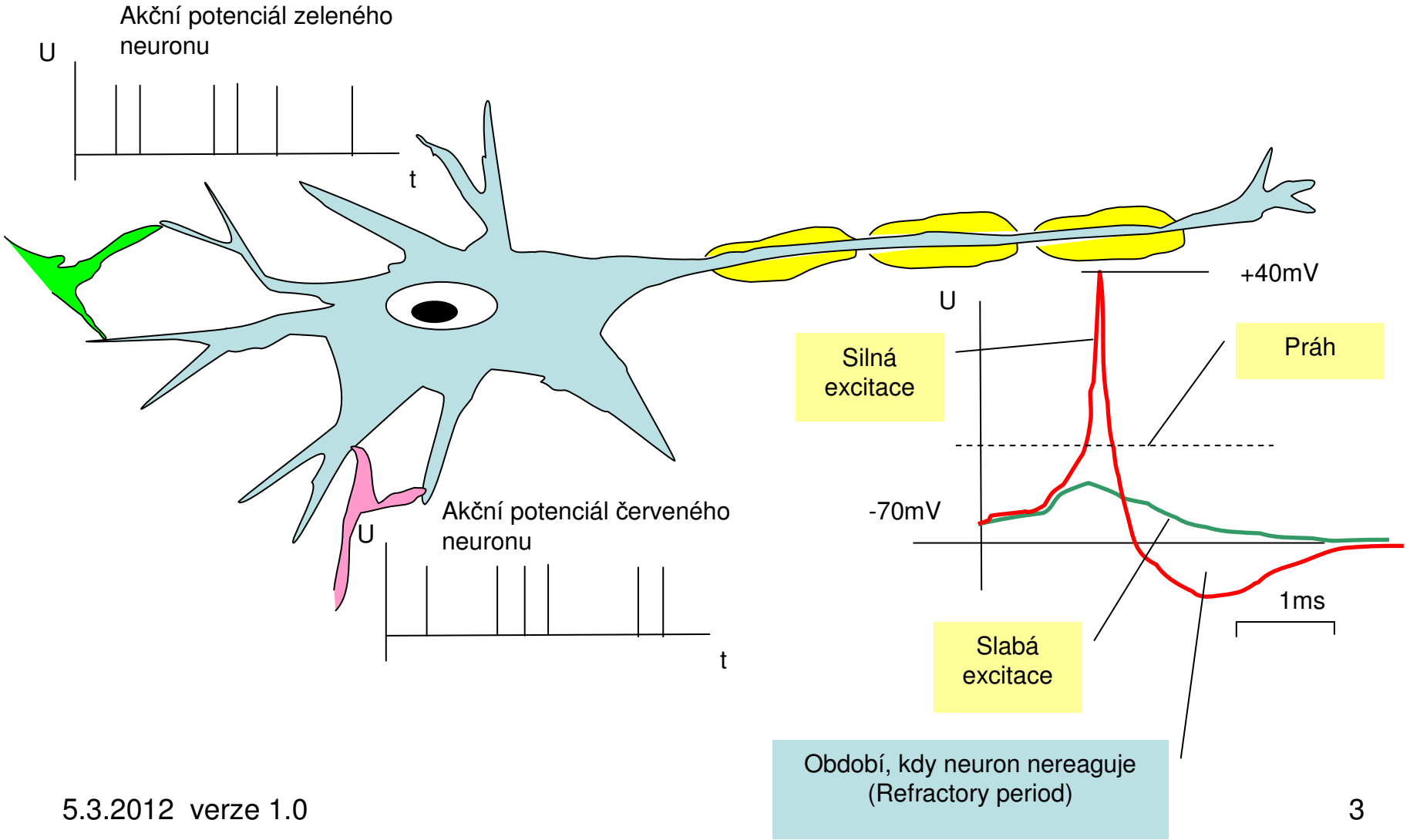
LS2011/12

Biologický neuron

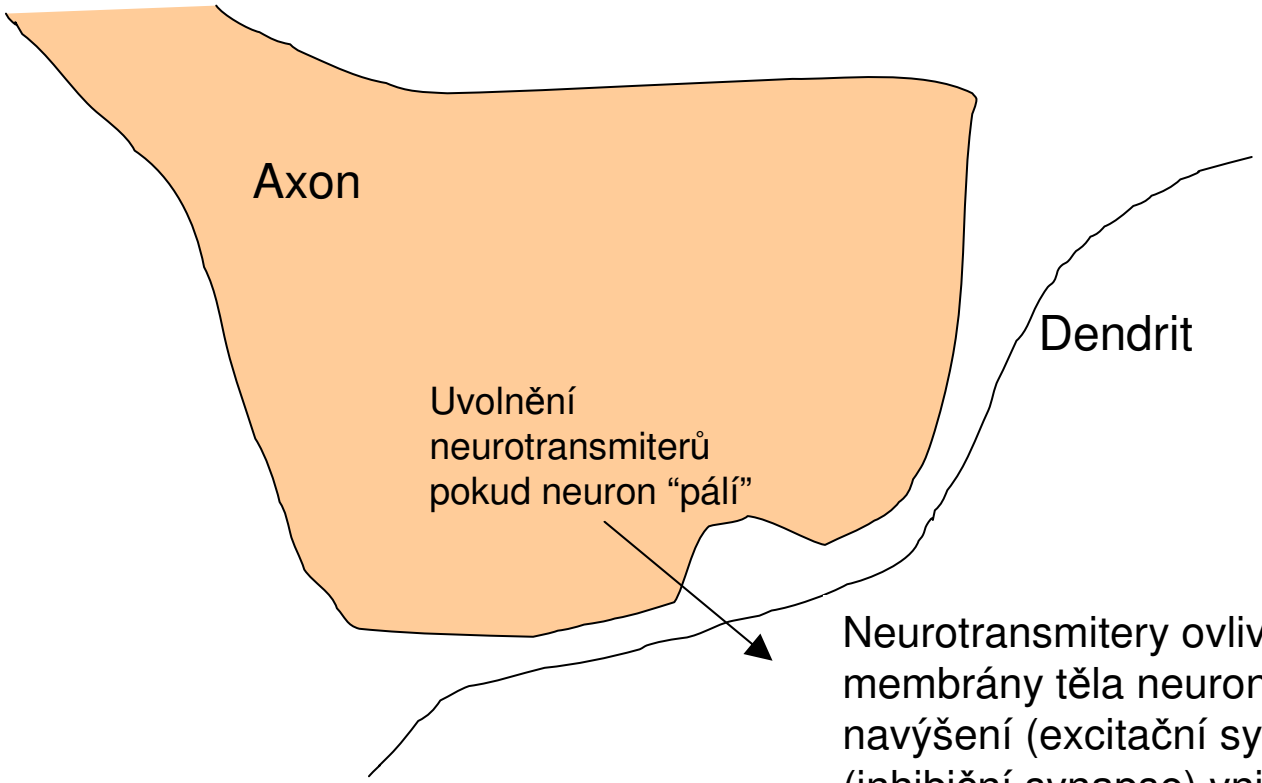


1835 J. E. Purkyně

Funkce neuronu



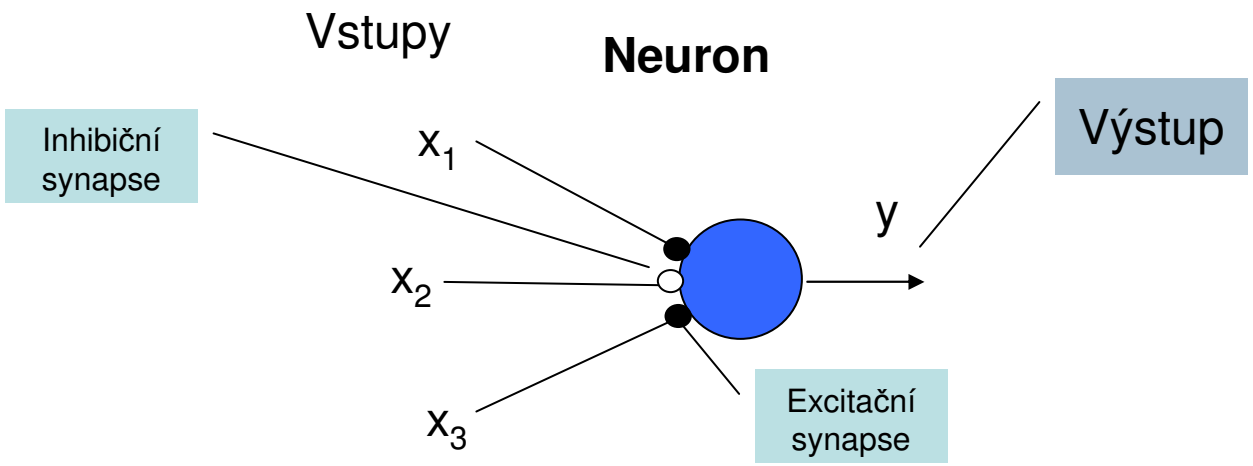
Synapse



Neurotransmitery ovlivňují vlastnosti membrány těla neuronu a vedou tak k navýšení (excitační synapse) nebo snížení (inhibiční synapse) vnitřního potenciálu neuronu.

Synapse vykazují plasticitu, tj. vlastnosti synapse se mění v čase, mluvíme o procesu adaptace nebo učení.

Model neuronu McCulloch-Pitts (1943)



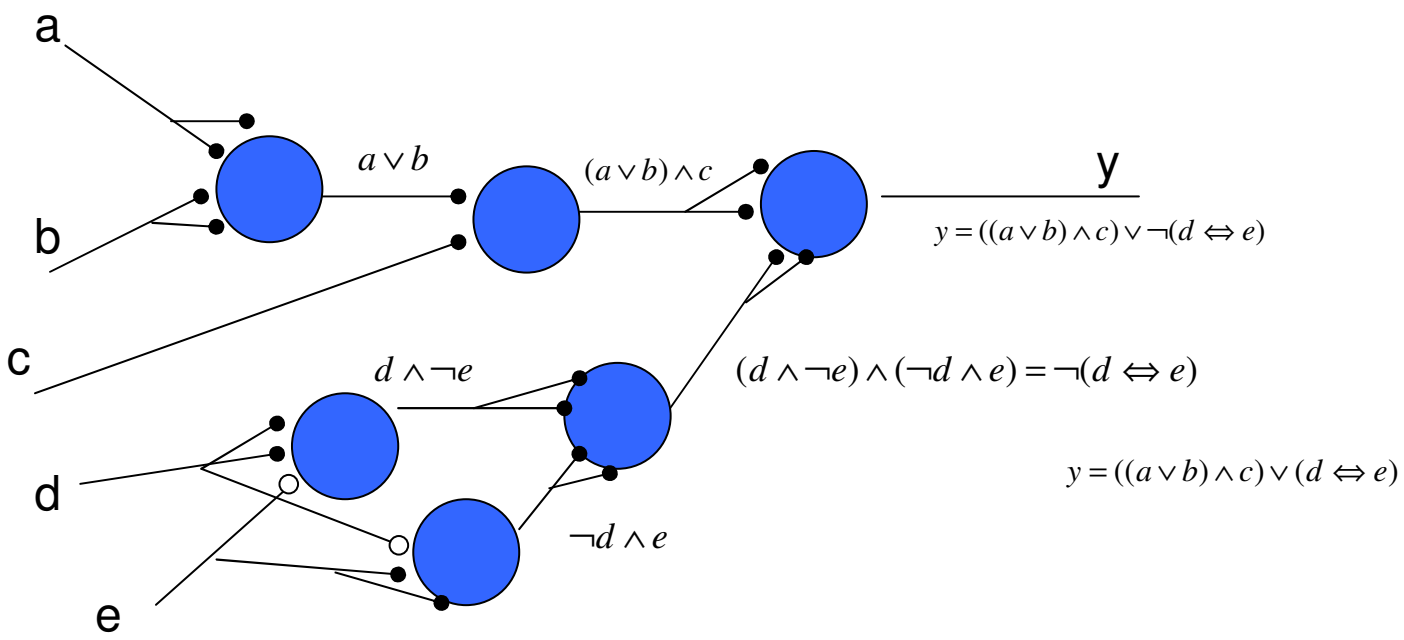
- Neuron pracuje na principu všechno nebo nic
  (0 – neaktivní (inhibovaný), 1 – aktivní (excitovaný))
- Neuron je excitován, pokud více než jeden vstup s excitační synapsí je aktivní
- Jakmile je alespoň jedna inhibiční synapse aktivní, neuron nemůže být nikdy excitován.
- Struktura zapojení mezi neurony se nemění
- Významné zpoždění mezi neurony je zpoždění synapsí

Převzato z: Freeman, J. A.,Skapura D. M.: Neural networks. Algorithms, applications, and programming techniques. Addison-Wesley Publishing Company. Červen 1992. ISBN 0-201-51376-5

Příklad

Nakreslete neuronovou síť dle McCulloch-Pittse realizující logickou funkci:

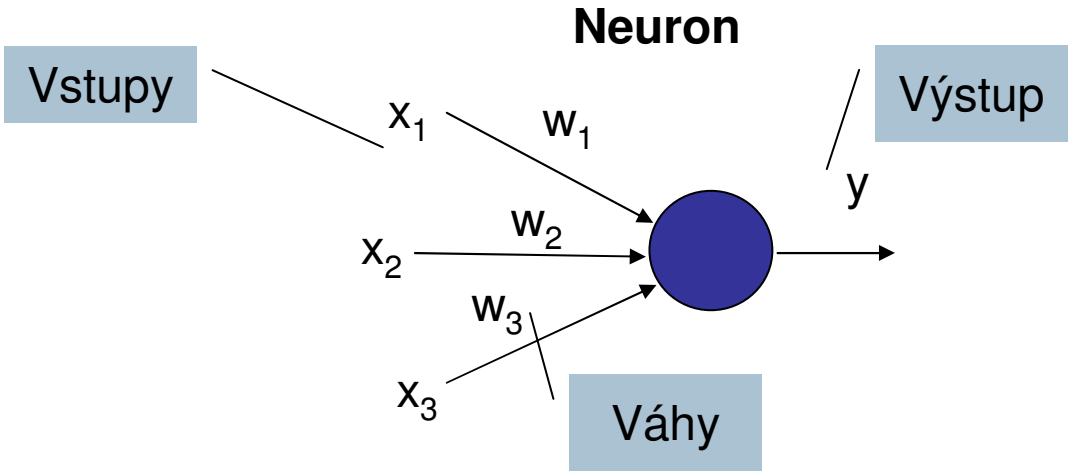
$$y = ((a \vee b) \wedge c) \vee \neg(d \Leftrightarrow e)$$



Hebovské učení

- Je-li buňka A na pokraji excitace a buňka B je opakovaně nebo trvale excitovaná, pak se posílí vazba mezi buňkou A a B (buňka A se stane citlivější na signál buňky B)
  [Donald O. Hebb, Organization of Behavior, 1949]
- Tento princip je jedním ze základních učících principů v umělých neuronových sítích

Spojité umělé neurony



Nelineární výstupní funkce

Perceptron
(Frank Rosenblatt,
50 léta 20. století)

$$y = S\left(\sum_{i=1}^{N} w_i x_i + \Theta\right)$$

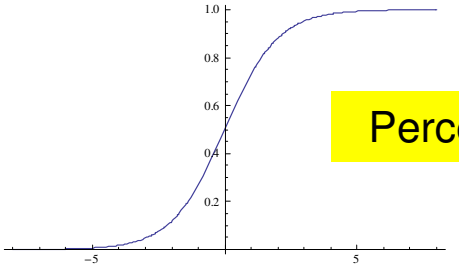
Práh

RBF Neuron

$$y = G\left(\sqrt{\sum_{i=1}^{N} (x_i - w_i)^2}\right)$$

Výstupní funkce neuronu

Výstupní funkce - Sigmoida



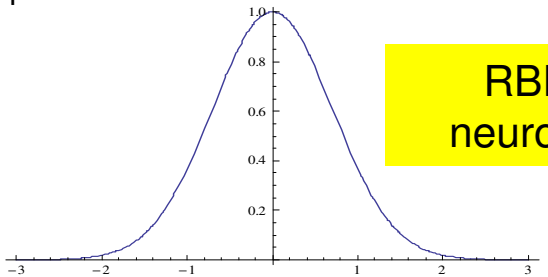
$$y = S(x) = \frac{1}{1 + e^{-\gamma . x}}$$

Perceptrony

Pozn.: parametr γ určuje strmost křivky. Obvykle konstanta pro všechny neurony v síti.

Do kategorie nepatří

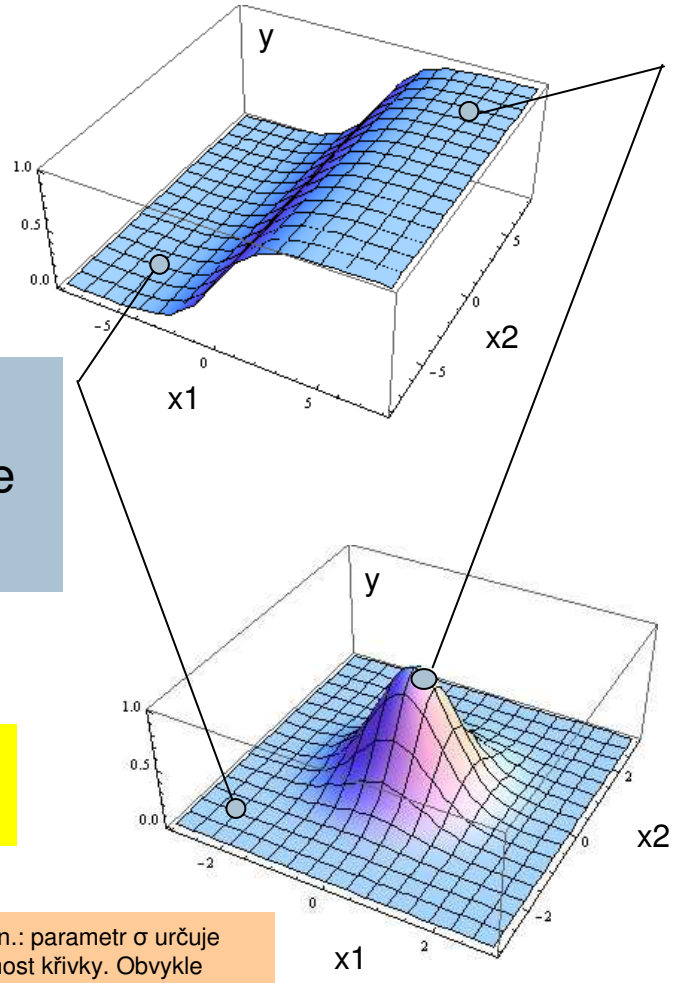
Výstupní funkce – Gaussova křivka



$$y = G(x) = e^{-\frac{x^2}{2\sigma^2}}$$

RBF neurony

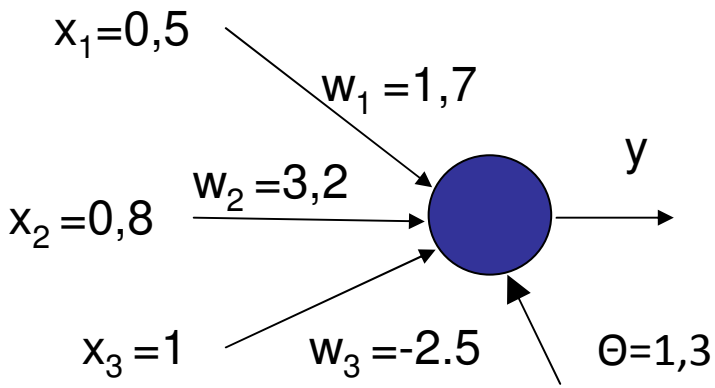
Odezva neuronu se dvěma vstupy



Do kategorie patří

Pozn.: parametr σ určuje strmost křivky. Obvykle konstanta pro všechny neurony v síti.

