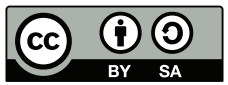
brmiversity: Umělá inteligence a teoretická informatika

Přednáška č. 4

Petr Baudiš ⟨pasky@ucw.cz⟩

brmlab 2011

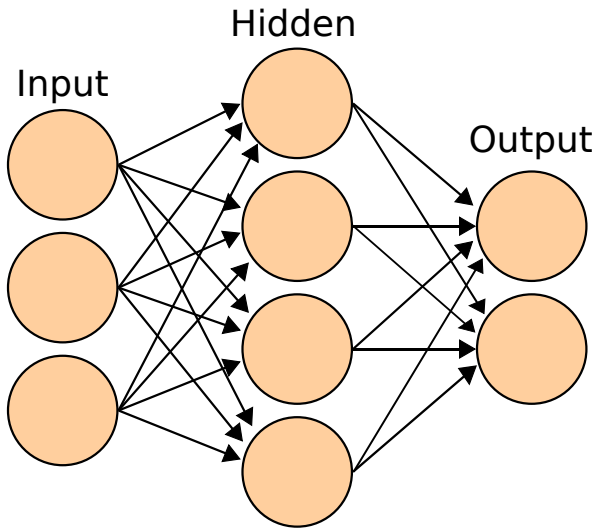


brmlab hackerspace prague

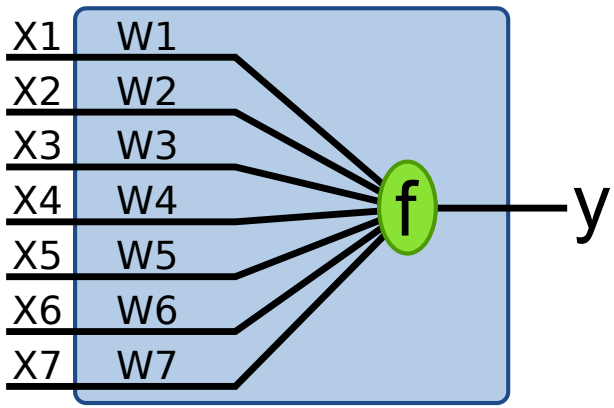
Outline

Umělá neuronová síť

- Umělé neurony (“výpočetní krabičky”) dostávají vstupy (čísla) a na jejich základě generují výstup (číslo)
- Obvykle: Vrstvy striktně oddělené, vstupní vrstva se vstupy zvnějšku, výstupní vrstva s výstupem pro uživatele, skryté vrstvy vyhodnocují různé charakteristiky vstupů
- Dnes: Jediný umělý neuron

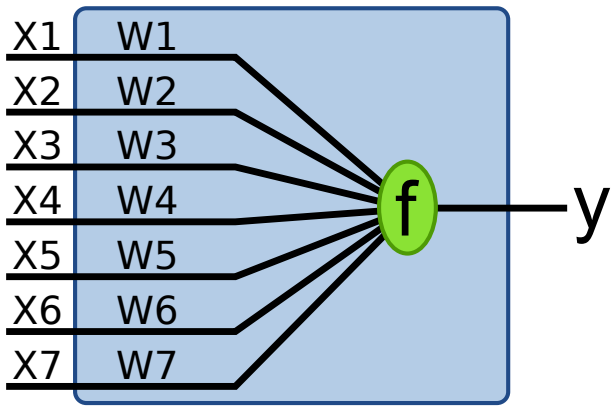


Perceptron



- n vstupů a práh $\Leftarrow$ výstup
- Lineární kombinace — vstupy mají různé váhy,
  vynásobíme, sečteme a otestujeme
- Výstup je 1/0 nebo něco složitějšího

Perceptron



- n vstupů a práh $\Leftarrow$ výstup
- Lineární kombinace — vstupy mají různé váhy,
  vynásobíme, sečteme a otestujeme
- Výstup je 1/0 nebo něco složitějšího
- Vstup $\vec{x}$, váhový vektor $\vec{w}$, práh b
- $\sum_{i=0}^{n} w_i \cdot x_i > b$
- Zjednodušení — virtuální vstup pro práh

Geometrická interpretace

Non-free artwork. See http://www.mblondel.org/journal/2010/10/31/kernel-perceptron-in-python/

Co dokáže perceptron?