Příklad: spočtěte výstup perceptronu



Perceptron $$y = S(\sum_{i=1}^{N} w_i x_i + \Theta) = \frac{1}{1+e^{-1,5(0,5.1,7+0,8.3,2+1.(-2,5)+1,3)}}$$

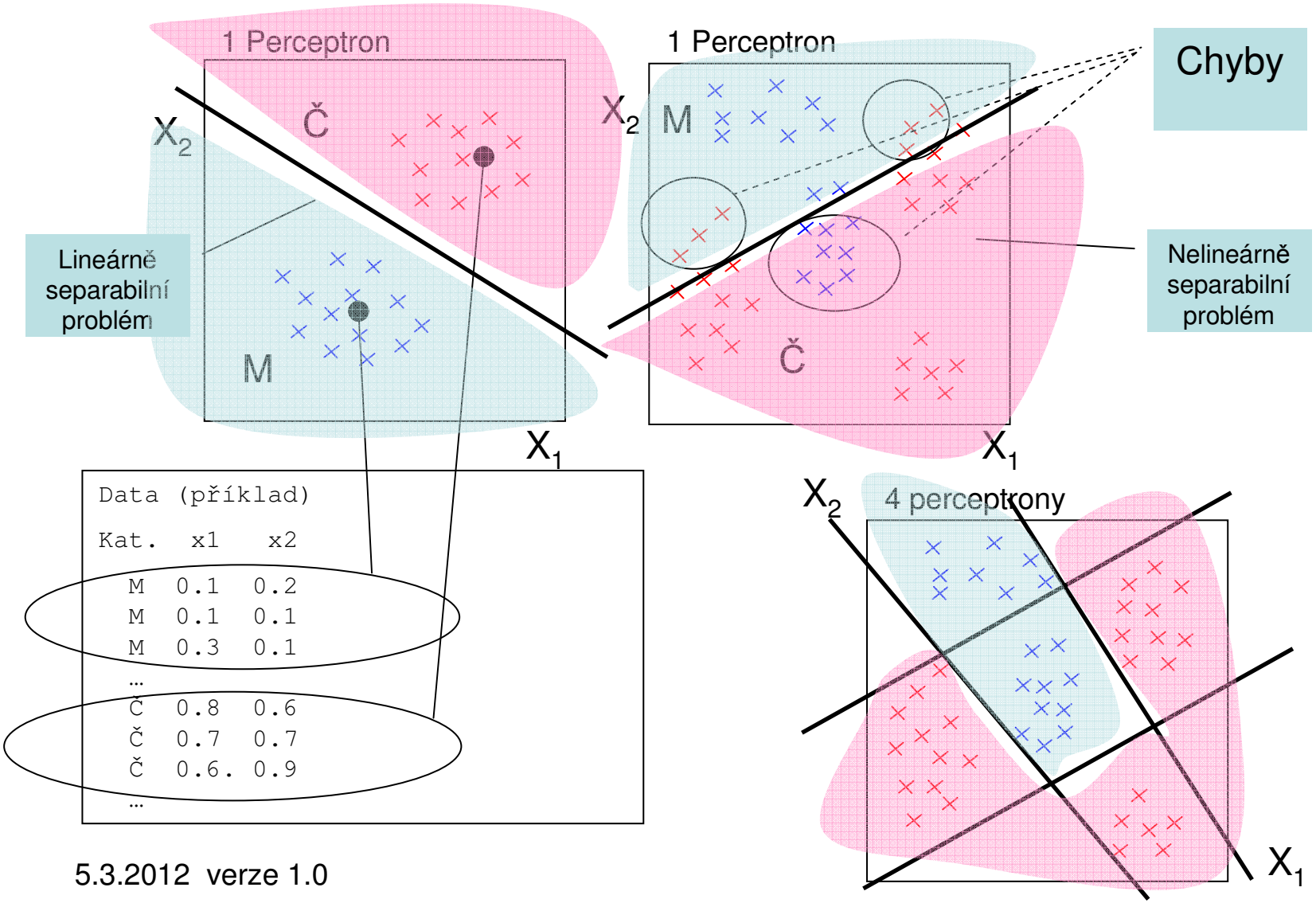
Neuronové sítě a výpočetní inteligence

Neuronové sítě

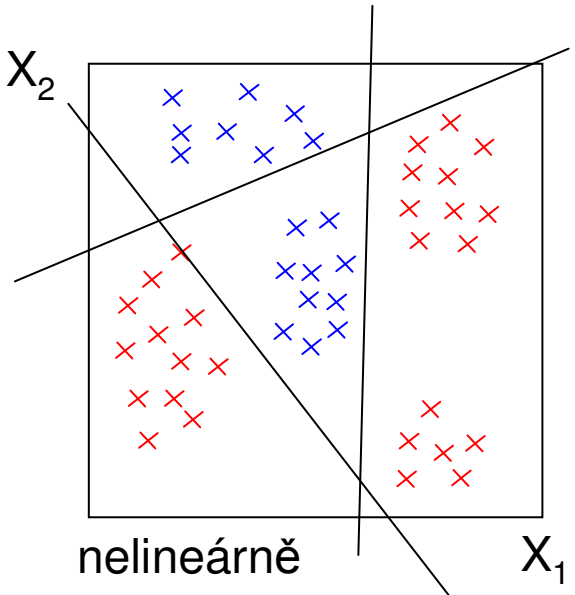
Miroslav Skrbek ©2010

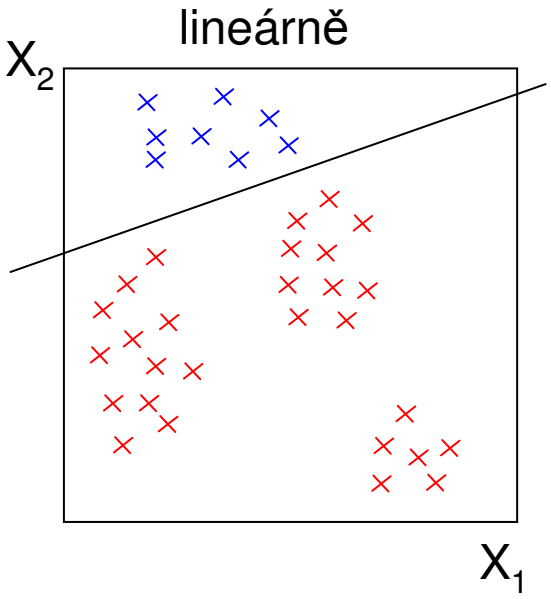
LS2011/12

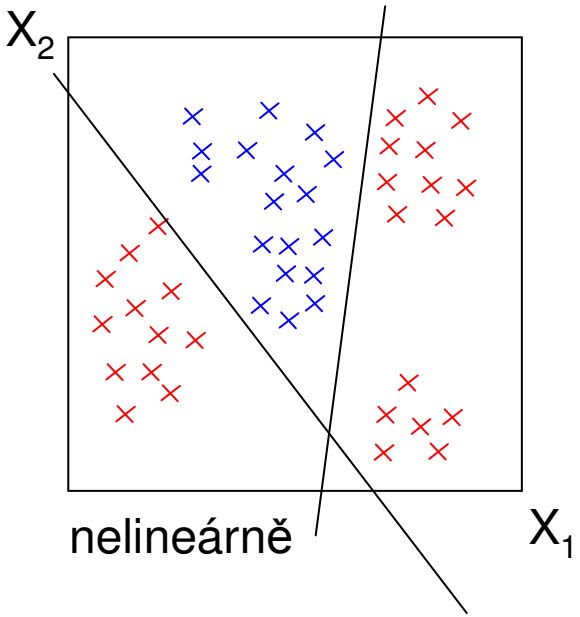
Lineární a nelineární separabilita



Úkol: určete, který problém je lineárně separabilní ?

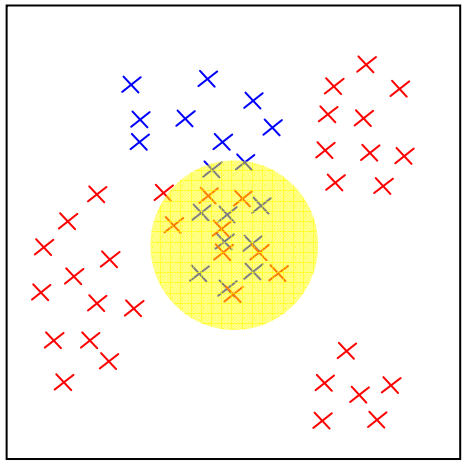






Příklad neseparovatelných dat: zjištění identity člověka (jméno a příjmení) podle jeho výšky. Často najdeme dva lidi různého jména se stejnou výškou.

Pozn.: měříme-li více antropologických dat (např. šířku očí, vzdálenost roviny očí od nosu, úst a brady, šířku lebky v rovině očí apod.) získáme nelineárně separovatelný problém (viz. biometrická data v cestovním pasu).



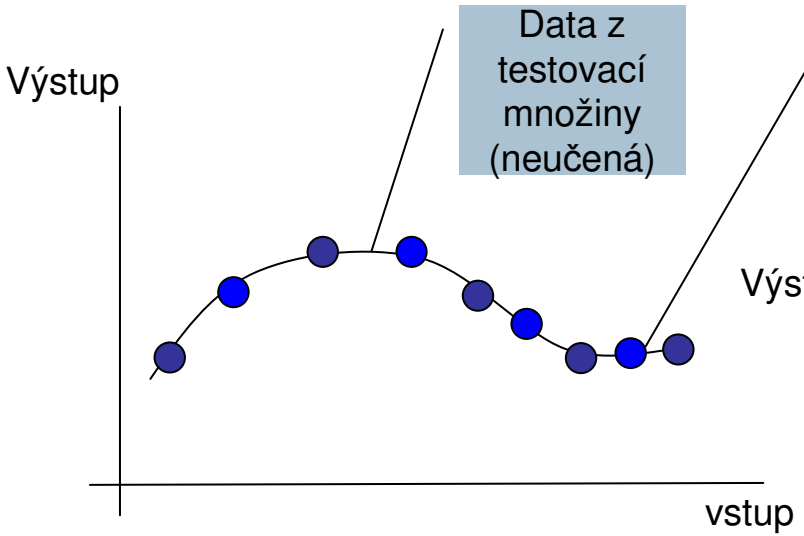
A co tento ?

Chyták ! Tohle je neseparovatelný problém

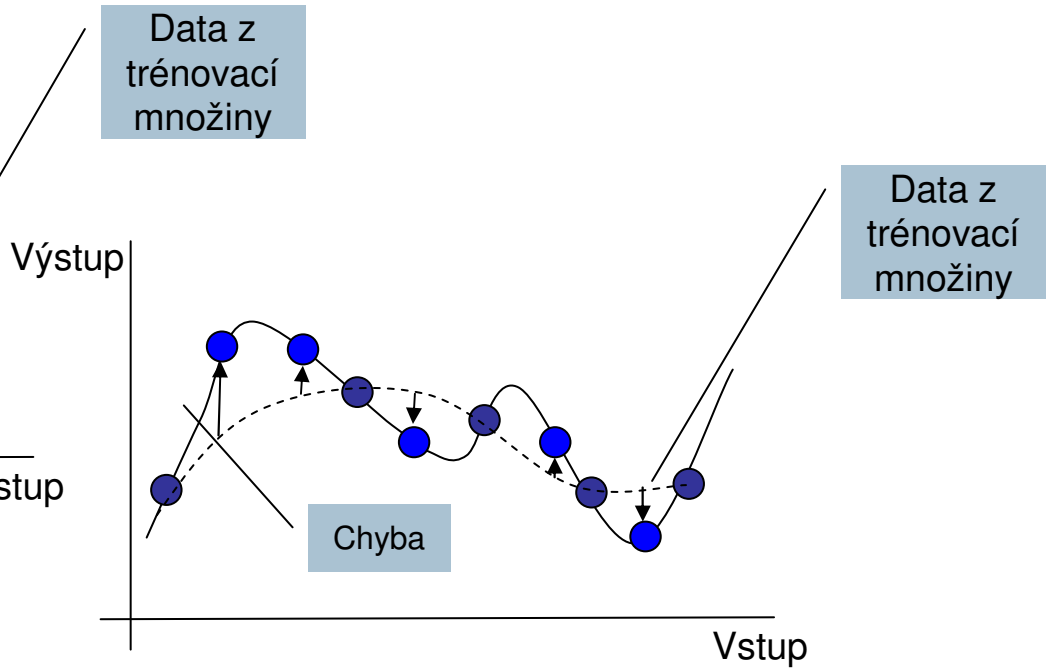
Množiny M a Č promíchány, nelze je oddělit jinak, než prvek po prvku nebo po malých skupinách. Častý případ, shodná data reprezentují různé kategorie.

Zevšeobecnění

Zevšeobecnění je schopnost neuronové sítě reagovat na neučená data s minimální chybou.



Neuronová síť s dobrou schopností zevšeobecňovat (=> kvalitní model)

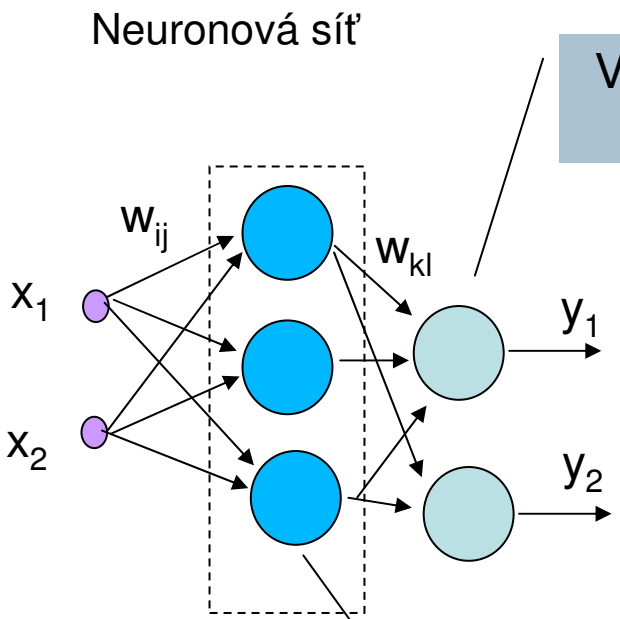


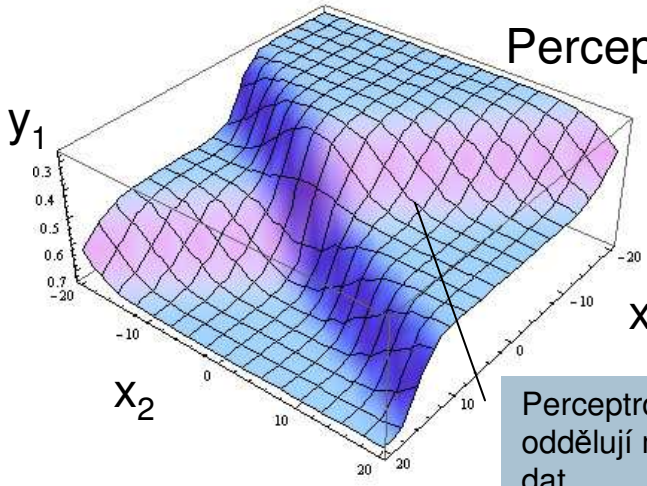
Přeučená neuronová síť, malá chyba při učení, velké chyby při testování (=> špatný model)

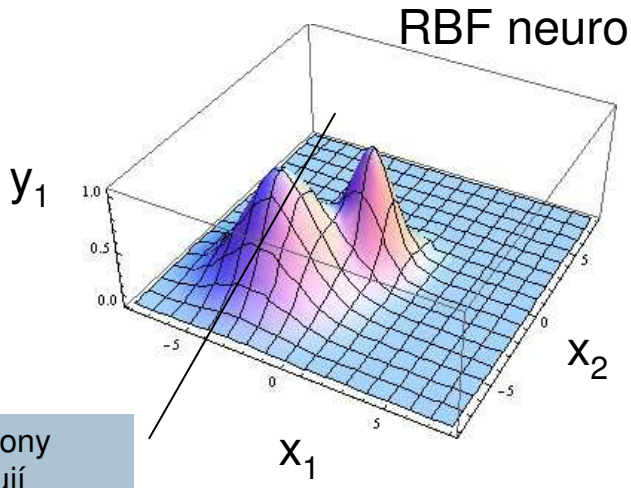
Umělá neuronová síť

- Dle teorie třívrstvá neuronová síť může aproximovat libovolnou funkci, tj. má schopnost modelovat libovolnou závislost vstup-výstup
- Neuronová sít se učí na základě příkladů (páry vstup-výstup), nepotřebuje předem analýzu problému (jak je tomu např. při realizaci modelu programem)
- Učení je proces, ve kterém se hledají takové hodnoty vah neuronů sítě, pro které bude rozdíl mezi odezvou neuronové sítě a daty minimální
- Jsou-li váhy nalezeny, pak přiložíme vstupní data na vstupy sítě a dostaneme odezvu (modelu), která např. napoví jestli hodnota našich akcií poroste (klesne, případně o kolik), pokud na vstupech byla hodnota akcie za posledních několik dnů

Neuronové sítě







Topologie neuronových sítí

- S dopředným šířením
  - Realizuje mapovací funkci y=f(x)
  - signál se šíří pouze od vstupu k výstupu
  - Jednokrokový výpočet od vstupu k výstupu (každý neuron počítán jen jednou)
  - realizují funkci, která mapuje vstup na výstup bez ohledu na pořadí hodnot přiložených postupně na vstupy
- Rekurentní
  - Má vnitřní stav, mapuje $y_{t+1}=f(q_t),g(q_t+1,x_t)=q_t$
  - vhodné pro časové řady
  - Kromě dopředných spojení mezi neurony existují i zpětné vazby (signál se šíří směrem ke vstupu)
  - Iterativní (opakovaný) výpočet, některé neurony se počítají vícekrát.

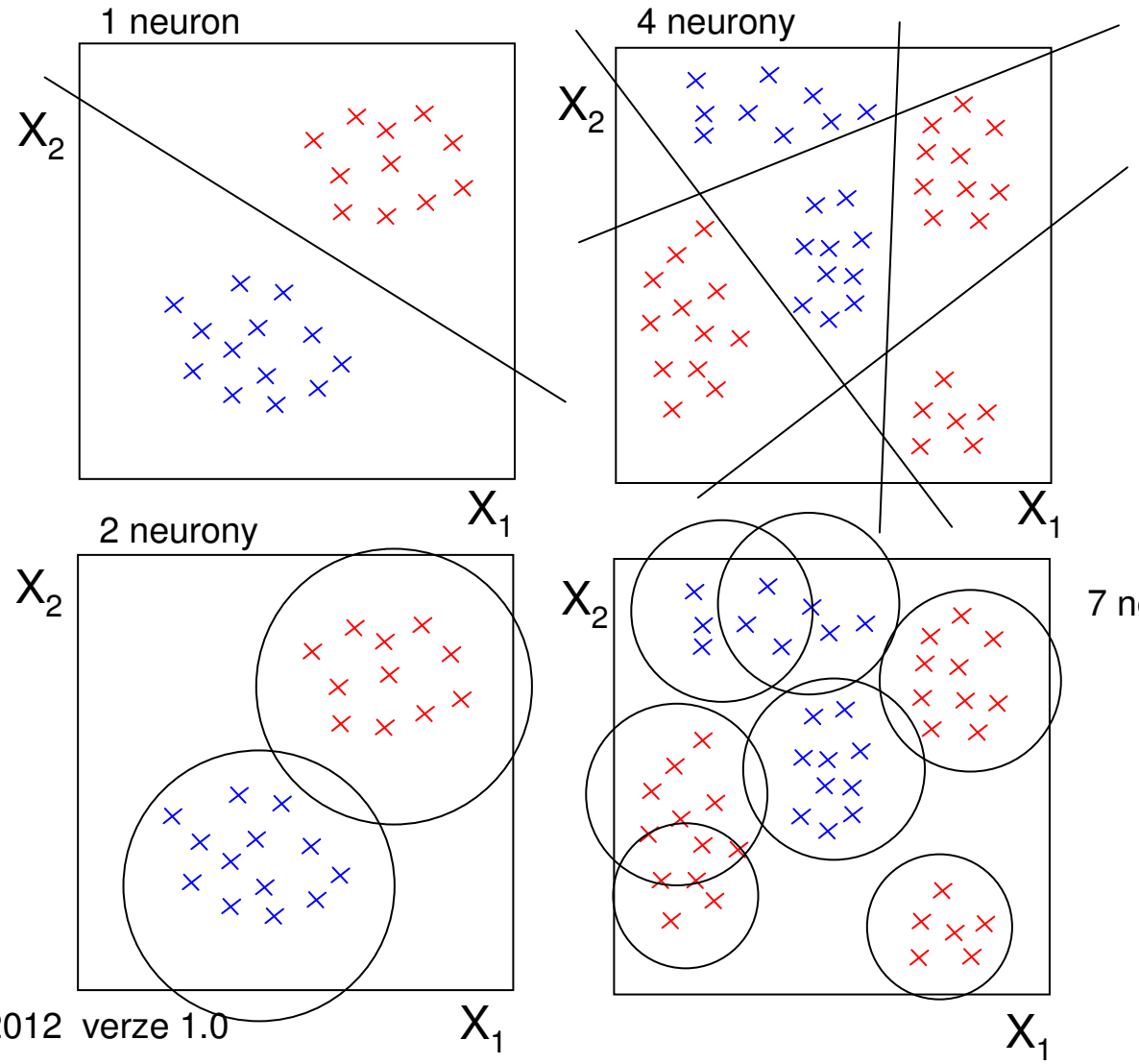
Co určuje schopnosti neuronové sítě

- Topologie
  - Typ
    - Dopředná
    - Rekurentní
  - Hustota propojení
    - Úplné propojení
    - Částečné propojení
  - Počet vrstev
  - Počet neuronů (ve skrytých vrstvách)
- Vlastnosti neuronů
  - Typ aktivační funkce
  - Metrika

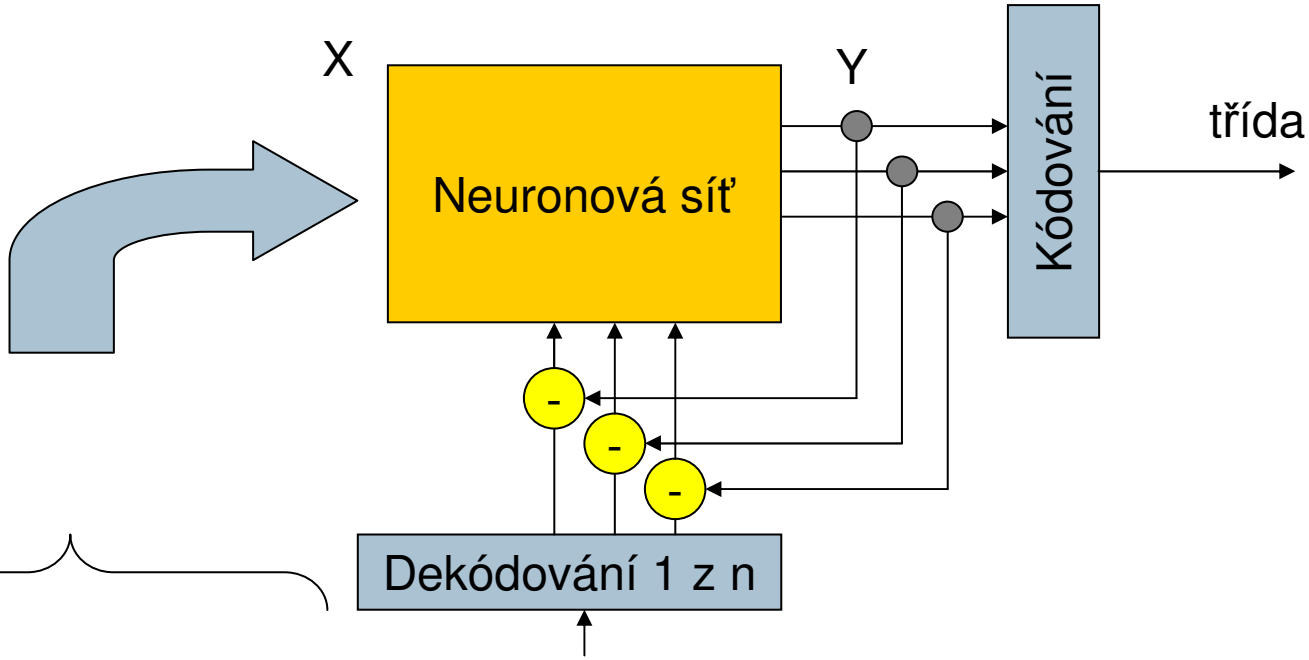
Jak velkou neuronovou síť zvolit ?

- Počet neuronů ve vstupní a výstupní vrstvě je dán aplikací, volíme pouze počty skrytých neuronů
- Neexistují jednoznačná pravidla
- Platí zásada méně je více
- Začínáme s jednou skrytou vrstvou, více vrstev přidáme, pokud jsme nespokojeni s výsledky
- Počet neuronů ve skryté vrstvě by měl odpovídat struktuře (členitosti) dat (více shluků potřebuje více neuronů)
- Více neuronů→snadnější přeučení→malá generalizace →horší model

Počet neuronů v skryté vrstvě

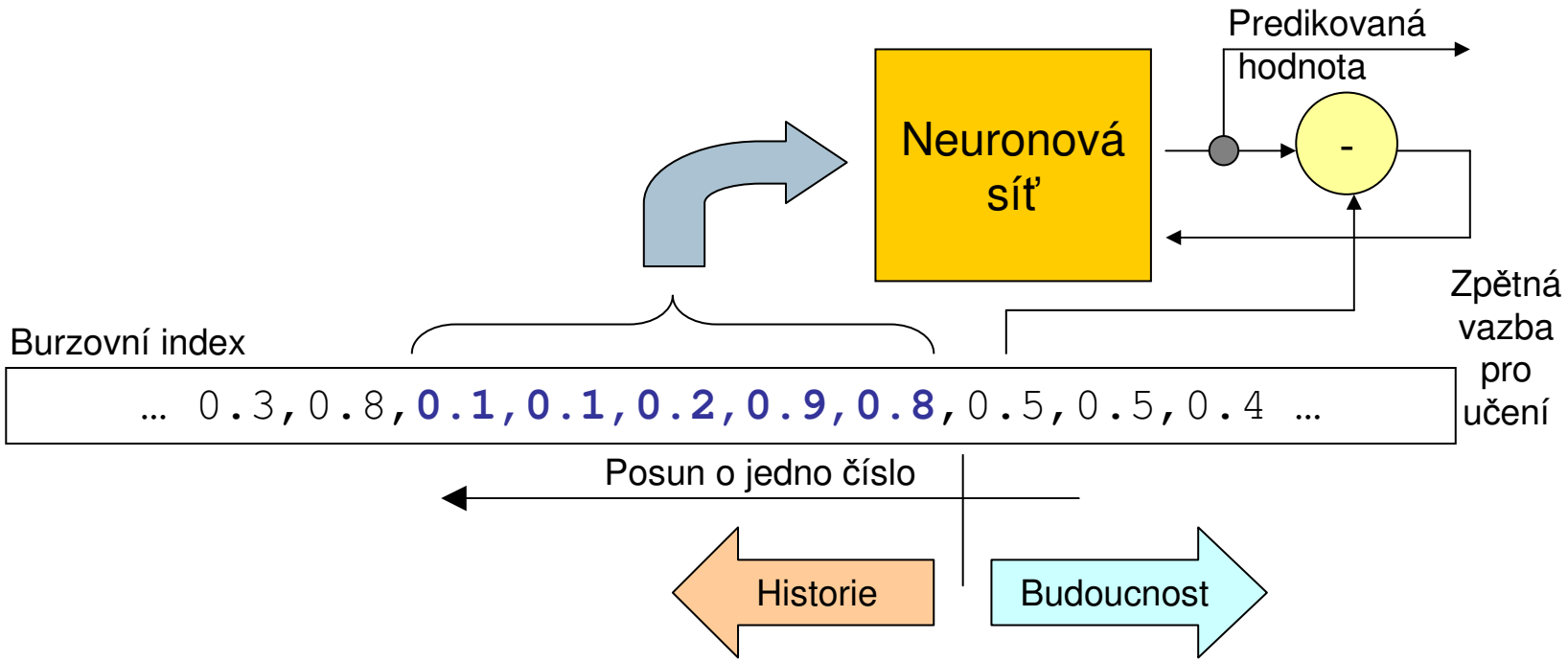


Typická úloha - klasifikace



X1	x2	x3	x4	třída
0.1	0.2	0.1	-0.2	1
0.2	-0.8	-0.5	0.4	7
0.8	-0.1	0.4	0.4	3
…				
0.1	-0.4	0.4	0.7	4

Predikce



Predikovat lze více budoucích hodnot, nutno počítat růstem chyby

Neuronové sítě a výpočetní inteligence

Neuronové sítě

Miroslav Skrbek ©2010

Učení neuronové sítě

- Učení je optimalizační úloha, která najde pro danou trénovací množinu dat, takovou konfiguraci vah, kdy je průměrná chyba sítě přes celou trénovací množinu menší než požadované kriterium
- Často používané jsou gradientní metody nebo přírodou inspirované algoritmy (např. genetický algoritmus)
- Požadavek na příliš nízkou chybu vede k přeučení, tj. nízké chyby pro data v trénovací množině, pro jiná data vysoké chyby.

Pojmy v učení

- **Vzor**
  - Vzor je element datového souboru určeného pro zpracování neuronovou sítí
  - Typicky je to pár vstupní vektor a výstupní vektor. Vstupní vektor dat se přikládá na vstupy neuronové sítě, výstupní vektor reprezentuje požadovanou hodnotu výstupů. Některé neuronové sítě např. SOM výstupní vektor nepotřebují, pak vzor obsahuje pouze vstupní vektor.
- **Trénovací množina**
  - Množina vzorů určená pro učení neuronové sítě
- **Testovací množina**
  - Množina vzorů určená pro testování neuronové sítě
- **Validační množina**
  - Množina vzorů určená pro testování neuronové sítě během učení (používá se pro zabránění přeučení)
- **Vybavení**
  - Předložení jednoho vzoru neuronové sítí a výpočet výstupu neuronové sítě
- **Iterace**
  - Předložení jednoho vzoru neuronové sítě, provedení vybavení a následně jednoho kroku učení
- **Epocha**
  - Provedení iterace pro všechny vzory z trénovací množiny
- **Učení**
  - Proces výpočtu váhových (případně jiných) koeficientů na základě trénovací množiny
  - Učení se skládá z jedné či více epoch
- **Přeučení**
  - Nadměrná doba učení (příliš mnoho provedených epoch učení), která způsobí růst chyby pro netrénovaná data (zhoršení zevšeobecnění)
- **Lokální minimum**
  - Parazitní minimum na energetické (chybové) hyperploše, jehož hodnota je vyšší než chyba očekávaného globálního minima.
- **Zevšeobecnění (generalizace)**
  - Schopnost reakce správné reakce na neučená data
- **Hebbovo pravidlo**
  - Pravidlo pro úpravu vah. Posilování vah mezi neurony, které vykazují podobné reakce na stejné podněty
- **Delta pravidlo**
  - Pravidlo pro úpravu vah v učícím algoritmu Back Propagation

Příklad učení v programu Mathematica

Matematika: parciální derivace

Mějme funkci $f(x_1, x_2, x_3, \ldots x_n)$ více proměnných. Tato funkce má v bodě $B=(b_1,b_2, b_3,\ldots,b_n)$ parciální derivaci podle x_i, pokud existuje limita

$$\frac{\partial f(x_1, x_2, \ldots, x_n)}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1 + h, x_2, \ldots, x_n) - f(x_1, x_2, \ldots, x_n)}{h}$$

Pro výpočet parciální derivace využíváme stejná pravidla, jako v případě funkce jedné proměnné, přičemž všechny proměnné vyjma té, podle které derivujeme, považujeme za konstanty (tj. jejich derivace jsou rovny nule).

Příklad: $f(x, y, z) = 5x^2 - 8\log(y) + \sin(\omega z)$

$$\frac{\partial f(x, y, z)}{\partial x} = 10x, \frac{\partial f(x, y, z)}{\partial y} = -8\frac{1}{y}, \frac{\partial f(x, y, z)}{\partial z} = \omega\cos(\omega z)$$

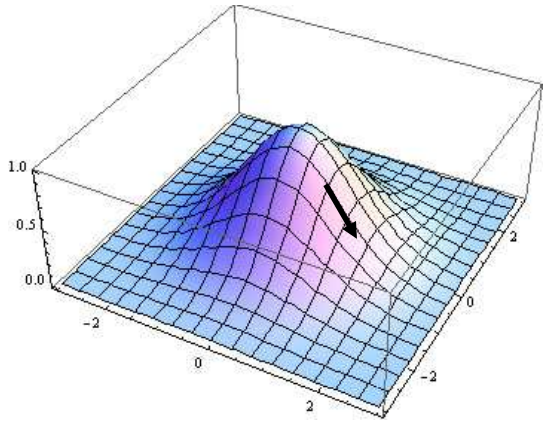
Matematika: gradient

Gradient je diferenciální operátor. Je-li aplikován na funkci f více proměnných, pak je jeho výsledkem vektor parciálních derivací podle všech proměnných funkce f.

nabla

$$\nabla f(x_1, x_2, \cdots, x_n) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n} \right)$$

Gradient má jasnou geometrickou interpretaci. Je to směr největšího spádu v daném bodě na ploše funkce f.



$$\nabla G(x,y) = \nabla(e^{\frac{-(x^2+y^2)}{2\sigma^2}}) = \left(-\frac{x}{\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}, -\frac{y}{\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \right)$$

Matematika: optimalizace

Optimalizace je proces, ve kterém hledáme množinu parametrů $x_1, x_2, \ldots, x_n$ tak, abychom minimalizovali tzv. cenovou (cílovou, účelovou) funkci.

Lineární regrese: mějme N bodů grafu. Cílem je, co nejoptimálněji tyto body proložit přímkou. *Cenová funkce je součet kvadrátů chyby mezi prokládající přímkou a hodnotami zadaných bodů.*

Snadno řešitelné metodou nejmenších čtverců

Problém batohu: mějme N různě těžkých předmětů, přičemž každý předmět má svoji cenu (v Kč) a objem. Nalezněte takový výběr předmětů, který se ještě vejde do batohu o daném objemu a zároveň má ze všech přípustných výběrů největší cenu. *Cenovou funkcí je zde cena předmětů v batohu vetovaná přípustným objemem batohu.*

Těžko řešitelné - NP problém

Neuronové sítě a výpočetní inteligence

Neuronové sítě

Miroslav Skrbek ©2010

Učení s učitelem

Učení s učitelem je založeno na minimalizaci chyby mezi momentální a požadovanou odezvou neuronové sítě.

Roli učitele zde hraje požadovaná odezva sítě, kterou se snažíme neuronovou síť naučit.

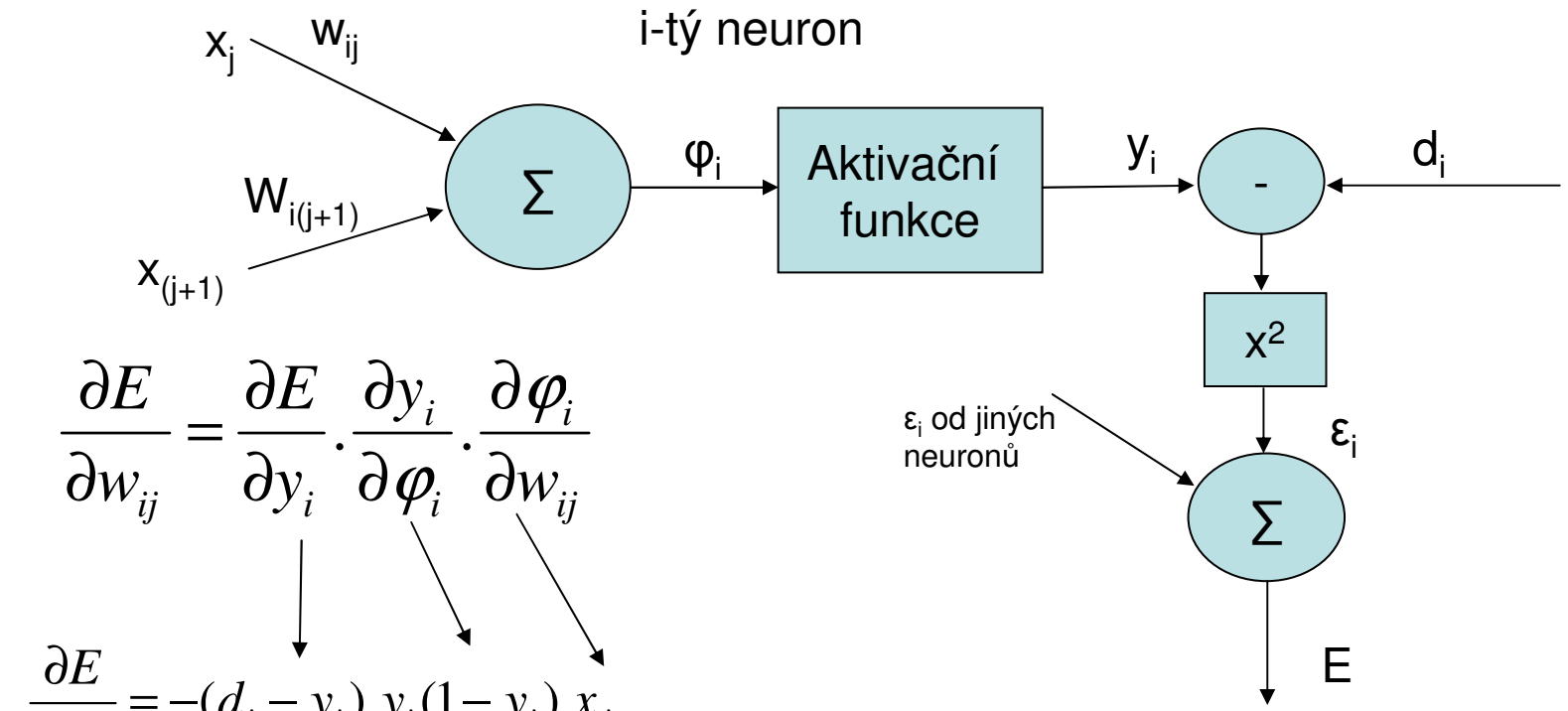
V průběhu učení změnami vah minimalizujeme chybu

$$E = \frac{1}{2}\sum_{j=1}^{M}\sum_{i=1}^{N}(y_i^{(j)} - d_i^{(j)})^2$$

Kde N je počet výstupů neuronové sítě a M je počet vzorů v trénovací množině. Pozor ! $y_i^{(j)}$ je i-tý výstup j-tého vzoru z trénovací množiny.

Pozn.: pro zobrazovací účely a účely stop kriteria se výše uvedený vzorec transformuje na střední kvadratickou chybu (RMS). RMS=√(2E/M/N), která nabývá hodnoty mezi 0 a 2 a je nezávislá na velikosti neuronové sítě (na počtu výstupů) a počtu vzorů v trénovací množině.

Odvození algoritmu Back Propagation



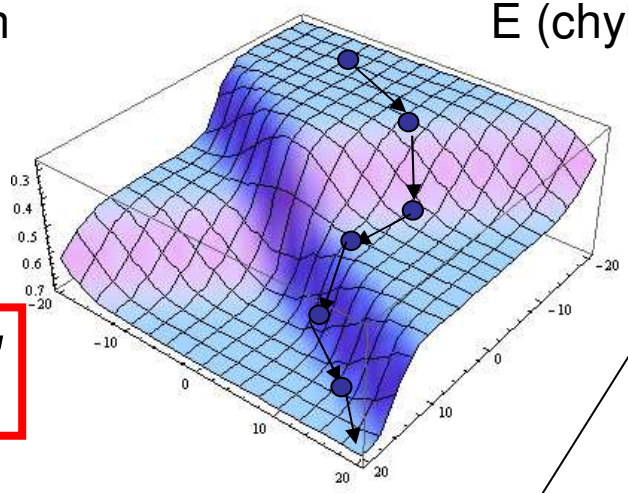
$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial \varphi_i} \cdot \frac{\partial \varphi_i}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = -\underbrace{(d_i - y_i).y_i(1-y_i)}_{\delta_i}.x_j$$

Pozn.: y(1-y) je derivace aktivační funkce, konkrétně sigmoidy a gamma=1. Pro tanh bude tato derivace jiná: (1+y)(1-y). Derivace se odvodí z derivace inverzní funkce (proto je funkcí y a ne x). Ta je totiž jednodušší a lépe se počítá.

Gradientní metoda učení algoritmem Back Propagation

Inicializace vah náhodně ±0.5



Postup v místě nejvyššího spádu

$$\Delta w = -\eta \nabla E$$

Chyba připadající na i-tý neuron

Aktualizace vah (delta pravidlo)

$$\Delta w_{ij}(t+1) = \eta * \delta_i(t) x_j + \alpha \Delta w_{ij}(t)$$

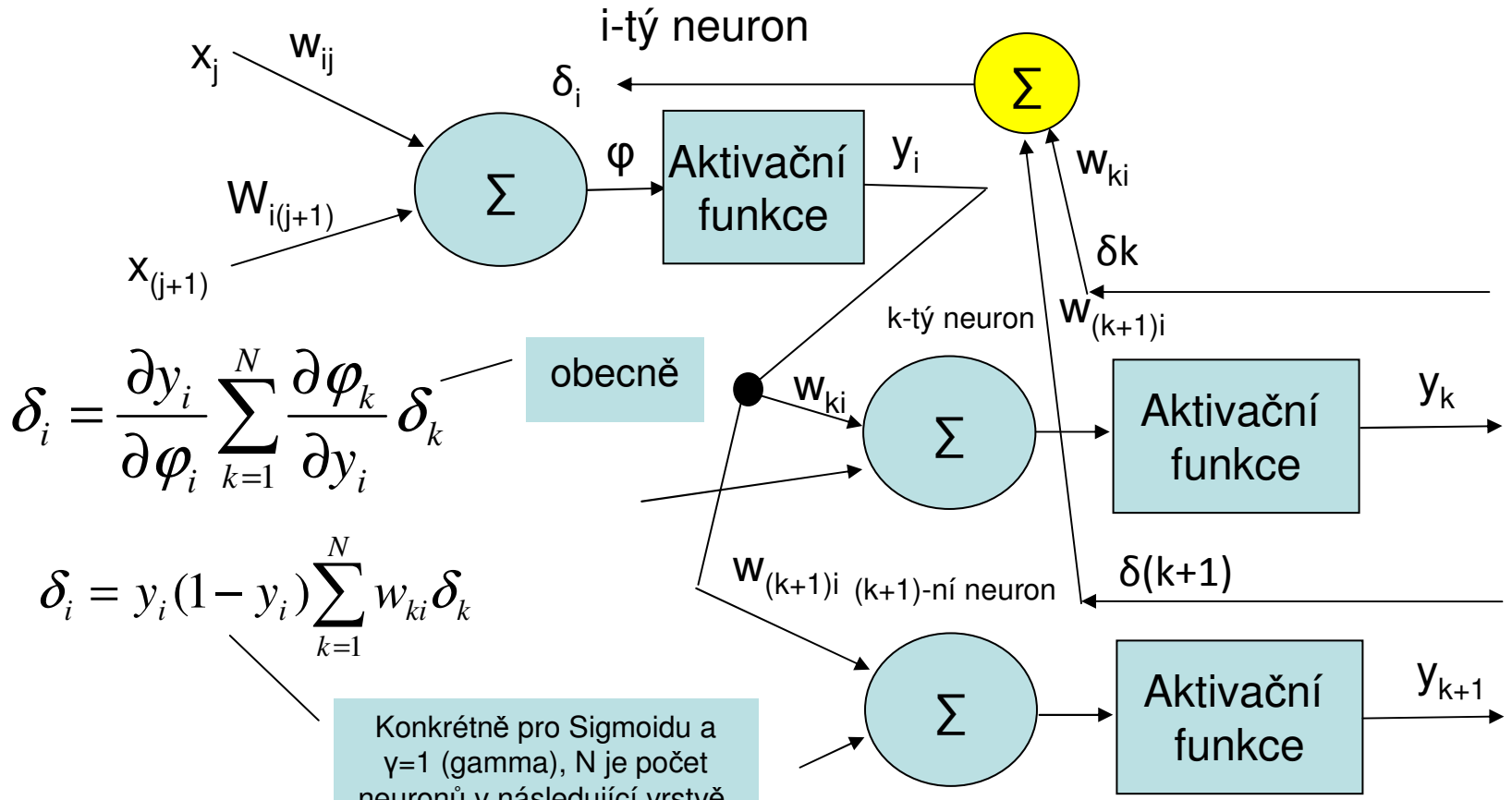
Moment <0,1) Alfa

Změna váhy mezi i-tým a j-tým neuronem

Rychlost učení <0,1> Eta

j-tý vstup neuronu

Zpětná propagace chyb do vnitřních (skrytých) vrstev



$$\delta_i = \frac{\partial y_i}{\partial \varphi_i} \sum_{k=1}^{N} \frac{\partial \varphi_k}{\partial y_i} \delta_k$$

$$\delta_i = y_i(1-y_i)\sum_{k=1}^{N} w_{ki}\delta_k$$

Konkrétně pro Sigmoidu a γ=1 (gamma), N je počet neuronů v následující vrstvě, tj. vrstvě blíže výstupům

Pozn.: Delta pravidlo je pro vnitřní vrstvy stejné

Algoritmus učení

1. Náhodně inicializujeme váhy v neuronové síti
2. Vezmeme pár vstup-výstup z trénovací množiny dat a část vstup přiložíme na vstup sítě
3. Vypočteme výstupy sítě
4. Porovnáme výstup sítě s částí výstup z páru vstup-výstup (vypočteme odchylku)
5. Na základě odchylky upravíme o kousek váhy sítě (získáme o něco lepší řešení)
6. Pokud jsme ještě nevyčerpali celou trénovací množinu pokračujeme bodem 2
7. Pokud je průměrná chyba přes celou trénovací množinu vyšší než požadované kriterium, pokračujeme bodem 2
8. Síť je naučena, konec

Parametry učení (u gradientních metod)