- Lze vzorky čistě rozdělit podrovinou?
- **Lineární separabilita** množin
- Omezení — např. XOR potřebuje "dvě čáry"

Non-free artwork omitted. See http://cs-alb-pc3.massey.ac.nz/notes/59318/l7.html

brmlab hackerspace prague

Učení perceptronu

- "Rotační" algoritmus
- Náhodná inicializace přímky
- Iterujeme učení podle vstupních množin:
  - Postupně iterujeme body a klasifikujeme
  - Správná klasifikace — neděláme nic
  - Špatná klasifikace — pootočíme vektor, aby "pojmul" i nový bod

Non-free artwork omitted. See `http://www.willamette.edu/~gorr/classes/cs449/Classification/perceptron.html`

Učení perceptronu

- Geometrické pozorování: $\vec{w} \cdot \vec{x} > 0$, pak úhel mezi nimi je menší než 90 deg
- Geometrické pozorování: Sečteme-li dva vektory, vyjde nám vektor "mezi"
- Výstup je 0 a měl být 1: $w_i' = w_i + x_i(t)$
- Výstup je 1 a měl být 0: $w_i' = w_i - x_i(t)$

Non-free artwork omitted. See `http://www.willamette.edu/~gorr/classes/cs449/Classification/perceptron.html`

Otázky?

Příště: Vícevrstvé neuronové sítě a jejich učení (backpropagation).

Outline

Filosofie

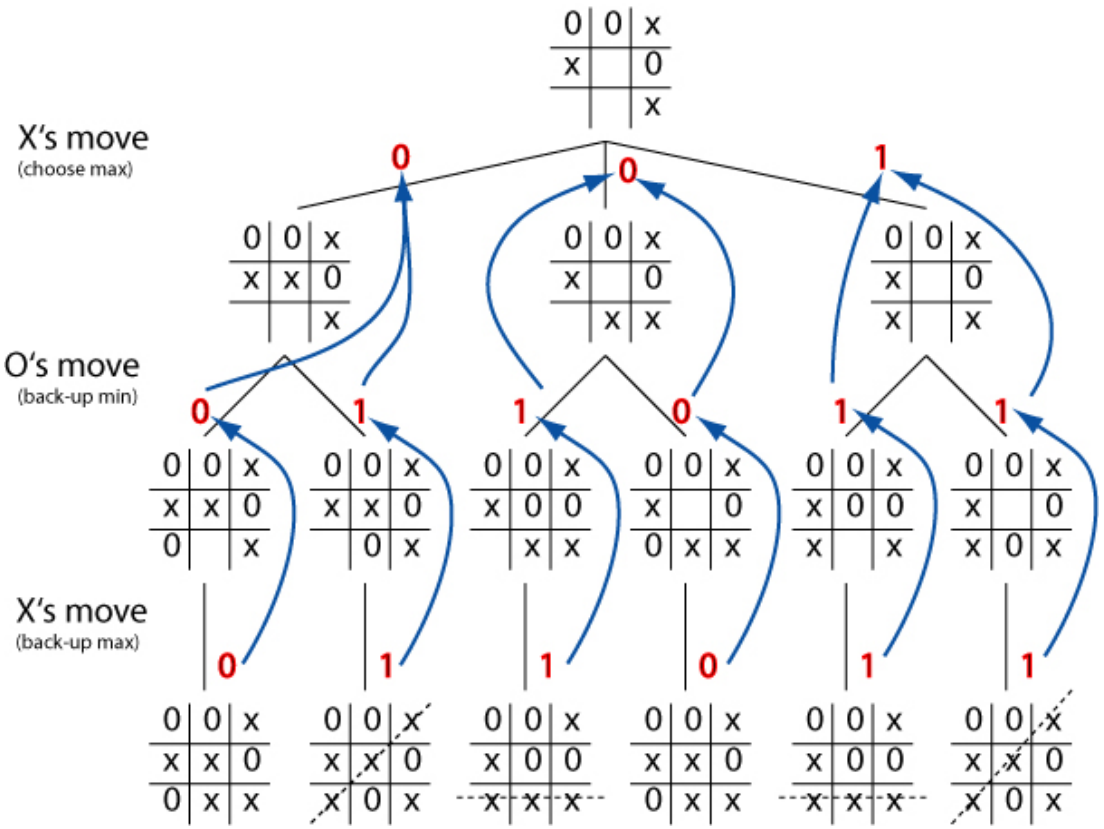
- Silná vs. slabá AI
- AI-complete úlohy
- Bude víc 23.11.
- Dnes jeden příklad slabé AI (je AI-complete?)