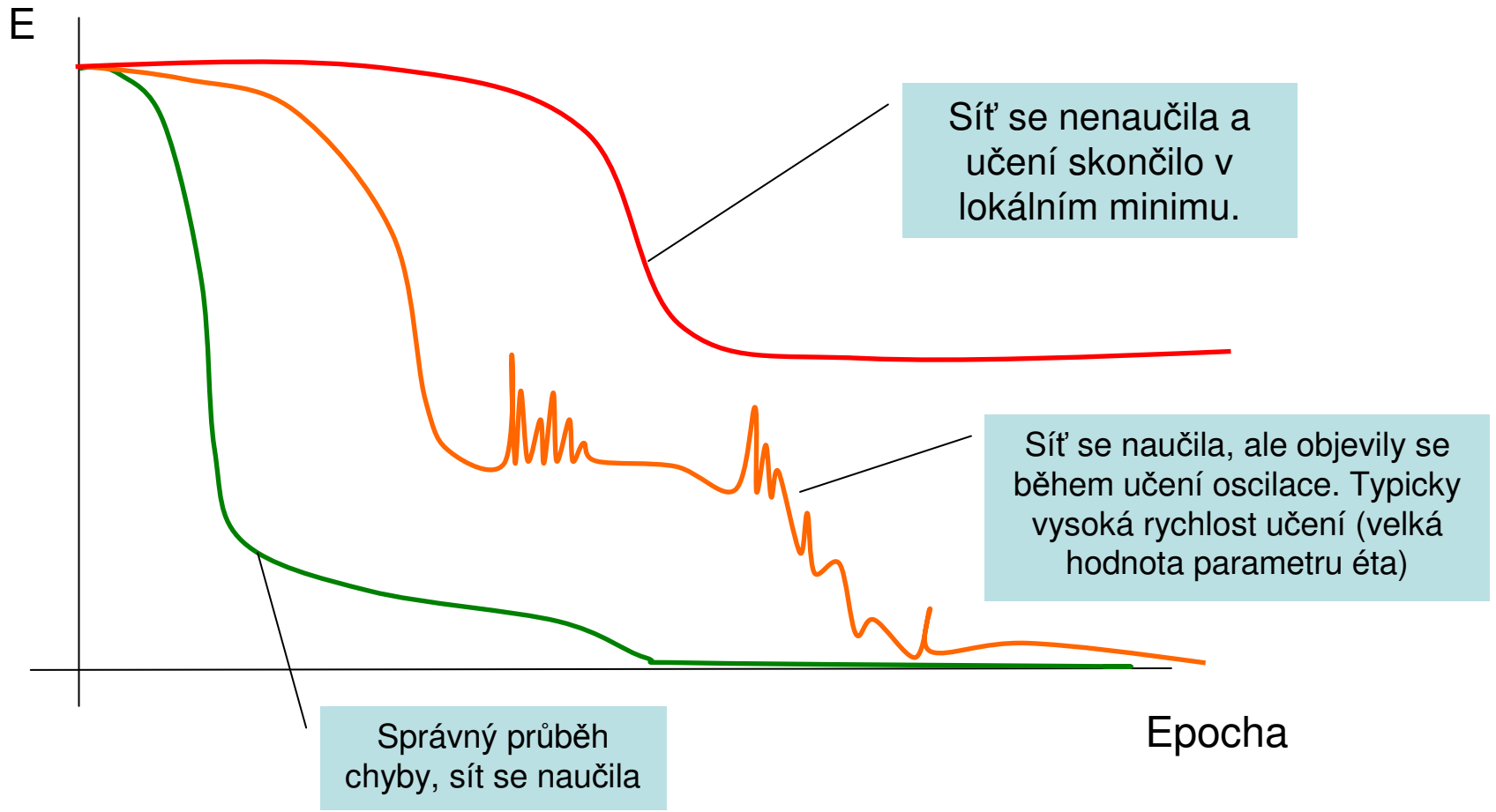
Rychlost učení (Eta)

Je koeficient, který určuje o kolik (relativně k velikosti chyby) se má upravit váha neuronu (bod 5.). Čím je koeficient větší, tím rychleji se síť učí (rychleji dosáhne bod 8.). Pozor ! vysoká hodnota, ale může způsobit, že se nenajde optimální řešení a bodu 8 se nikdy nedosáhne. Při vysoké rychlosti může docházet k oscilacím. Trpělivost se tedy vyplácí a s nižším koeficientem můžete nalézt řešení spíše než s vysokou hodnotou.

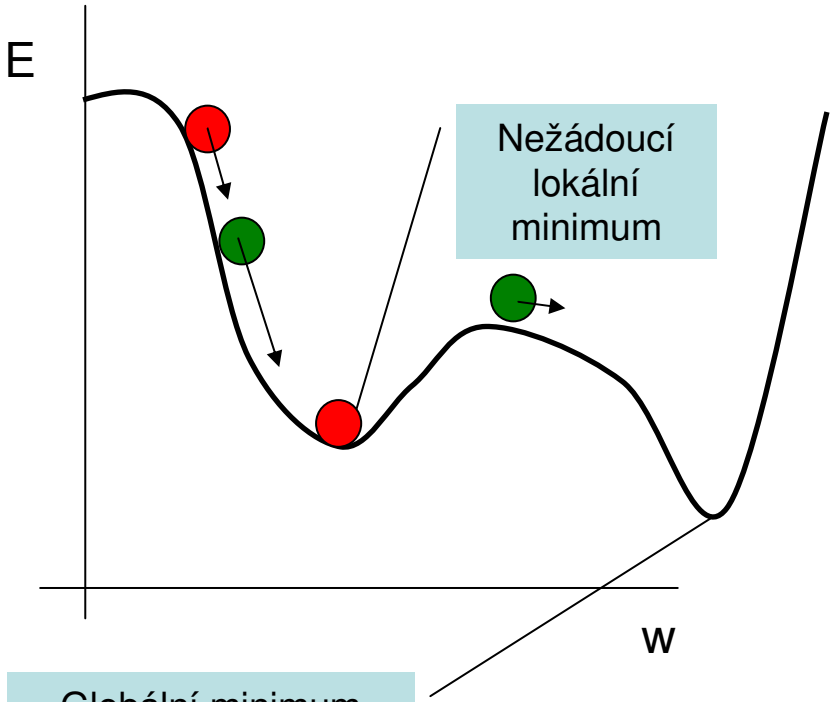
Setrvačnost (Alfa)

Je koeficient, který udává, jakou měrou se má do změny váhy započítat její předchozí změna. Změna váhy má dvě složky: chybu odezvy sítě, která se započítává koeficientem rychlosti učení a předchozí změna váhy. Pokud je změna váhy malá, dominuje složka chyby sítě, v opačném případě, když dočasně chyba sítě klesne, je váha stále upravována ve velikosti předchozí změny (podobně jako auto ve vysoké rychlosti přejíždí hrbol). Setrvačnost může pomoci se vymanit z lokálního minima a nalézt lepší konfiguraci vah s menší chybou.

Průběh chyby během učení



Uváznutí v lokálním minimu



Kulička představuje aktuální hodnoty všech vah a posouvá se dle gradientu E, tj. dle gradientu se přičte ke každé váze delta w dle hodnoty gradientu násobeného parametrem eta (to je základní princip algoritmu).

Dospěje-li kulička do lokálního minima, krok kterýmkoliv směrem znamená zvýšení chyby a za „kopcem“ algoritmus globální minimum nevidí (z tohoto pohledu je slepý). Algoritmus proto skončí končí s chybou lokálního minima.

Časté řešení

Zavedeme setrvačnost (parametr alpha, viz. Delta pravidlo), kulička získá setrvačnost, která jí udržuje v pohybu nezávisle na gradientu. Zelená kulička se setrvačností se bude chovat jako reálná kulička s určitou hmotností, která se kutálí z kopce. Díky setrvačnosti vyskočí z důlku a pokračuje do globálního minima. Setrvačnost musí být ale dostatečná, jinak nepřekoná „kopec“ a bude oscilovat v lokálním minimimu do zastavení.

Řešení uváznutí v lokálním minimu

- Správné nastavení parametrů eta a alpha
  - zvýšení alpha a snížení eta (není to univerzální lék, závisí na úloze, naopak může vést k nežádoucím oscilacím, mnohdy pomůže specifický (experimentálně zjištěný) poměr eta a alpha)
- Výběr vzorů z trénovací množiny v náhodném pořadí
  - Obvykle se používá sekvenční (postupný) výběr vzorů, kdy se pořadí vzorů nemění v průběhu epochy. Toto někdy vede „mylným závislostem“, které si neuronová sít během učení vyvodí a to může někdy vést do lokálního minima (odpovíte stejně rychle na otázku kolik je 6x9 nebo 9x6).
  - Výběr vzorů se uplatní pouze pokud se váhy aktualizují v každé iteraci, pokud se aktualizují na konci epochy (Batch verze algoritmu), pak na pořadí nezáleží.
- Strukturou trénovací množiny
  - Stejné zastoupení vzorů ve všech třídách
  - Vynechání obdobných vzorů v rámci kategorie (zředění dat)
- Použitím verze gradientní metody s proměnným krokem
  - Díky konstantnímu parametru eta můžeme považovat krok algoritmu za konstantní (stejně dlouhý), existují druhy algoritmů s proměnným krokem.

Příprava trénovací množiny

- Dostupná data rozdělíme na
  - Trénovací množinu
  - Testovací množinu
- Stejné procentní zastoupení vzorů ze všech tříd
  - Selektivní výběr vzorů
    - Nevýhodou je, že neuronová síť se učí jen na zlomku dostupné informace, než je k dispozici
  - Duplikace
- Normalizace vstupních dat
  - Změna měřítka (pozor na dynamicky počítané měřítko, trénovací a testovací množina může mít odlišné hodnoty)
  - Transformace přes nelineární funkci (např. log(x), potlačení velkých hodnot, zdůraznění malých hodnot)

Alternativy algoritmu Back Propagation

- Dávkové (Batch) verze
  - Základní
    - Během epochy se akumulují Δw pro každou váhu, k aktualizaci vah dojde až na konci epochy.
  - QuickProp
    - Fahlman, 1988
  - Quasi-Newton
    - Bishop, 1995, Shepherd, 1997
  - Conjugate gradient descent,
    - Bishop, 1995, Shepherd, 1997
  - Levenberg-Marquardt,
    - Levenberg, 1944, Marquardt, 1963,
  - Delta-bar-Delta
    - Jacobs, 1988, Patterson, 1996

Neuronové sítě a výpočetní inteligence

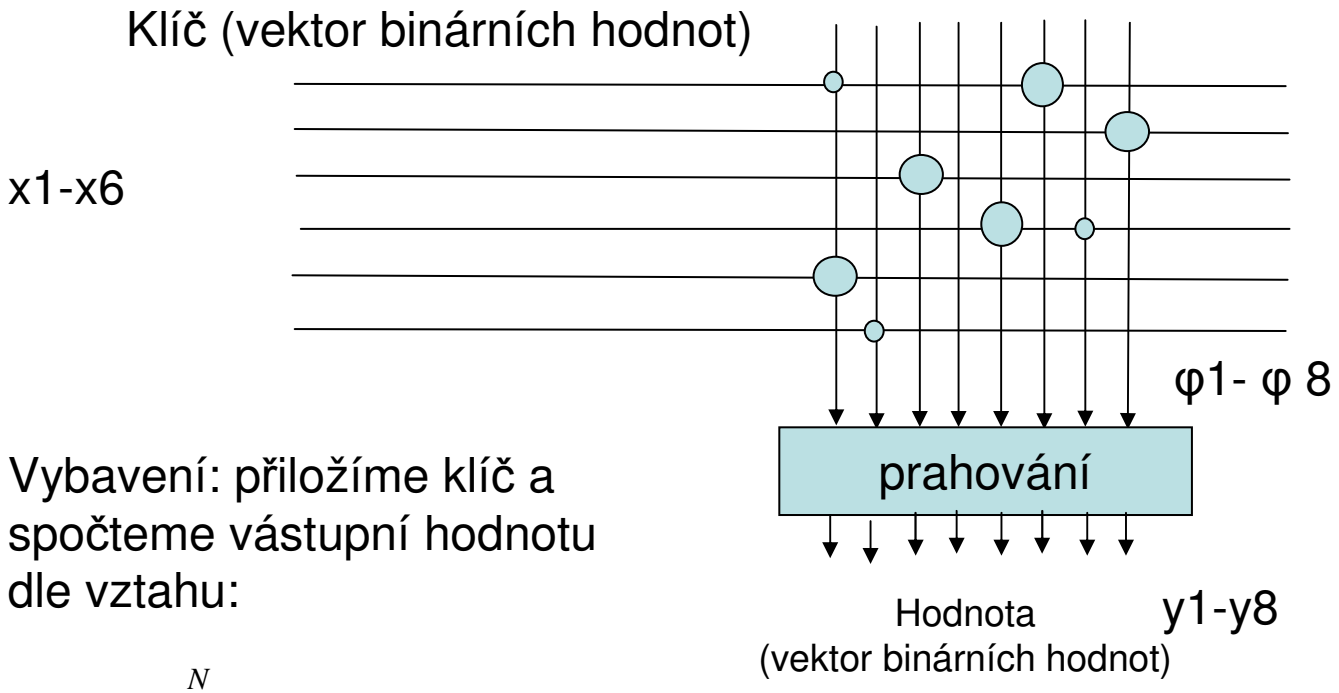
Neuronové sítě

Miroslav Skrbek ©2010

Asociativní paměti

- Přiřazují vstupnímu vektoru výstupní, typicky binární vektor
- Mají dvě varianty
  - Heteroasociativní
    - Klíč přichází na zvláštní vstupy
  - Autoasociativní
    - Klíč je součástí hodnoty a není přesně vymezen

Příklad heteroasociativní paměti



Vybavení: přiložíme klíč a spočteme vástupní hodnotu dle vztahu:

$$\varphi_i = \sum_{j=0}^{N} w_{ij} x_i$$

$$y_i = \begin{cases} 1 & \text{if } \varphi_i > \Theta_i \\ 0 & jinak \end{cases}$$

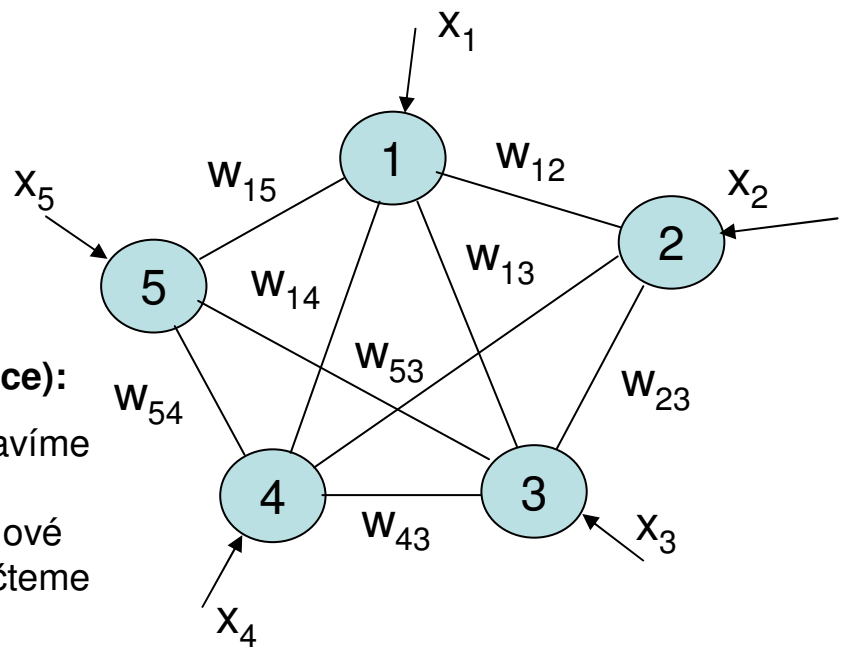
Učení: pokud váha je binární hodnota (0/1), pak je váha rovna jedné pokud pro tuto váhu vzor i hodnota nabývá hodnoty jedné.

Pro spojité váhy, lze nahradit jednovrstvou neuronovou sítí a pro nalezení vah použít např. gradientní optimalizační algoritmus.

Hopfieldova sít

- Je příkladem autoasociativní paměti
- Klíč a hodnota nelze odlišit
- Význam má pro rekonstrukci (opravu) dat
- Problémem je nízká paměťová kapacita

Hopfieldova síť



Učení je jednorázové pro danou množinu vzorů.

$$w_{ij} = \sum_{k=0}^{M} \sum_{\forall i,j: i \neq j}^{N} x_{ki} x_{kj}$$

Kde x_{ki} (x_{kj}) je i-tá (j-tá) dimenze k-tého vzoru. Proměnná k iteruje přes všechny vzory trénovací množiny.

Vybavení (jedna iterace):

Výstupy neuronů nastavíme podle vzoru na binární hodnotu (-1 nebo 1). Nové hodnoty výstupů vypočteme dle vztahu:

$$\varphi_i = \sum_{j=0}^{N} w_{ij} y_i^{(t-1)}$$

$$y_i = \begin{cases} 1 & \text{if} \quad \varphi_i > 0 \\ y_i^{(t-1)} & \text{if} \quad \varphi_i = 0 \\ -1 & \text{if} \quad \varphi_i < 0 \end{cases}$$

Iterace provádíme opakovaně až do ustálení výstupů.

Pozn: $y_i^{(t-1)}$ znamená předchozí hodnotu y_i (tj. pro φ_i=0 se výstup nemění)

Neuronové sítě a výpočetní inteligence

Neuronové sítě

Miroslav Skrbek ©2010

Shluková analýza

- Důležitá metoda analýzy dat
- Řeší problém nalezení podobností v neznámých datech
- Podobná data typicky leží v prostoru blízko sebe (tvoří shluky)
- Zajímají nás parametry shluku jako je střed (težiště) a velikost (rozptyl). Vzory v okolí středu můžeme pomažovat za reprezentanty shluku (typické hodnoty)

Příklad

Máme data z vyšetření od skupiny pacientů a máme zjistit, zda existují takové podskupiny skupiny pacientů, kteří mají podobné výsledky vyšetření. V dalším kroku pak zkoumáme, zda tyto skupiny náležejí k zdravým, či nemocným, případně s jakou závažností nemoci, případně které nemoci.

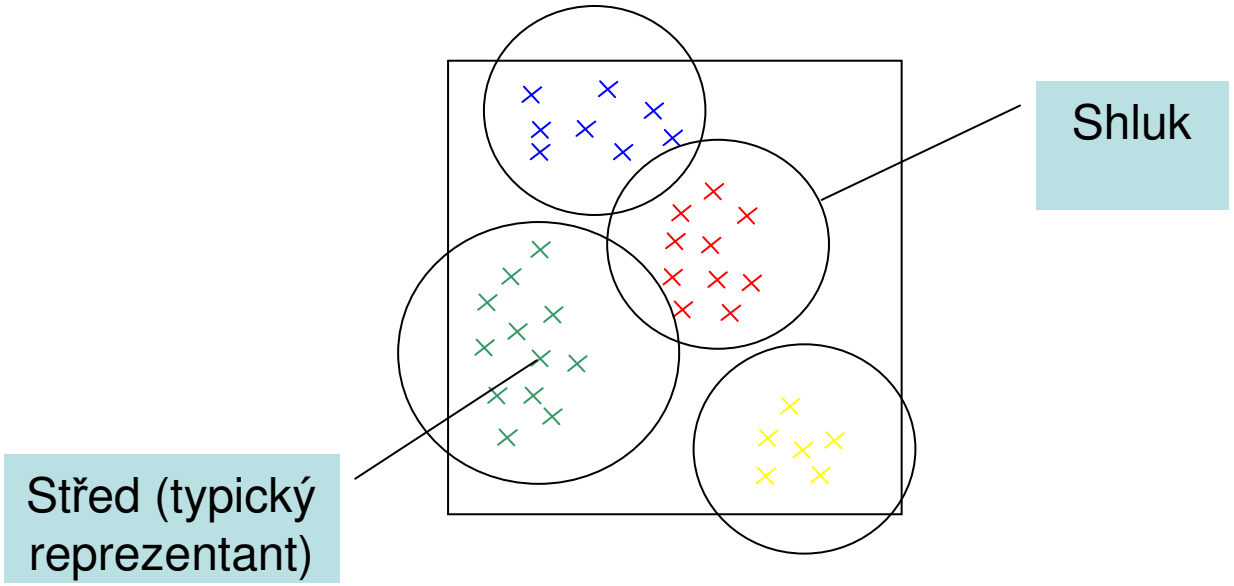
Data pacientů s obdobnými příznaky nemoci budou podobná a budou tvořit shluky. Pacienti, ležící ve středu shluku reprezentují typického pacienta s danou formou nebo závažností nemoci.

Očekáváme, že najdeme vztah mezi výsledky vyšetření a úsudkem lékaře (když to lékař pozná musí takový vztah existovat).

Pokud nenajdeme vztah shluků ke zdraví pacientů, pak patrně bude chybná diagnostická metoda, která dostatečně nevypovídá o diagnostikované nemoci. Předpokládáme, že známe spolehlivě stav pacientů z úsudku lékaře, který používá jinou diagnostickou metodu.

Jiným příkladem pak bude zkoumání dat o zákaznících mobilních operátorů, kde nám může např. tato metoda pomoci nalézt novou cílovou skupinu zákazníků. Podobně se můžeme zaměřit na uživatele Internetu, e-maily (spam), hledání typického chování hackerů, apod.

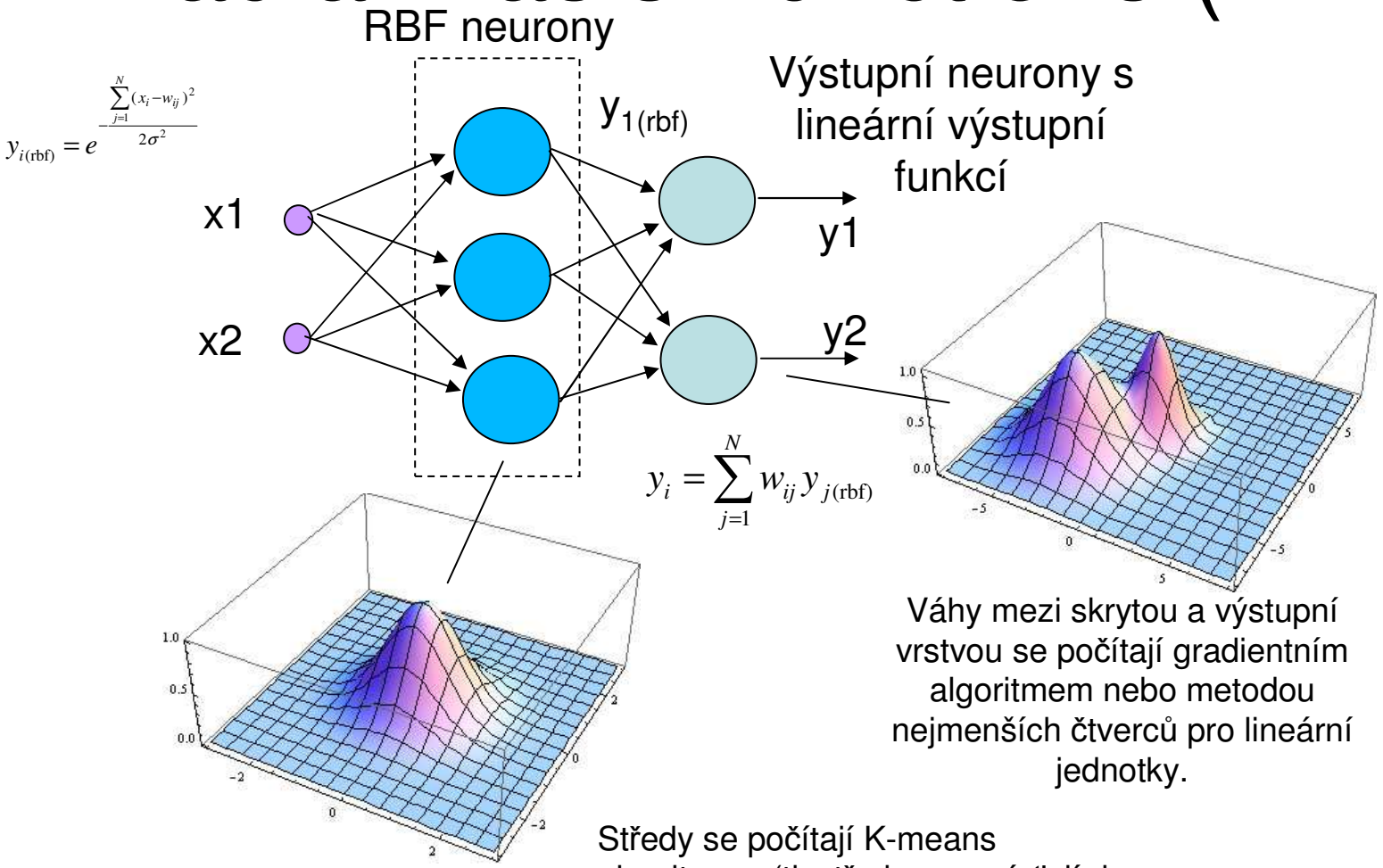
Příklad shluků v datech



K-means algoritmus

1. Odhadneme počet shluků
2. Náhodně vygenerujeme středy shluků. Počet středů je roven počtu shluků, střed je reprezentován vektorem s stejnou dimenzí, jako je dimenze vektorů, které tvoří shlukovaná data.
3. Určíme, k jakému shluku patří jaký vektor z dat. Daný vektor patří do toho shluku, k jehož středu je nejblíže (minimální Euclideovská vzdálenost od daného středu)
4. Z dat, která náleží shluku vypočteme nový střed (postupně po složkách průměr přes všechny vektory shluku) a nahradíme jím původní střed shluku. Toto provedeme pro všechny shluky
5. Body 3 a 4 opakujeme do ustálení (tj. do doby, kdy se středy posunou o méně než je zadaná hodnota)
6. Ukončíme algoritmus

Radial Basis Functions (RBF)



Váhy mezi skrytou a výstupní vrstvou se počítají gradientním algoritmem nebo metodou nejmenších čtverců pro lineární jednotky.

Středy se počítají K-means algoritmem (tj. středy se umísťují do středů shluků). Parametr σ se určí jako rozptyl shluku.

Neuronové sítě a výpočetní inteligence

Neuronové sítě

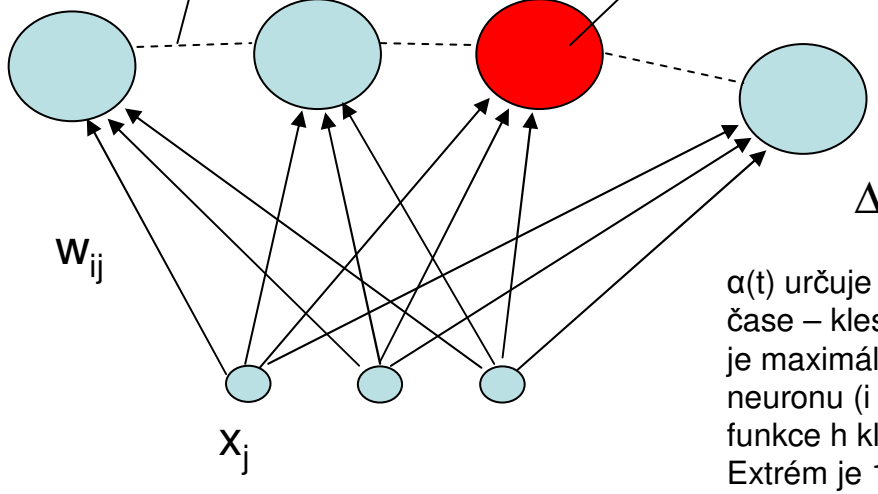
Miroslav Skrbek ©2010

Samoorganizující se mapy (SOM)

- SOM se mnohdy nazývá dle autora Kohonenova síť
- Reprezentuje neuronovou sít učenou bez učitele
- Provádí shlukovou analýzu
- Oproti K-means algoritmu není třeba předem znát počet shluků

SOM

$$y_i = \sum_{j=1}^{N} (x_j - w_{ij})^2$$



$$\Delta w_{ij} = \alpha(t)(x_j - w_{ij})h(i,b)$$

α(t) určuje rychlost učení (je proměnná v čase – klesá do nuly). h(i,b) je funkce, která je maximální pokud i je index vítězného neuronu (i = b), pokud je to jiný neuron, pak funkce h klesá se vzdáleností od vítěze. Extrém je 1 pro i=b, 0 jinak.

Vybavování:

Na vstupy předložíme vzor a vypočteme odezvy. Vítěz (neuron s nejmenší odezvou) je reprezentantem shluku, kam vzor patří.

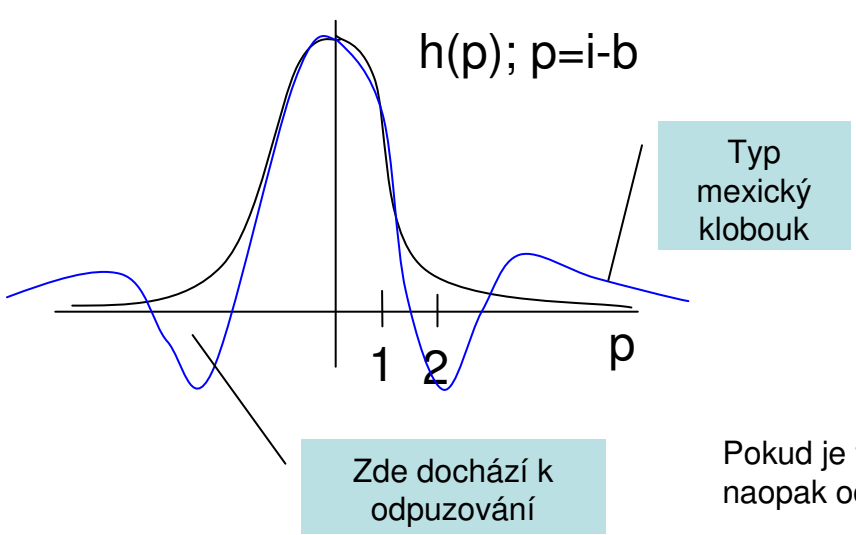
Učení: na vstupy předložíme vzor a určíme vítěze. Váhy vítěze posuneme o kousek blíže k učenému vektoru. Kromě vítěze upravíme sousední neurony (typicky čím dále, tím méně). Postup opakujeme do ustálení (α(t) = 0).

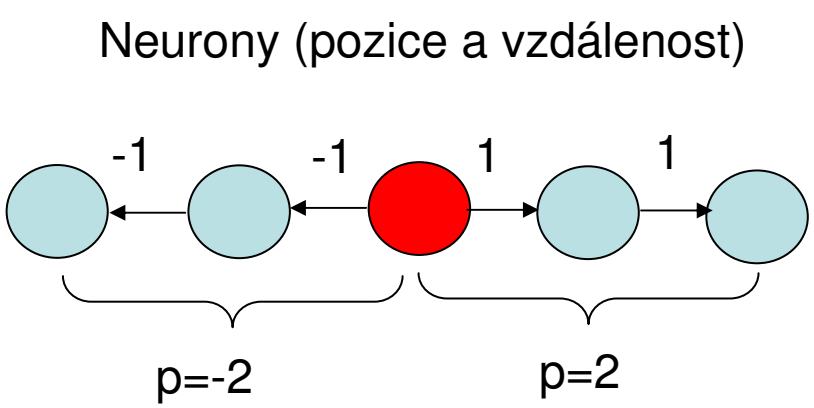
Algoritmus SOM

1. Inicializujeme váhy neuronů náhodnými hodnotami a nastavíme počáteční hodnotu koeficientu $\alpha(t)$.
2. Vybereme vektor z učící množiny a přiložíme na vstupy sítě
3. Vypočteme výstupy neuronů (vzorec pro y_i, viz. minulý slide)
4. Zjistíme neuron s nejmenší odezvou (to je reprezentant pro vstupní vektor)
5. Pro každý neuron upravíme váhu o Δwij (viz. minulý slide)
6. Pokud jsme nevyčerpali učící množinu, pokračujeme bodem 2
7. Snížíme koeficient $\alpha(t)$. Pokud je $\alpha(t) > 0$, pokračujeme bodem 2, opět od počátku učící množiny.

Funkce h(i,b)

Funkce h(i,b) jak se budou měnit váhy neuronů blízkých vítěznému neuronu. Jinými slovy, jak se budou prototypy reprezentované neurony posouvat směrem k učenému vzoru. Nejvíce bude přitahován vítězný neuron, neméně neuron, který je od něho nejvíce vzdálen.



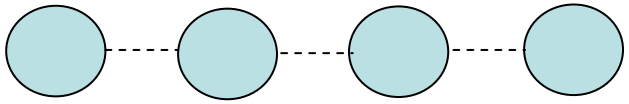


Pokud je funkce h(i,b) záporná, pak jsou nejbližší neurony naopak odpuzovány, aby se vytvořily bariery mezi shluky.

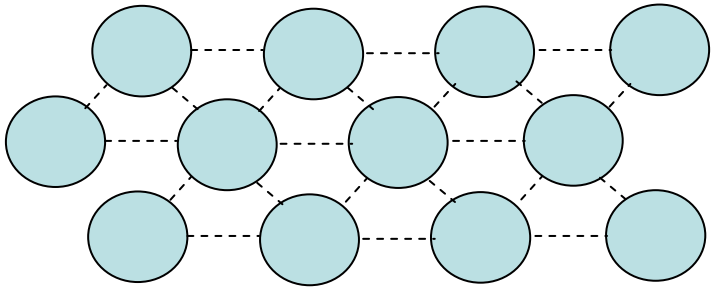
Při učení se začíná s plochou h(i,b), která se postupně s časem mění na strmější (zužuje se).

Uspořádání neuronů v SOM

Lineární mřížka



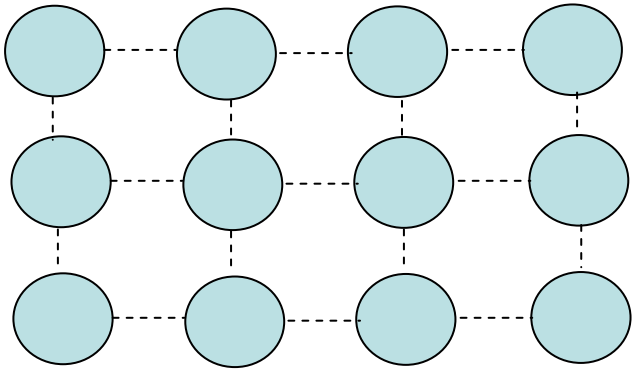
Hexagonální mřížka



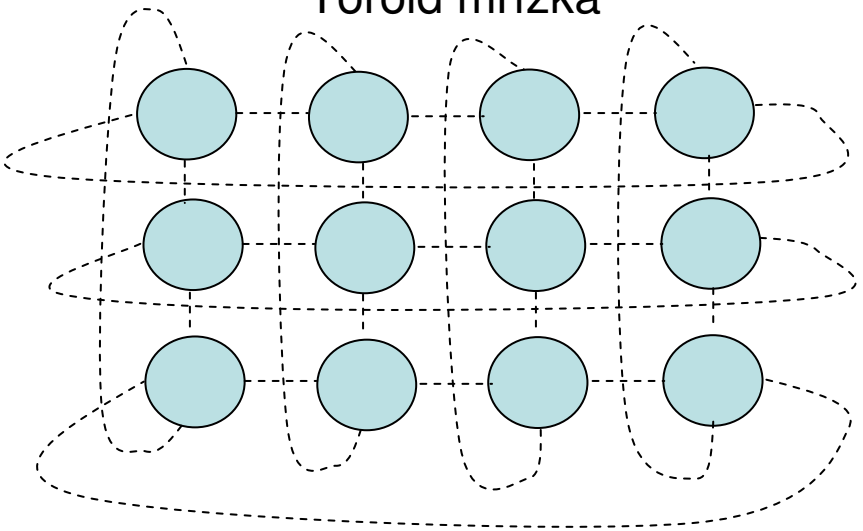
V hexagonální a lineární mřížce je vzdálenost mezi sousedními neurony 1 ve všech směrech.

Ve čtvercové a toroidní je vzdálenost rovnoběžne s osami 1 a uhlopříčně dva (Manhattanská vzdálenost) nebo odmocnina ze dvou (Eukleidovská).

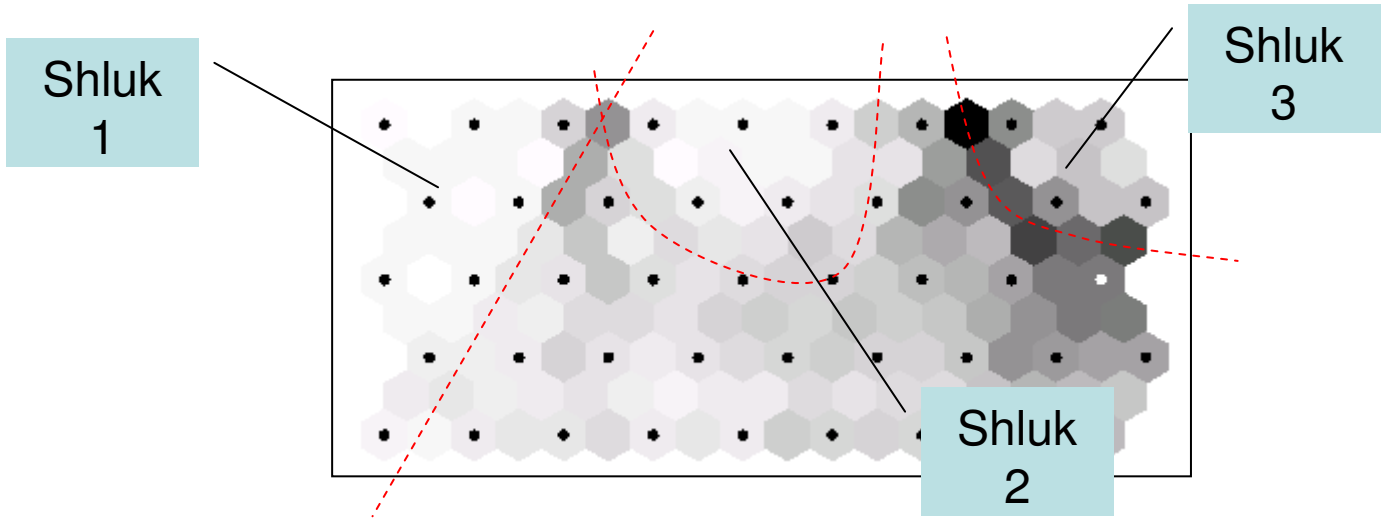
Čtvercová mřížka



Toroid mřížka



U-matice



Slouží k vizualizaci odezvy neuronové sítě SOM. Barva neuronů v plástu hexagonální mřížky indikuje průměrnou vzdálenost vah daného neuronu od jeho sousedů.

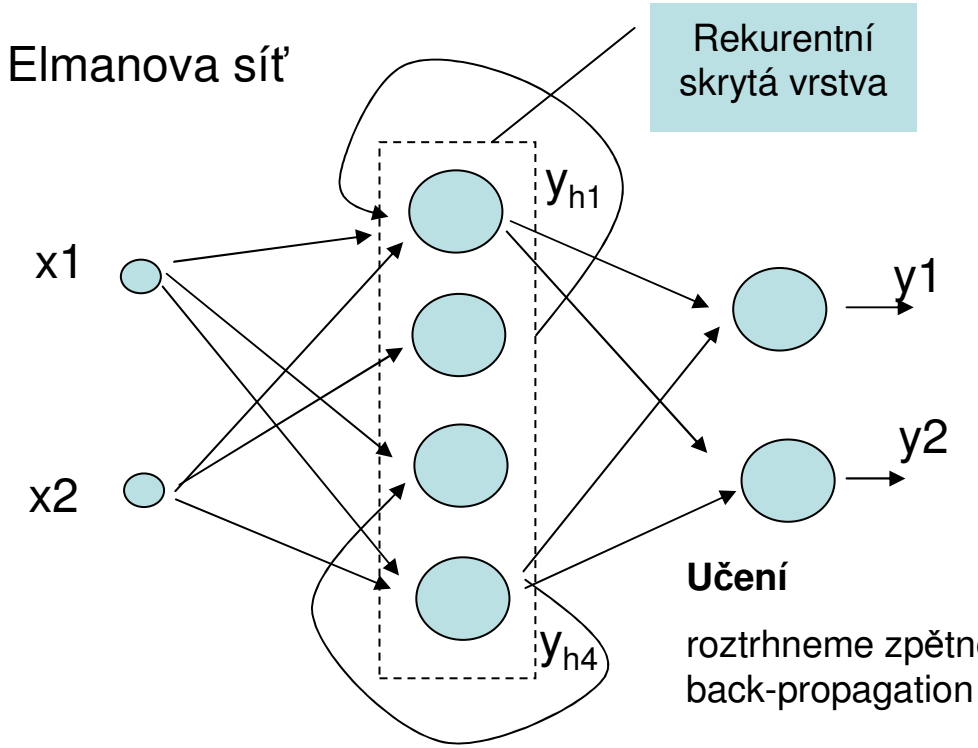
Sousední neurony s bílou barvou reprezentují shluk (mají podobné váhy, tedy reprezentují stejná vstupní data), tmavé brázdy vzdálených neuronů oddělují shluky. Z U-matice je proto možno vyčíst počet shluků v datech a neurony, které je reprezentují.

Obrázek převzat z: http://www.cis.hut.fi/jhollmen/dippa/node24.html

Neuronové sítě zohledňující čas

- Jsou to
  - rekurentní neuronové sítě
    - Naučí se chování konečného automatu
  - Time-delay-neural-networks (TDNN)
    - Výstup generuje na základě historie dat
- Vhodné pro zpracování časových řad
  - Časová řada je posloupnost hodnot x_t, kde má význam t času. Stanovíme-li nějké $t=t_0$ jako současnost, pak $x_{t=t_0}$ je současná hodnota, $x_{t<t_0}$ jsou minulé hodnoty (historie) a $x_{t>t_0}$ jsou hodnoty budoucí
- Umí se učit sekvence

Rekurentní neuronové sítě

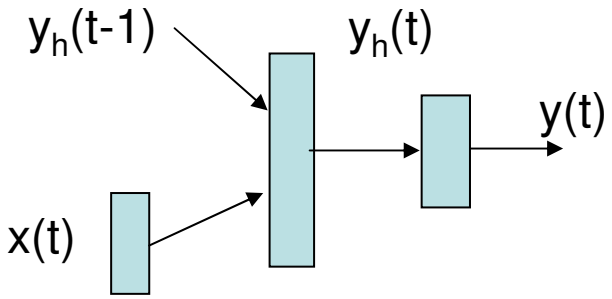


Vybavení neuronové sítě probíhá tak, že nastavíme hodnoty výstupu rekurentních neuronů na nulu a pak na vstup přikládáme postupně vektory a provedíme výpočet y. Pro hodnoty ve zpětných vazbách použijeme hodnoty z předešlého kroku. Odezva sítě tedy nezávisí jen na vstupech, ale taky na předchozím stavu sítě, tj. předchozích hodnotách ve skryté vrstvě.