Matematické hry

- Teorie her — hráči se střídají v akcích, každý dostane nějakou výplatu po dokončení sekvence akcí
- Hry dvou hráčů s úplnou informací a nulovým součtem

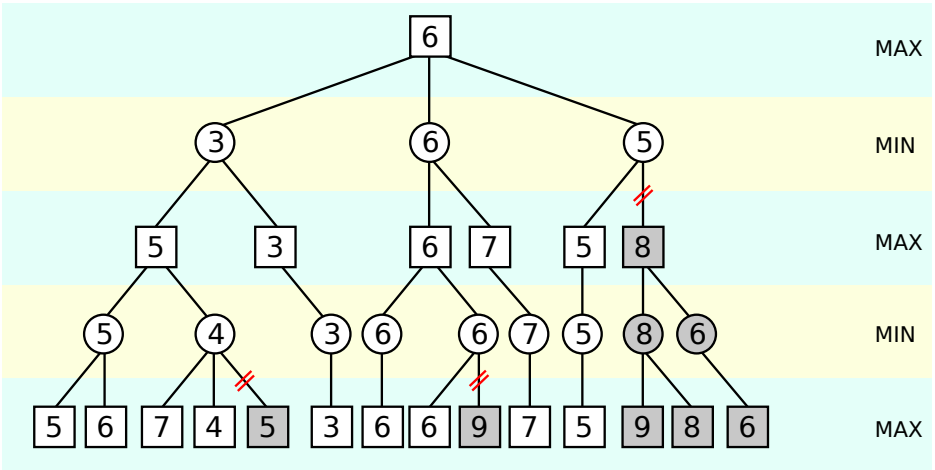
Minimaxový strom

- "Adversarial planning"
- Tah je tak dobrý jako nejnepříjemnější tah protihráče
- Omezení na hloubku stromu — vyhodnocovací funkce



α, β prořezávání

- Základní urychlovací heuristika
- Pamatujeme si nejhorší a nejlepší hodnoty z minimaxových sousedů, zařízneme prohledávání uzlu, je-li jasné, že bude horší (max, α) nebo lepší (min, β)
- Pro efektivitu je důležité pořadí vyhodnocování tahů



Další vylepšení

- Transpoziční tabulky
- Prořezávání, singular extensions, atd.
- Problémy — dobrá funkce, větvící faktor, efekt horizontu

Monte Carlo Tree Search

- Technika Monte Carlo
- Minimaxový strom s očekáváními
- Multi-armed bandit

Non-free artwork.

Otázky?

Příště: Prohledávání v grafech.

Outline

Levenšteinova editační vzdálenost

- Dva stringy; editace jsou přidání, změna, odebrání
- Dynamický algoritmus:

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

brmlab hackerspace prague

Další způsoby porovnávání

- Myersův algoritmus (nejdelší společná podposloupnost neboli editační skript; jen + a −)
- bdiff (nejdelší společný podřetězec — rekurzivně)
- xdelta (rsync; Rabinovo hashování)

Outline

Metody tvorby algoritmů

- Kuchařka pro efektivní algoritmy na obecné třídy problémů
- Rozděl a panuj — problém rozdělíme na podproblémy a ty počítáme rekurzivně
- Hladový algoritmus — problém budujeme inkrementálně, v každé chvíli víme, kudy dál
- Dynamický algoritmus — problém budujeme inkrementálně, sdílíme výsledky jednodušších podproblémů