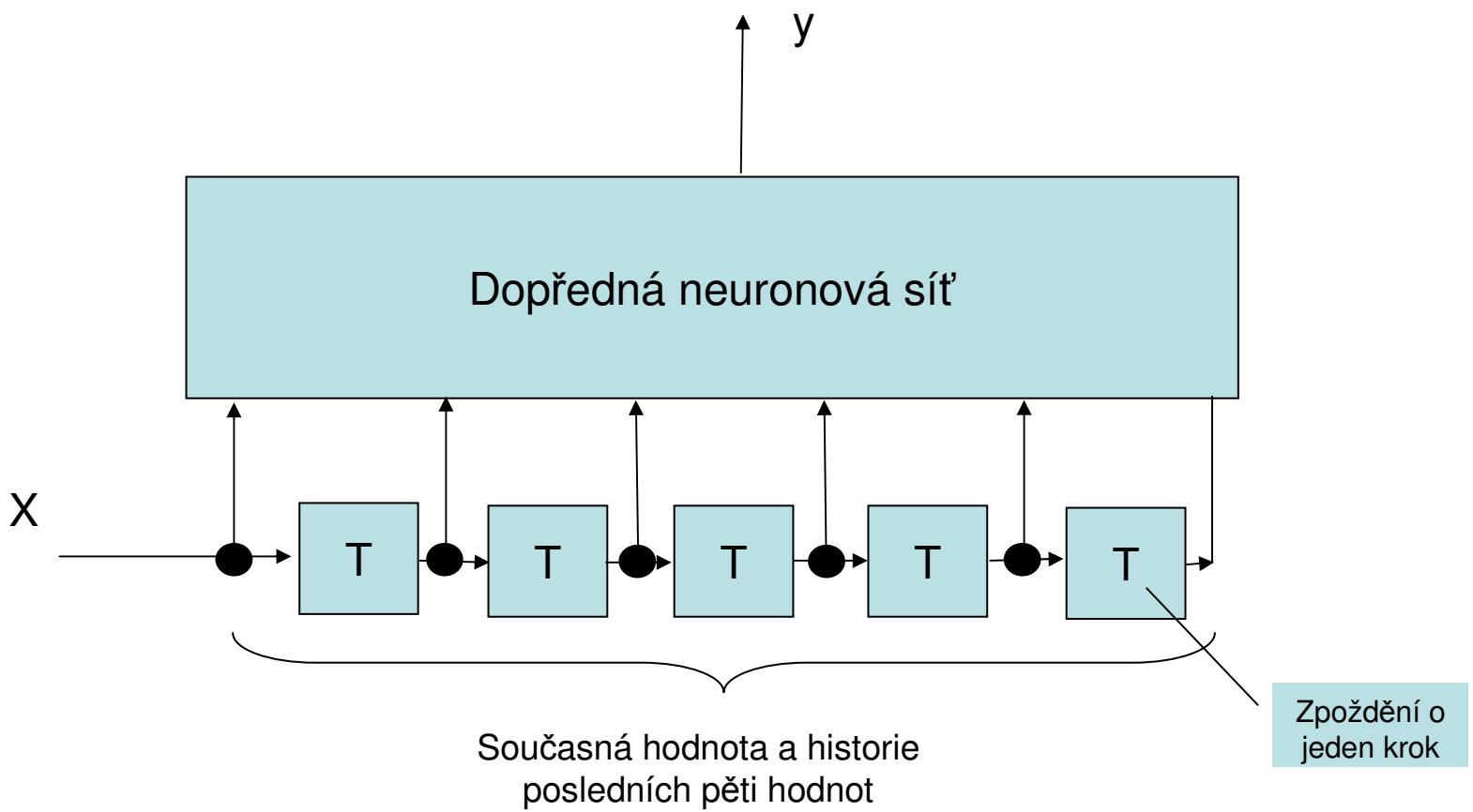
Učení

roztrhneme zpětné vazby a učíme algoritmem back-propagation

Jordanova síť je modifikovaná Elmanova síť, která nemá výstupní vrstvu neuronů a výstupy se berou rovnou z rekurentní vrstvy.



Síť TDNN


y
Dopředná neuronová síť
X
T
T
T
T
T
Zpoždění o jeden krok
Současná hodnota a historie posledních pěti hodnot

Neuronové sítě a výpočetní inteligence

Neuronové sítě

Miroslav Skrbek ©2010

Fuzzy Logika

Fuzzy logika byla odvozena teorie fuzzy množin (Lotfi A. Zadeh, 1965)

Fuzzy množina je rozšířením klasických množin o spojitou hodnotu příslušnosti, přičemž prvek množiny může mít nenulovou příslušnost k více fuzzy množinám.

Fuzzy znamená v angličtině neostrý, neurčitý, rozmazaný

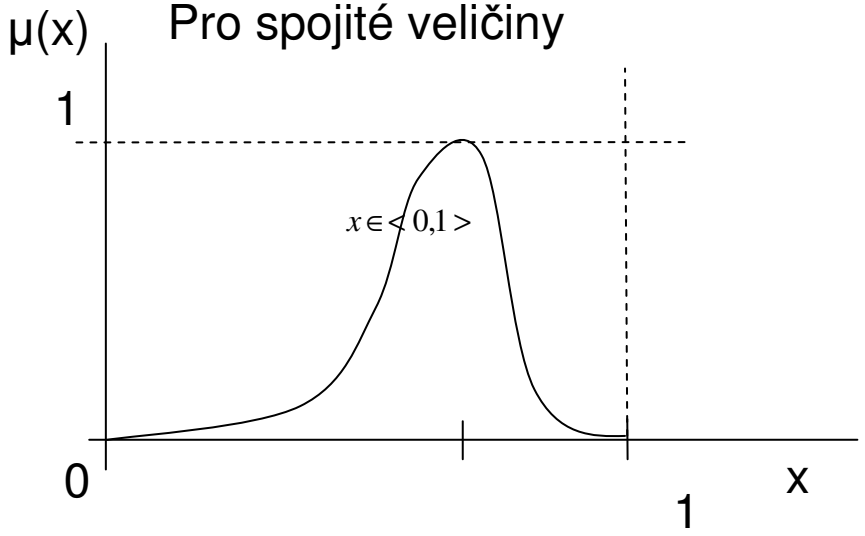
Klasická množina: prvek (x) do množiny patří, prvek do množiny nepatří

Fuzzy množina: prvek do množiny patří s mírou příslušnosti m (např. m=0.6).

Funkce příslušnosti

Funkce příslušnosti je zobrazením množiny prvků do množiny měr příslušností k množině. Označuje ji μ(x).

Pro spojité veličiny



Pro diskrétní veličiny

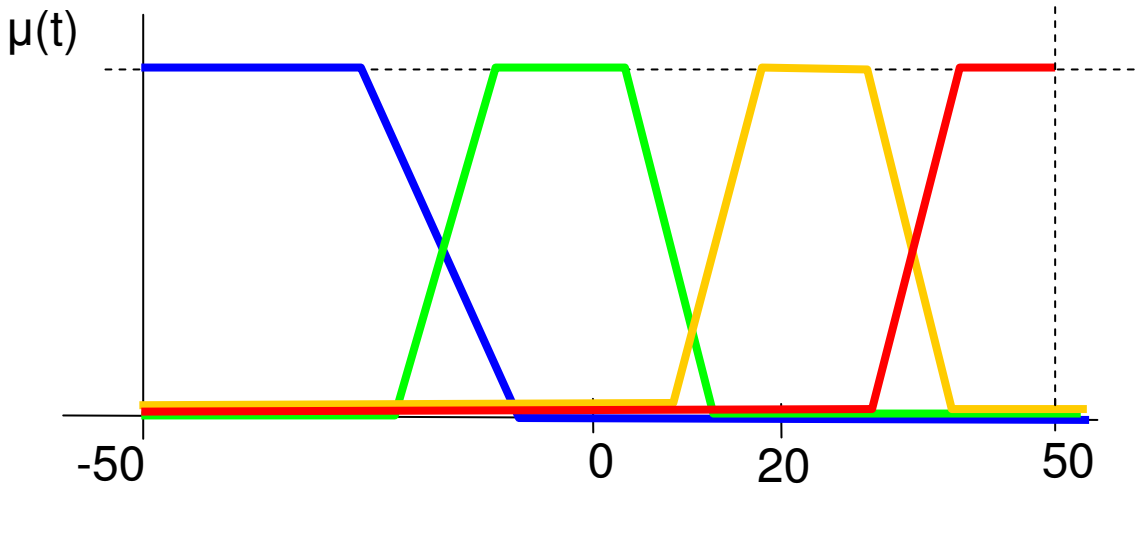
x	μ(x)
A	0,1
B	0,3
C	0,8
D	0,3

Příklad funkce příslušnosti

Předpokládejme veličinu: teplota (myšleno venkovní teplota vzduchu)

Pro tuto veličinu budeme definovat fuzzy množiny:

velká zima, zima, teplo, horko.

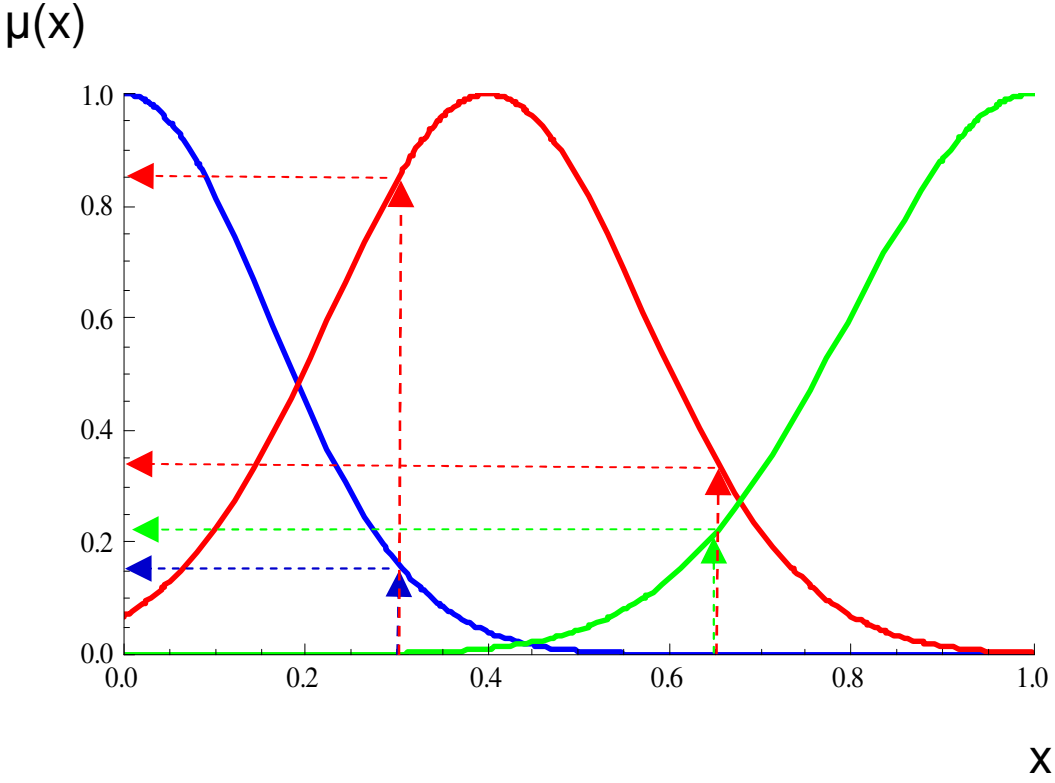


Funkce příslušnosti pro jednotlivé množiny odlišeny v grafu barevně

Úloha

Mějme tři fuzzy množiny A, B, C a k nim funkce příslušnosti zadané analyticky:

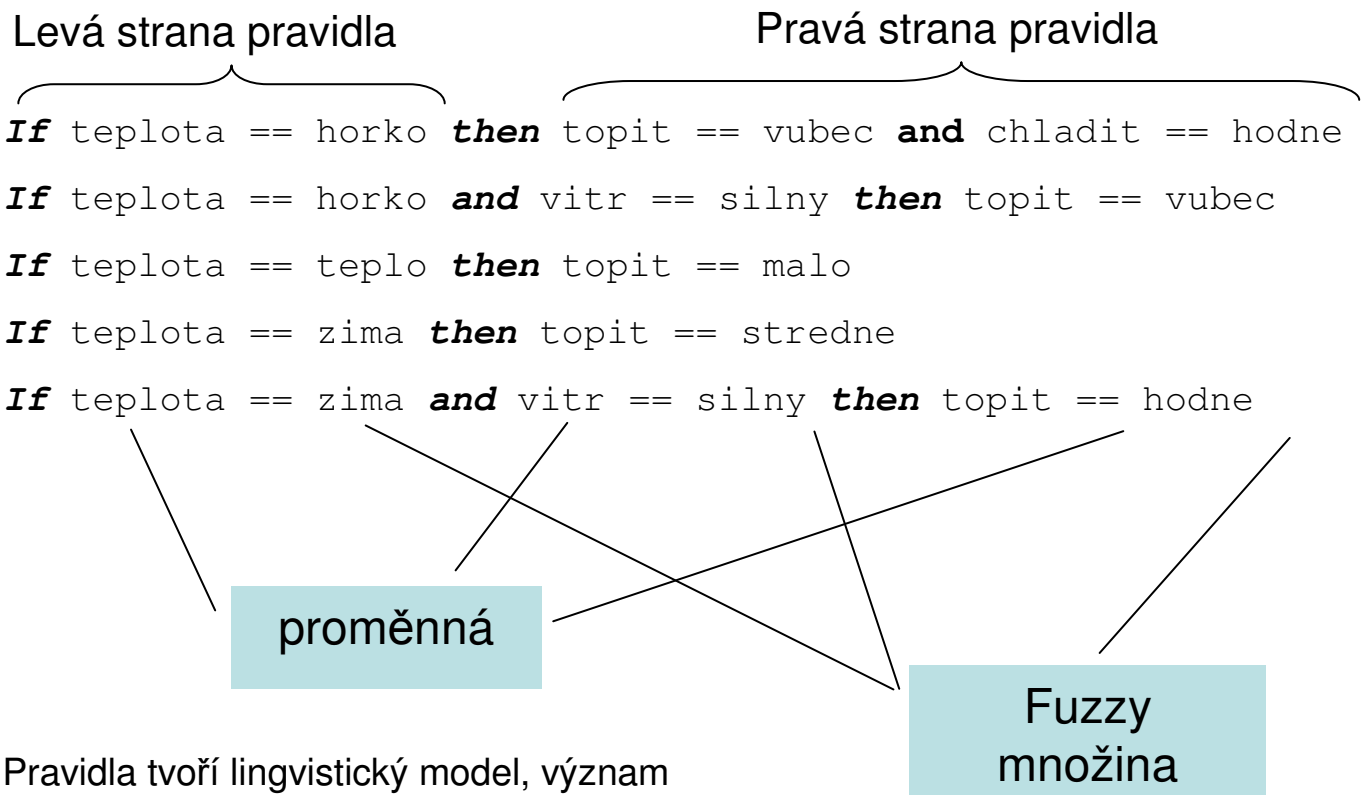
$$\mu_A(x) = e^{-\frac{x^2}{0,05}}, \mu_B(x) = e^{-\frac{(x-0,4)^2}{0,06}}, \mu_C(x) = e^{-\frac{(x-1)^2}{0,08}}$$



Doplňte prázdná pole

x	$\mu_A(x)$	$\mu_B(x)$	$\mu_C(x)$
0	1		$3,7.10^{-6}$
0,3	0,165	0,85	
0,45	0,017		
0,65		0,35	0,21
0,8			

Fuzzy pravidla



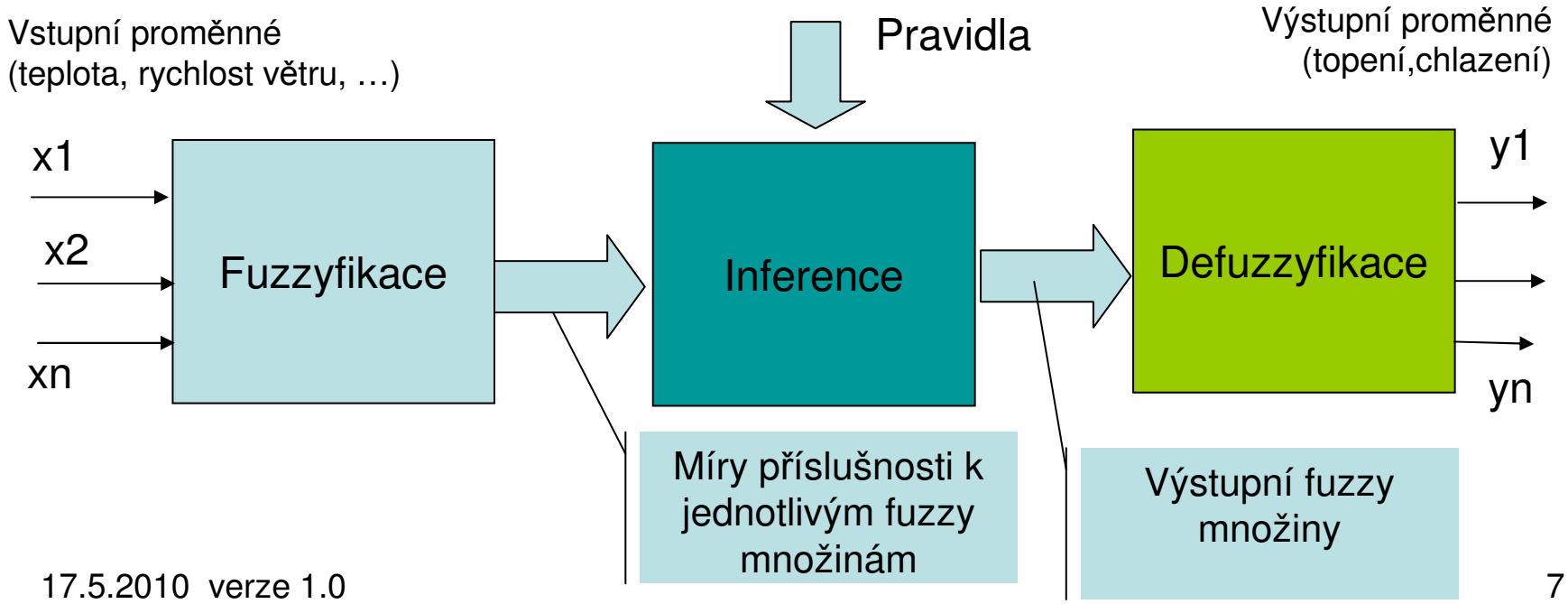
Pravidla tvoří lingvistický model, význam jednotlivých termínů (zima, teplo, malo, …) vyjadřují funkce příslušnosti.

Pojmy fuzzyfikace, inference a defuzzyfikace

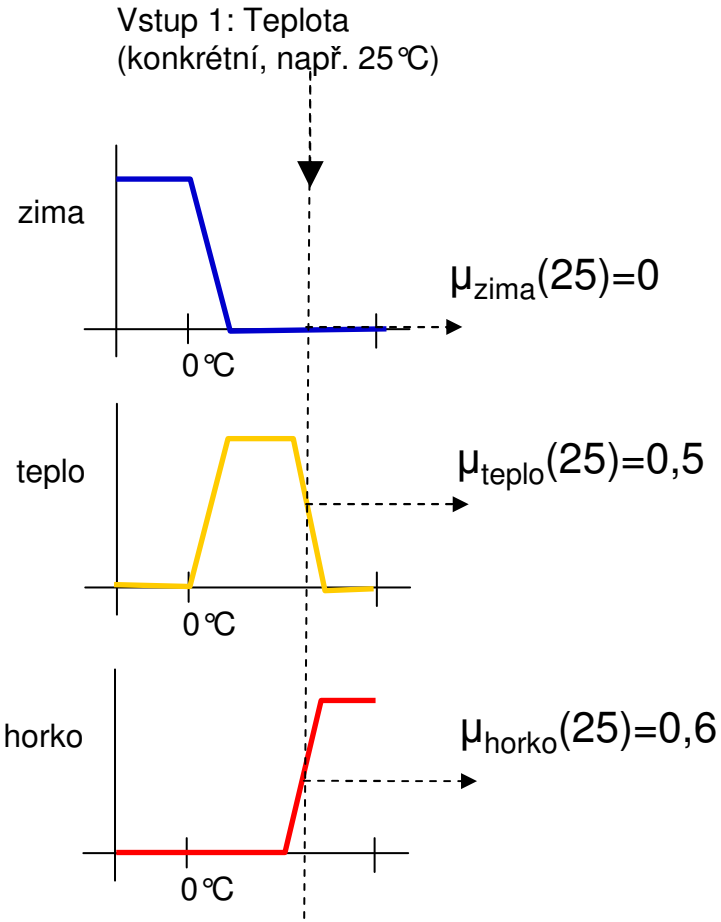
Fuzzyfikace – převod hodnot proměnných na míry příslušnosti

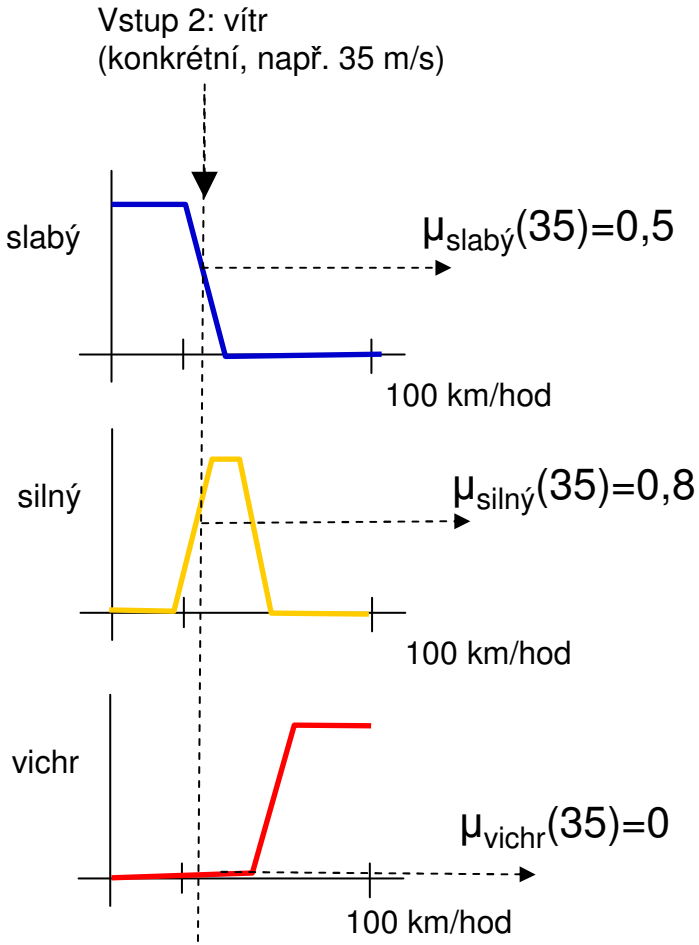
Fuzzy inference – vyhodnocení fuzzy pravidel a získání výstupních fuzzy množin

Defuzzyfikace – převod výstupních fuzzy množin na výstupní proměnné

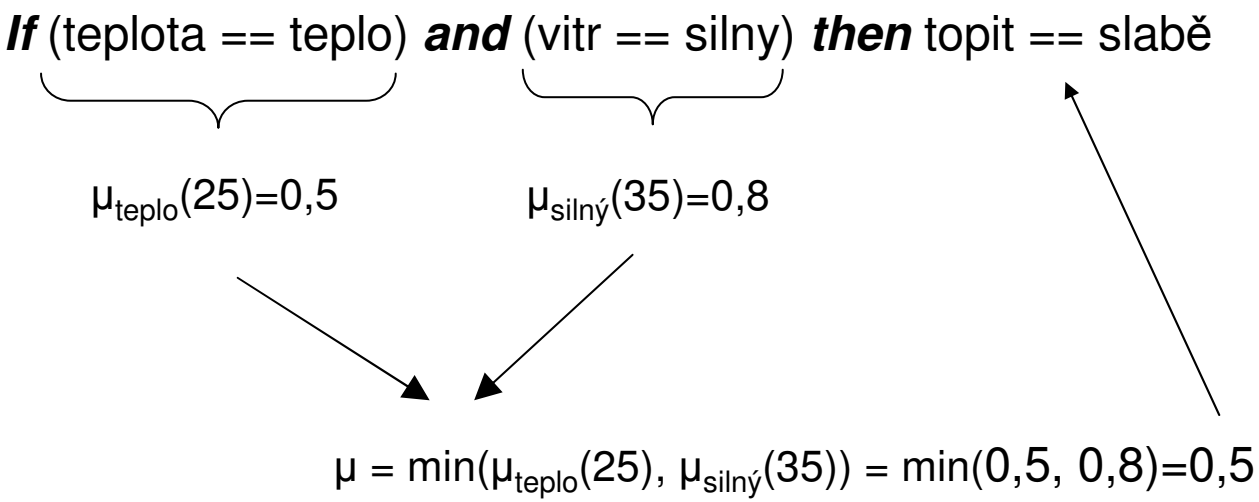


Příklad fuzzyfikace





Inference



Pokud je použita logická spojka ***or***, pak se použije funkce maximum.

Pokud se tatáž výstupní proměnná vyskytuje ve více pravidlech, pak se fuzzy množiny připadající na jednotlivé výskyty této proměnné vyhodnocují tak, jako by byly spojeny logickou spojkou **or** (počítá se maximum).

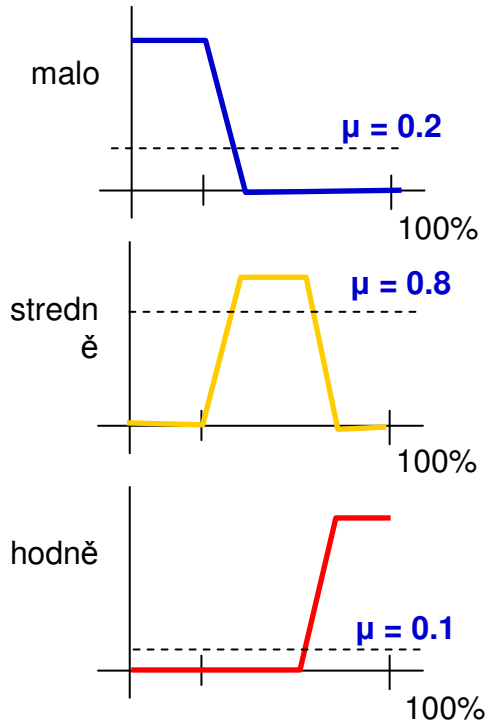
Vyhodnocení pravých stran pravidel

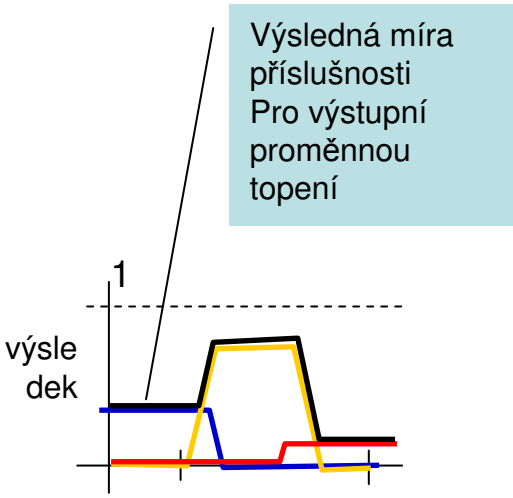
μ = 0.2
… then topeni == malo

μ = 0.8
… then topeni ==středně

μ = 0.1
… then topeni ==hodně

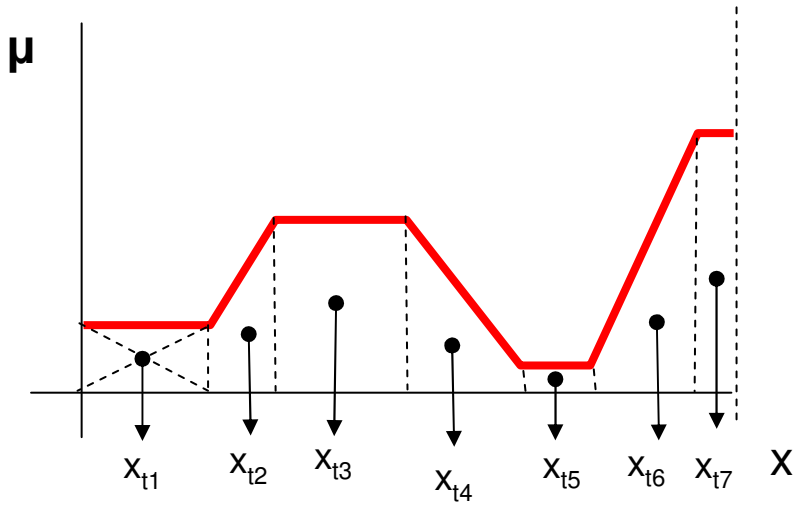
Pozn.: toto je nutno provést pro všechny výstupní proměnné





Defuzzyfikace

Defuzzyfikace se provádí nejčastěji výpočtem těžiště dané fuzzy množiny



Obecný vzorec

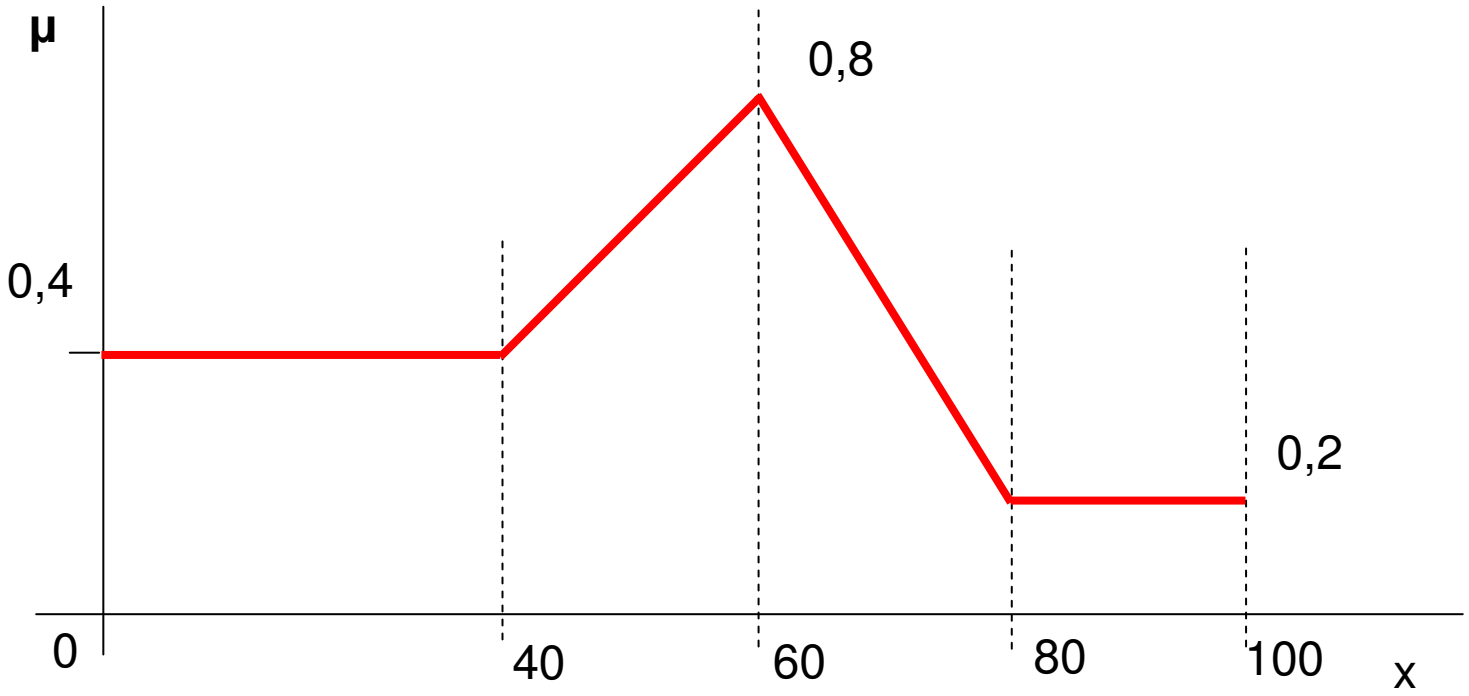
$$x_t = \frac{\int_{x_{\min}}^{x_{\max}} x.\mu(x)dx}{\int_{x_{\min}}^{x_{\max}} \mu(x)dx}$$

Složitý tvar rozložíme na jednodušší geometrické útvary a spočteme dílčí těžiště. Pak použijeme vztah

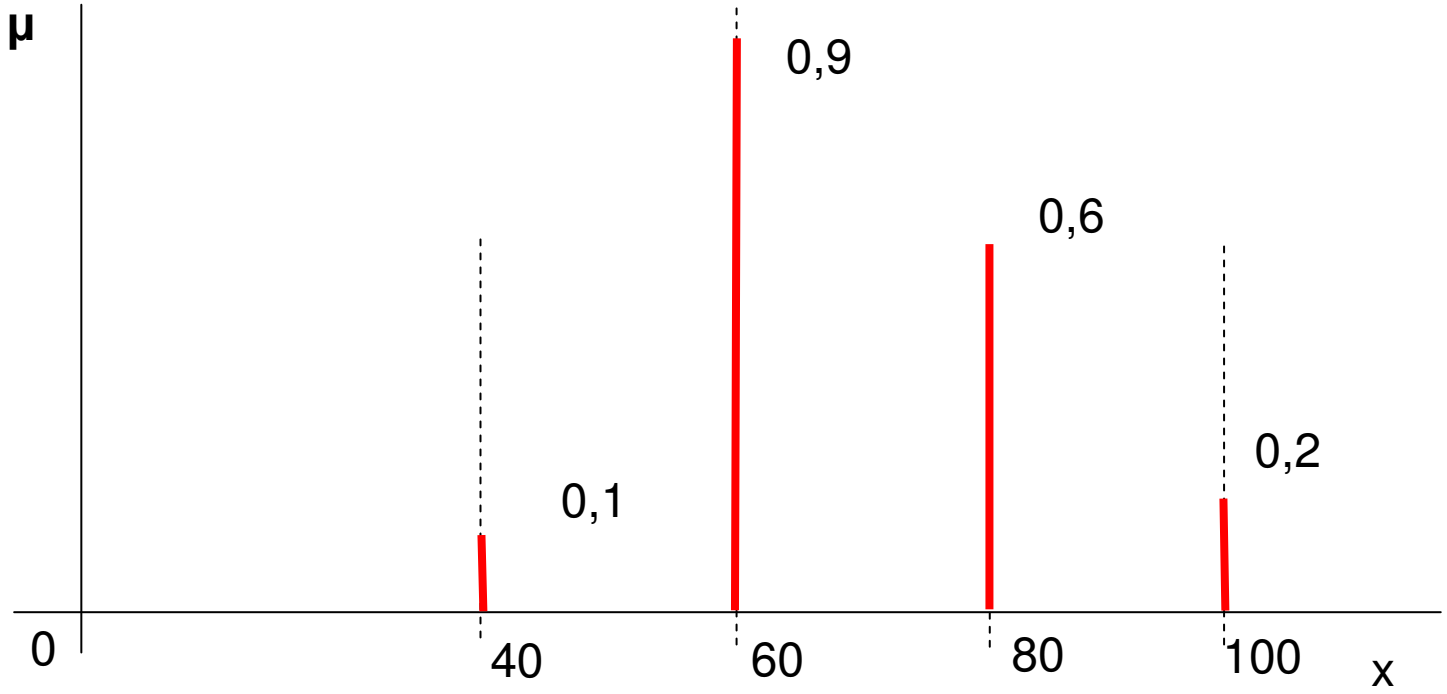
$$x_t = \frac{\sum_{k=1}^{N} x_{t_k}.m_k}{\sum_{k=1}^{N} m_k}$$

Kde m_k je plocha geometrického útvaru, který odpovídá těžišti x_{tk}

Příklad: spočtěte hodnotu pro zadanou fuzzy množinu výstupní veličiny



Příklad: spočtěte hodnotu pro zadanou fuzzy množinu výstupní veličiny



Genetický algoritmus

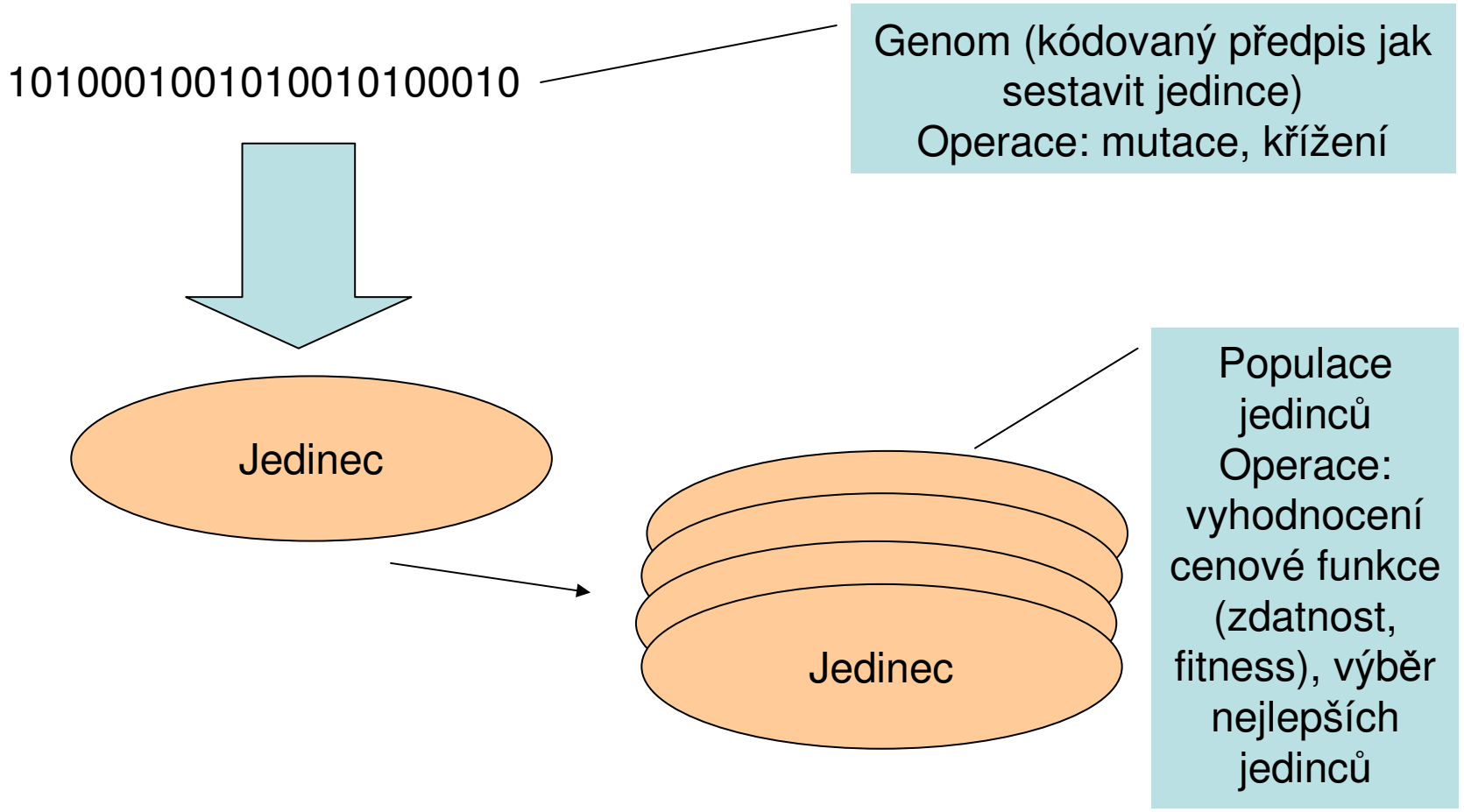
- Algoritmus inspirovaný evolučními teoriemi živočichů
- Robustní heuristický optimalizační algoritmus
- Náročný na výkon a systémové zdroje počítače

Neuronové sítě a výpočetní inteligence

Neuronové sítě

Miroslav Skrbek ©2010

Základní pojmy



Selekce

- Výběr jedinců do další populace
- Často používaná metoda je ruletová selekce (roulette wheel selection). Jedinci se vybírají do nové populace náhodně s pravděpodobností danou jejich zdatností (vyšší zdatnost => vyšší pravděpodobnost)
- Jinou metodou je převzít do nové populace určité procento jedinců s největší zdatností

Mutace

Mutace je náhodná změna genomu. Jedná se o unární operaci nad genomem.

Pro binárně zakódovaný genom je mutace realizována jako změna náhodně vybraného bitu (případně bitů) v genomu.

Původní genom: **01000100101001**

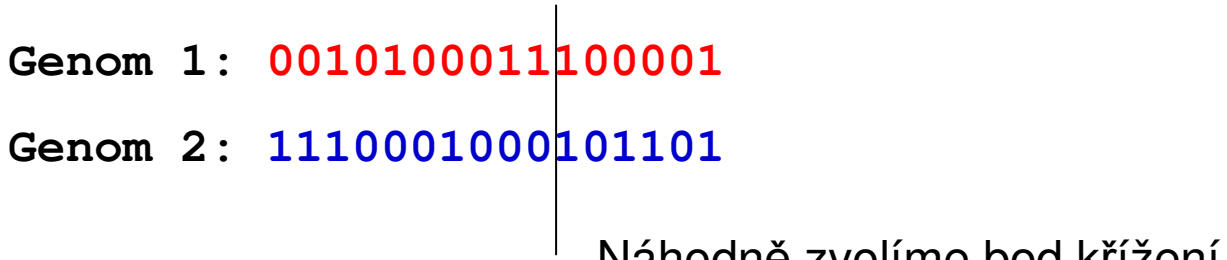
Generátor náhodných čísel generuje 5, změníme tedy pátý bit (bity počítáme z leva, nejlevější je bit 1):

Nový genom: **01001100101001**

Mutace může také např. prodloužit genom přidáním bitu.

Křížení

Křížení je rekombinace genetické informace dvou jedinců. Jedná se o binární operaci.



Genom : 1110001000100001

Jsou možná i více bodová křížení (více bodů, kde rozdělíme genom)

Základní algoritmus

1. Náhodně vygenerujeme počáteční populaci jedinců
2. Vyhodnotíme zdatnost u všech jedinců v populaci
3. Část jedinců s nejhorší zdatností odstraníme
4. Populaci doplníme o další jedince (křížení, mutace, reprodukce)
5. Pokračujeme bodem 2, pokud není splněna podmínka ukončení
6. Vezmi jedince s nejlepší zdatností jako výsledné řešení

Problém batohu

Máme následující předměty

Předmět	Hmotnost [kg]	Cena [Kč]
A	2	10
B	1	50
C	5	5
D	4	100

Jaké předměty mohu odnést, unesu-li max. 6 kg

Příklad genomu:

A**B**C**D**
Genom:**0101**

V batohu je B a D

Zdatnost je: $cena_B + cena_D$
= 50 + 100 = 150

Příklad populace:

Genom 1:**0101** **zdatnost: 150**
Genom 2:**1101** **zdatnost: 0**
Genom 3:**1001** **zdatnost: 110**

Pozn: Genom 2: má zdatnost nula, protože překračuje limit 6kg.

Pojem fenotyp a genotyp

Genotyp je soubor všech genetických informací (pro účely GA lze chápat genotyp = genom jako u jednobuněčných živočichů)

Fenotyp je soubor znaků chování jedince. Tyto znaky jsou projevem určité podmnožiny genů (vloh) z genomu.

Pokud například šlechtíme neuronovou síť pro řízení robota, je genotypem informace popisující, jak neuronovou sít vytvořit (genom typicky popisuje topologii sítě) a některé parametry.

Fenotypem pak jsou schopnosti neuronové sítě (vytvořené na základě konkrétního genomu) a řídit robota tak, aby projel překážkovou dráhu. Zdatností je v tomto případě úspěšnost projetí dráhy.

Výhody a nevýhody GA

- Výhody
  - Prověřuje několik řešení najednou (jedno řešení = jeden jedinec v populaci)
  - Na počátku je celá populace rozprostřena po prostoru řešení, tak je velká pravděpodobnost, že některý padne do blízkosti globálního minima
  - Výběr jedinců do populace se děje na základě výpočtu hodnotící funkce bez žádných specifických požadavků, proto je vhodný pro širokou škálu optimalizačních úloh
  - Mezi genotypem a fenotypem může být matematicky těžko popsatelný vztah
- Nevýhody
  - vysoká paměťová a operační náročnost
  - Vyhodnocení zdatnosti je nutné provést vždy pro všechny jedince

Diverzita populace

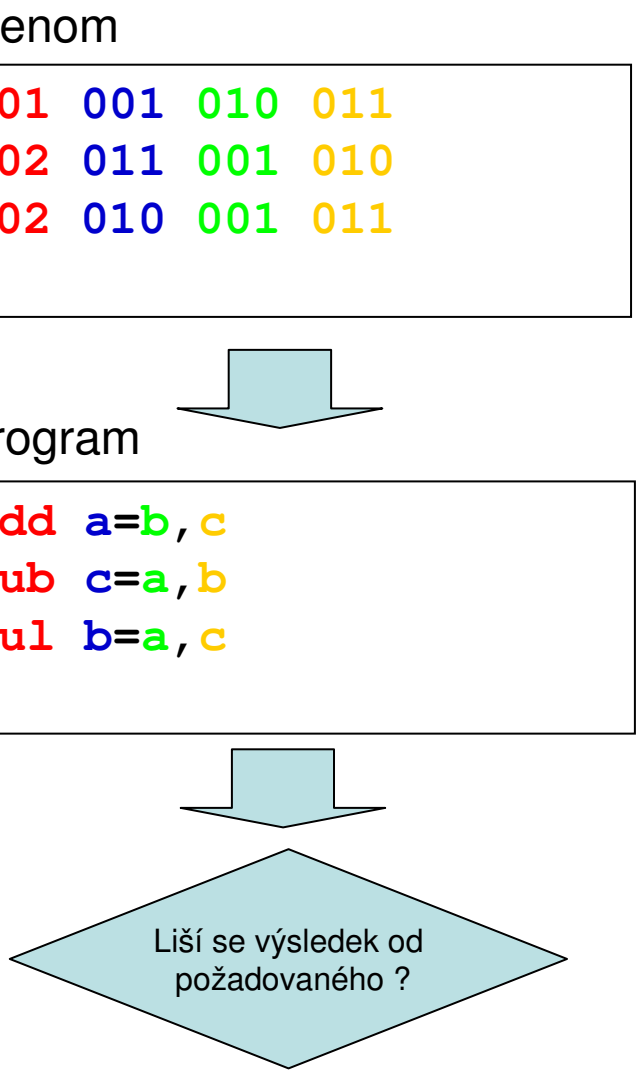
- V některých případech se může celá populace brzy soustředit do oblasti blízké některému lokálnímu maximu fitness funkce a tím zabránit nenalezení globálního maxima. Tomu se říká ***předčasná konvergence***. V tomto případě jsou všechny jedinci většinou genotypově podobní
- Tomu lze zabránit zajištěním diverzity populace
  - Nechat přežít jedince i s nižší zdatností
  - Zvýšit mutaci na genotypově podobných jedincích nebo podobnost s ostatními jedinci započítávat do zdatnosti (větší podobnost, menší zdatnost)
  - Zavést do populace druhy a nenechat je křížit mezi sebou nebo křížení omezit

Diferenciální evoluce

- Genetický algoritmu se speciálním typem křížení jedinců
- Vhodný pro spojitou optimalizaci (genom obsahuje číselné hodnoty)
- Křížení
  - Pro jedince x náhodně vybereme z populace jedince a, b, c.
  - Pro náhodně vybranou dimenzi genomu i určíme $x_i=a_i+k(b_i - c_i)$, kde k je váha diference mezi jedinci
  - Tak vytvoříme nového jedince, kterého zařadíme do populace je tehdy, pokud jeho zdatnost je větší než je zdatnost x.
- Slovně řečeno: nového jedince vytvoříme ze základního jedince x tak, že jeho některé geny změníme na základě tří náhodně vybraných jedinců a,b,c tak, že jedinec a bude tvořit bázi, ke které přičte vážený rozdíl jedinců b,c.