Rozděl a panuj — příklad

- Binární vyhledávání:
  1 3 9 10 12 13 17 19 24 26 27 30 31
- Quicksort
- Asymptotická složitost — Master Theorem
  (matematika viz Wikipedie)

Rozděl a panuj — příklad

- Binární vyhledávání:
  1 3 9 10 12 13 **17** 19 24 26 27 30 31
- Quicksort
- Asymptotická složitost — Master Theorem
  (matematika viz Wikipedie)

Rozděl a panuj — příklad

- Binární vyhledávání:
  1 3 9 10 12 13 17 19 24 26 27 30 31
- Quicksort
- Asymptotická složitost — Master Theorem
  (matematika viz Wikipedie)

Rozděl a panuj — příklad

- Binární vyhledávání:
  1 3 **9** 10 12 13 17 19 24 26 27 30 31
- Quicksort
- Asymptotická složitost — Master Theorem
  (matematika viz Wikipedie)

Rozděl a panuj — příklad

- Binární vyhledávání:
  1 3 9 10 12 13 17 19 24 26 27 30 31
- Quicksort
- Asymptotická složitost — Master Theorem (matematika viz Wikipedie)

Rozděl a panuj — příklad

- Binární vyhledávání:
  1 3 9 10 **12** 13 17 19 24 26 27 30 31
- Quicksort
- Asymptotická složitost — Master Theorem (matematika viz Wikipedie)

Rozděl a panuj — příklad

- Binární vyhledávání:

  1 3 9 10 12 13 17 19 24 26 27 30 31
- Quicksort
- Asymptotická složitost — Master Theorem (matematika viz Wikipedie)

Hladový algoritmus — příklad

- Minimální kostra (připomenutí)
- Rozvrhovací problém
- Dijkstra

Hladový algoritmus — příklad

- Minimální kostra (připomenutí)
- Rozvrhovací problém
- Dijkstra

- Hladový algoritmus lze použít, pokud problém vykazuje *optimální podstrukturu*
- Matematická abstrakce — **matroid** $M = (E, I)$,
  E je základní množina, I je systém nezávislých podmnožin:
  - $\emptyset \in I$ (prázdná množina je nezávislá)
  - $A \in I, A' \subset A \quad \Rightarrow \quad A' \in I \quad \forall A, A'$ (dědičnost)
  - $A \in I \wedge B \in I \wedge |A| > |B| \quad \Rightarrow \quad \exists x \in A, \quad B \cup x \in I$

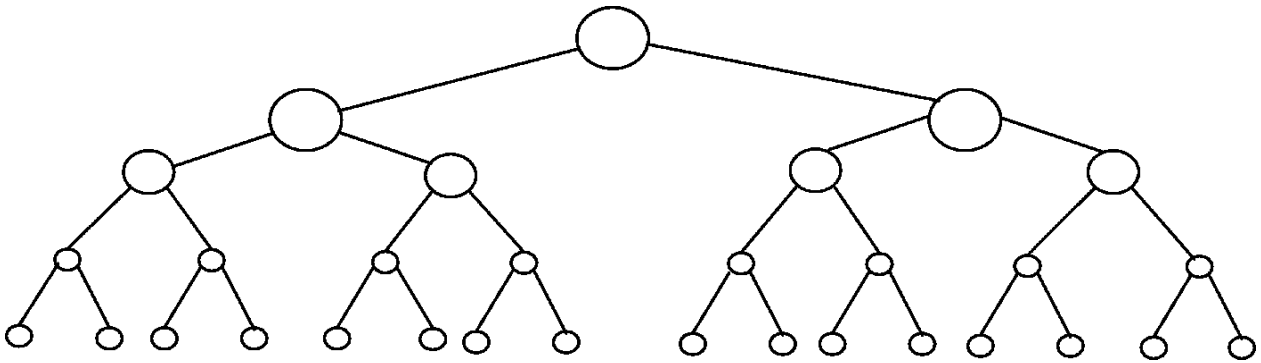
Dynamický algoritmus

- Editační vzdálenost
- Fibonacciho posloupnost
- Dijkstra

- Optimální podstruktura a překrývající se podproblémy

Třídění

- Jak dlouho trvá třídění? Spodní odhad je $O(n \cdot \log n)$
- Vyhledávací strom modeluje rozhodovací strom algoritmu
- Nebo ano? Radix sort, ale je to podvod!