Genetické programování

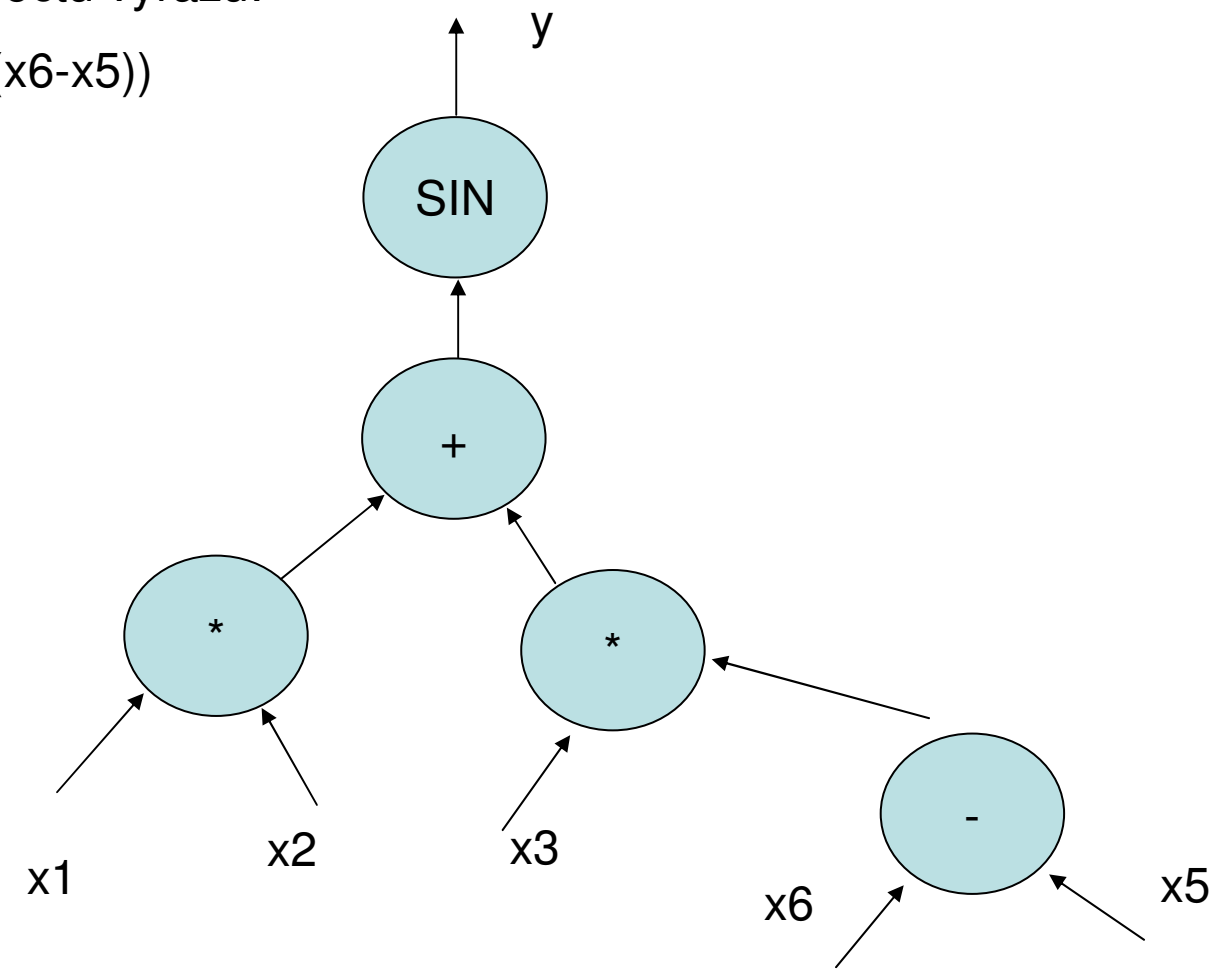
- Vytváření programu nebo dataflow grafu genetickým algoritmem
- Genom obsahuje instrukce výpočtu
- Zdatnost vyjadřuje rozdíl mezi výsledkem programu a požadovaným výsledkem. Menší rozdíl větší zdatnost
- Problémem je, že při mutaci nebo křížení může vzniknout mnoho nefunkčních řešení => navrhnout takové operátory křížení, které tento negativní jev potlačí. Např. operujeme nad dataflow grafem nikoliv přímo nad sekvencemi instrukcí.



Příklad dataflow grafu

Dataflow graf výpočtu výrazu:

y = sin(x1*x2+x3(x6-x5))



Genetický algoritmus pro učení neuronových sítí

- Používá se pro šlechtění topologie neuronové sítě ale i vah
- Při návrhu algoritmu je třeba zohlednit, že neuronová síť s dobrou topologií, ale špatně nastavenými vahami, může mít horší zdatnost, než neuronová síť s horší topologií, ale dobře nastavenými vahami.
  - Obvykle se šlechtí dvou úrovňově, nejprve topologie, pak se topologie zmrazí a nechá se čas algoritmu doladit váhy. Potom se teprve rozhodne o vyřazení jedince z populace
  - Kombinací algoritmů: topologie genetickým algoritmem, váhy gradientně (Backpropagation)

Neuronové sítě a výpočetní inteligence

Neuronové sítě

Miroslav Skrbek ©2010

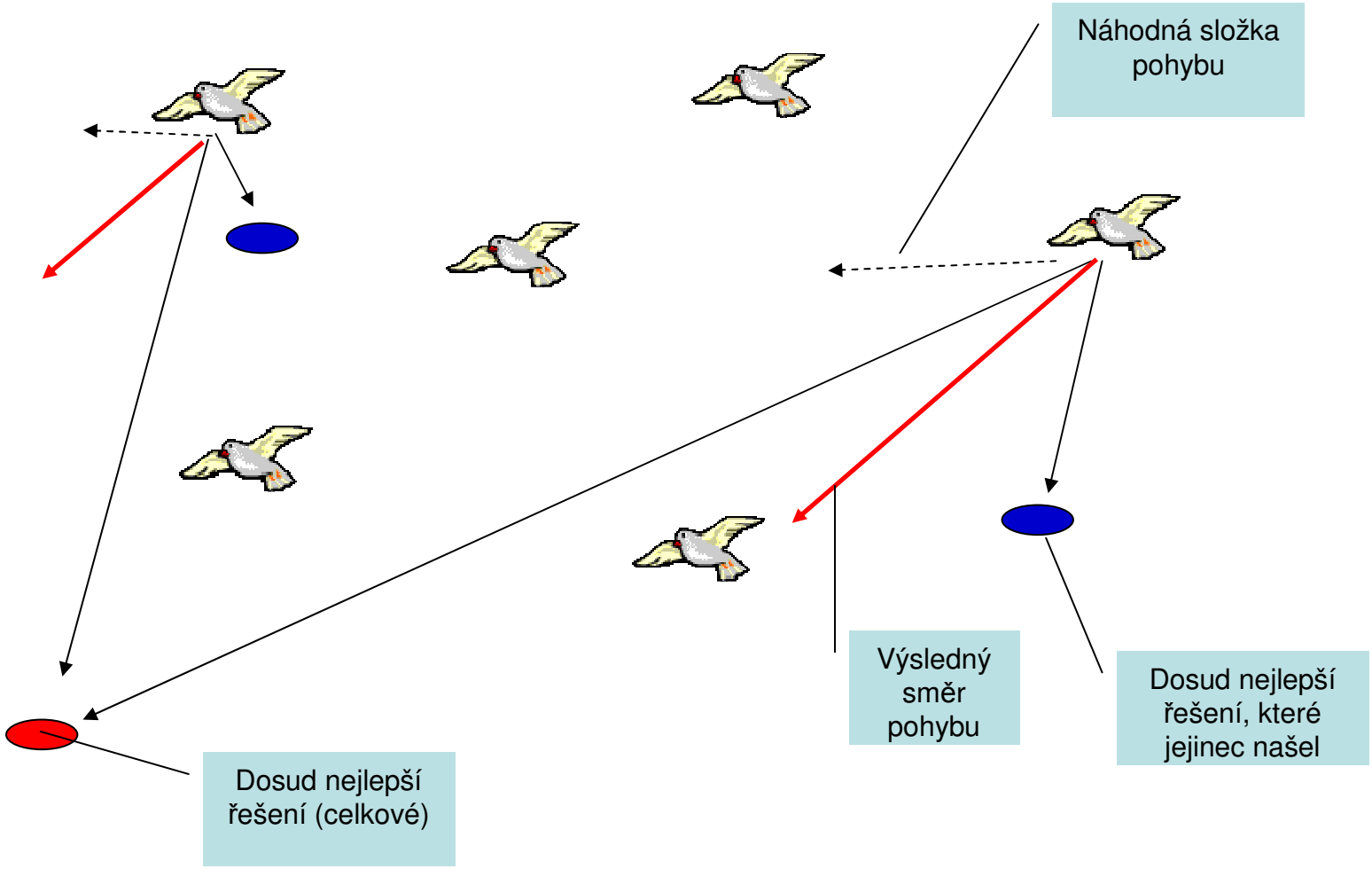
Particle swarm optimization

- Přírodou inspirovaný optimalizační algoritmus
- Modeluje chování hejn ptáků
- Vhodný pro spojitou optimalizaci
- Má řadu podobností s genetickým algoritmem

Popis algoritmu

- Základem je hejno částic (jedinců, ptáků)
- Každá částice se pohybuje po spojitém prostoru, kde hledáme řešení. Má tedy svoji rychlost a směr
- Každá částice vyhodnocuje fitness svého řešení v každém bodě prostoru, kde se vyskytuje a pamatuje si svoje nejlepší řešení (lbest)
- Pro všechny částice se udržuje nejlepší dosud nalezené řešení (gbest, to bude potom výsledným řešením po ukončení algoritmu)
- Algoritmus postupuje po krocích, ve kterých se vyhodnocuje nová pozice částic dle vztahu
  $\mathbf{x}(t+1)=\mathbf{x}(t)+\mathbf{v}(t)$, $\mathbf{v}(t)=\omega_p\mathbf{v}(t-1)+\varphi_p\mathbf{r}_p(\mathbf{p}_p-\mathbf{x}(t))+\varphi_g\mathbf{r}_g(\mathbf{g}-\mathbf{x}(t))$, kde $\mathbf{r}_p$ a $\mathbf{r}_g$ jsou náhodné vektory a ω_p, φ_p, φ_g jsou parametry
- Rychlost částice je ovlivněna vzdáleností částice od jejího nejlepšího řešení lbest (člen $\mathbf{p}_p-\mathbf{x}(t)$) a vzdáleností od globálního řešení gbest (člen $\mathbf{g}-\mathbf{x}(t)$). Jinými slovy částice je směrována jak k jejímu lokálnímu, tak ke globálnímu řešení, přičemž parametry φ_p, φ_g určují míru ovlivnění. Náhodné vektory $\mathbf{r}_p$ a $\mathbf{r}_g$ vnášejí do pohybu náhodnou složku. Parametr ω_p je mírou setrvačnosti částice

PSO

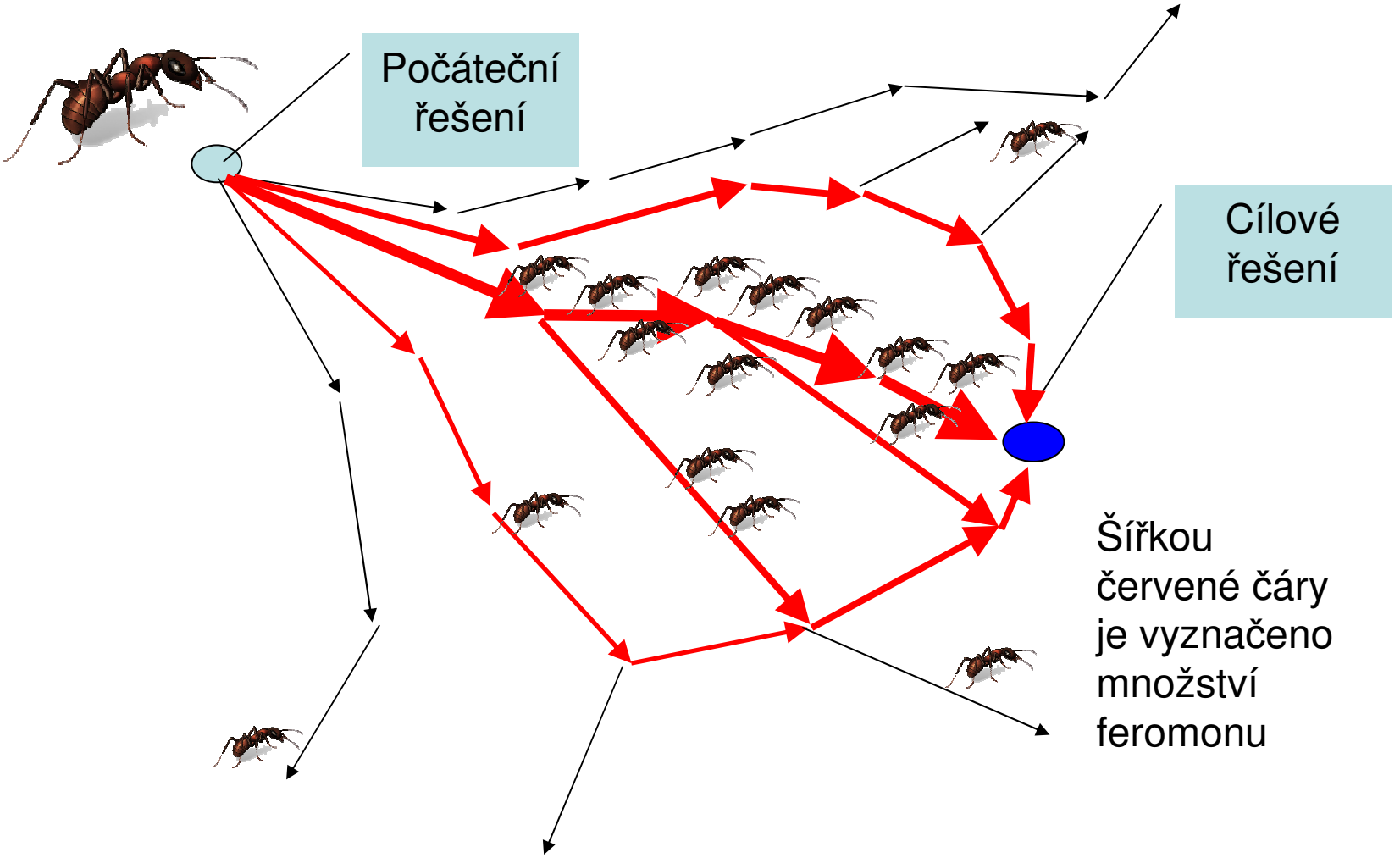

Náhodná složka pohybu
Výsledný směr pohybu
Dosud nejlepší řešení, které jejinec našel
Dosud nejlepší řešení (celkové)

Optimalizace mravenčími koloniemi

Ant Colony Optimization (ACO)

- Algoritmus optimalizace mravenčími koloniemi
- Mravenec vytváří řešení tím, že se v učitých krocích (typicky uzlech grafu) náhodně rozhoduje, kterou hranou se vydá. Představte si paket v síti (mravenec), který se rozhoduje, kterým portem se vydat k požadovanému cílovému počítači.
- Pravděpodobnost, kterou hranou se vydá je ovlivněna feromonem (těkavá látka, kterou mravenec klade).
- Feromon mravenec klade na hrany grafu, a to pokud našel cíl (tedy na zpáteční cestě). Klade ho o to víc kolik potravy našel. V případě příkladu s paketem by to byla doba, za kterou dosáhl cíle (čím kratší, tím více feromonu). Pak budou ostatní pakety chodit po feromonové cestě.
- Feromon v čase vyprchává, tzn. pokud některý mravenec nanese feromon a další ho nenásledují, protože našli lepší řešení, pak z cesty, která není navštěvována feromon vyprchá.
- Fermomon je určitou formou paměti (akumuluje zkušenost).Těkavost feromonu je určitou formou zapomínání. To umožňuje algoritmu se průběžně adaptovat na nové podmínky.

Mravenčí kolonie

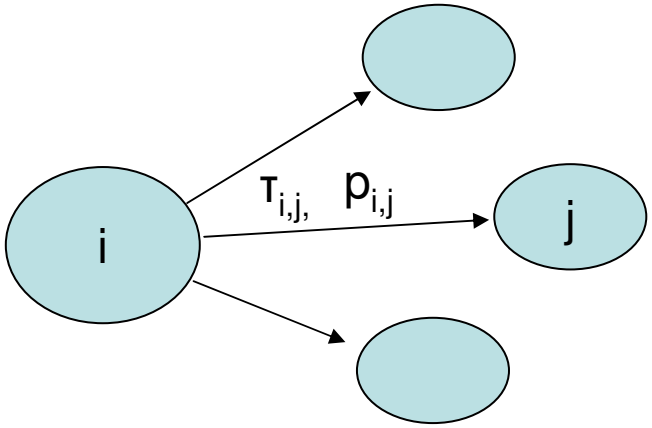


Popis algoritmu

$$p_{i,j} = \frac{\tau_{i,j}^{\alpha} \eta_{i,j}^{\beta}}{\sum_{j} \tau_{i,j}^{\alpha} \eta_{i,j}^{\beta}}$$

$\tau_{i,j}^{\alpha}$ ... množství feromonu na hraně i, j

$\eta_{i,j}^{\beta}$ ... dílčí cena řešení



Aktualizace feromonu

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \Delta\tau_{i,j}$$

Těkání feromonu

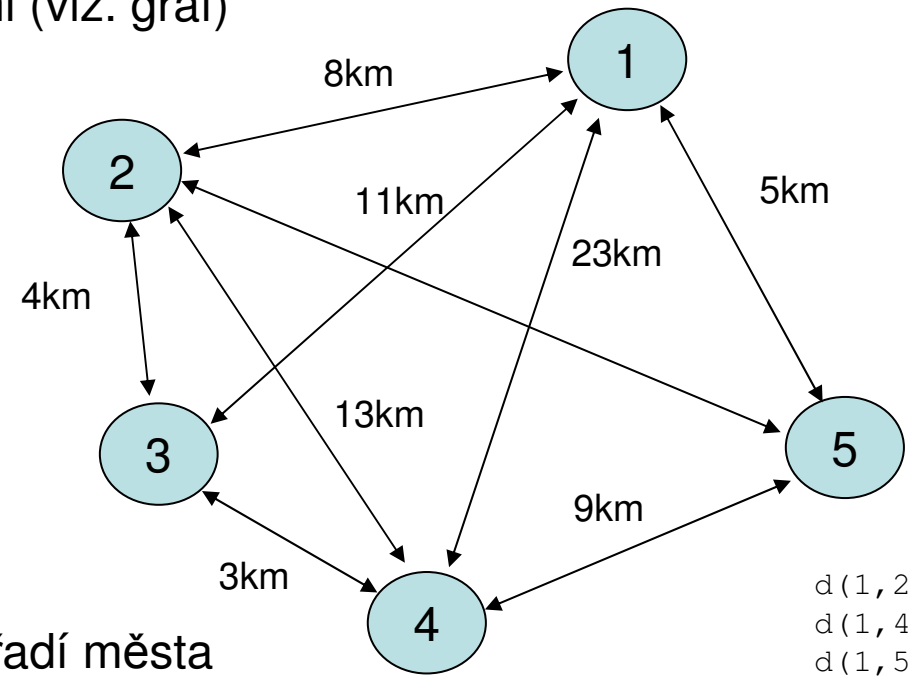
Přírustek feromonu od mravence (pokud projde hranou)

Aplikace

- Logistika
- Dopravní systémy
- Směrování v sítích
- Řešení standardní úlohy obchodního cestujícího (TSP – Travelling Salesman Problem)

Příklad

Mějme pět měst se vzdálenostmi (viz. graf)



Hodnoty $\eta_{i,j}$ jsou vzdálenosti mezi městy

$$\eta_{1,3} = 11$$

$$\eta_{4,2} = 13$$

```
d(1,2,3,4,5)=24km
d(1,4,3,2,5)=41km
d(1,5,4,2,3)=31km
… (n-1)! možností
```

V jakém pořadí města projít, abychom urazili minimální vzdálenost ?

Řešení: pošleme mravence, mravenec nesmí projít jedno město dvakrát