Otázky?

Příště: Pravděpodobnostní algoritmy.

Outline

Rekapitulace

- Rekapitulace: Nezajímá nás, jak rychle to poběží, ale jestli to někdy doběhne.
- Zkoumáme výpočetní *možnosti* algoritmů.

- Minule jsme přemýšleli, čím vším lze algoritmy vykonávat (a že je to ekvivalentní).
- Zapomněli jsme (mj.) na **celulární automaty**!

Halting Problem

- Máme program $z = x(y)$; x, y, z jsou čísla
  (kód, parametr / vstup, návratová hodnota / výstup)
- Program doběhne (v konečném čase) a vrátí hodnotu
  (*řeší* problém), nebo se zacyklí

Halting Problem

- Máme program $z = x(y)$; x, y, z jsou čísla
  (kód, parametr / vstup, návratová hodnota / výstup)
- Program doběhne (v konečném čase) a vrátí hodnotu
  (*řeší* problém), nebo se zacyklí

- Nekonečná smyčka v závislosti na datech,
  schovaná uprostřed složitého kódu
- Halting Problem: Zacyklí se program x na vstupu y?
- Dokážeme napsat program, který řeší halting problem?

Halting Problem: Důkaz

- Důkaz sporem — dejme tomu, že máme funkci $HALT(x, y)$, x je kód programu ($c(FUNKCE)$) a y je vstup
- Ukážeme, že existence takové funkce způsobuje logické trhliny v časoprostoru

Halting Problem: Důkaz

- Důkaz sporem — dejme tomu, že máme funkci $HALT(x, y)$, x je kód programu ($c(FUNKCE)$) a y je vstup
- Ukážeme, že existence takové funkce způsobuje logické trhliny v časoprostoru
- Trik: Nadefinujme si funkci
  $PODRAZ(x)\downarrow \iff HALT(x, x) = NE$
  - Tedy $PODRAZ(x)$ zavolá funkci $HALT(x, x)$
  - Pokud by se program x na vstupu x (kód sama sebe) zastavil, $PODRAZ(x)$ se zacyklí
  - Pokud by se program x na vstupu x (kód sama sebe) zacyklil, $PODRAZ(x)$ doběhne

Halting Problem: Důkaz

- Důkaz sporem — dejme tomu, že máme funkci $HALT(x, y)$, x je kód programu ($c(FUNKCE)$) a y je vstup
- Ukážeme, že existence takové funkce způsobuje logické trhliny v časoprostoru
- Trik: Nadefinujme si funkci
  $PODRAZ(x)\downarrow \iff HALT(x, x) = NE$
  - Tedy $PODRAZ(x)$ zavolá funkci $HALT(x, x)$
  - Pokud by se program x na vstupu x (kód sama sebe) zastavil, $PODRAZ(x)$ se zacyklí
  - Pokud by se program x na vstupu x (kód sama sebe) zacyklil, $PODRAZ(x)$ doběhne
- Zavolejme $PODRAZ(c(PODRAZ))$!
- $HALT(c(PODRAZ), c(PODRAZ)) = ANO \iff$
  $PODRAZ(c(PODRAZ))\downarrow \iff$
  $HALT(c(PODRAZ), c(PODRAZ)) = NE$
- To je spor, $ANO \neq NE$.

brmlab hackerspace prague

Otázky?

Příště: Rekurzivní a rekurzivně spočetné množiny.

Děkuji vám

pasky@ucw.cz

Příště: **Nebude.**

Popříště: Prohledávání grafů a hledání cesty (základní algoritmy, umělá inteligence, adaptivní agenti), neuronové sítě, vyčíslitelnost.

brmlab hackerspace prague