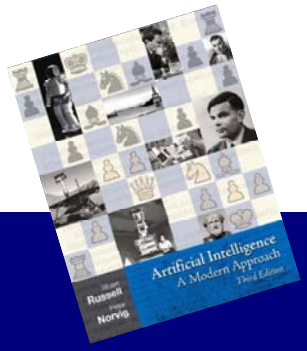


Umělá inteligence II

Roman Barták, KTIML

roman.bartak@mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

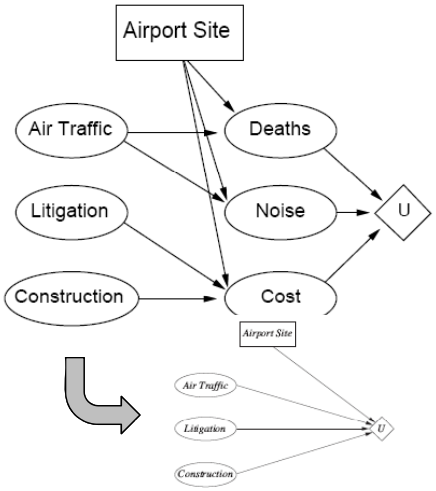
7

Úvodem

- **Teorie pravděpodobnosti** (probability theory) popisuje v co má agent věřit na základě pozorování.
- **Teorie užitku** (utility theory) popisuje co agent chce.
- **Teorie rozhodování** (decision theory) spojuje obě předchozí teorie dohromady a popisuje co by měl agent dělat.
  - podíváme se na statický případ (jedno rozhodnutí)
  - i na posloupnost rozhodnutí

Rozhodovací sítě

- Jak obecně popsat mechanismus rozhodování?
- **Rozhodovací síť** (influenční diagram) popisuje vztahy mezi vstupy (současný stav), rozhodnutími (budoucí stav) a užitkem (budoucího stavu).



- **Náhodné uzly** (ovál) reprezentují náhodné proměnné stejně jako v Bayesovských sítích.
- **Rozhodovací uzly** (obdélníky) popisují rozhodnutí, které může agent udělat (zatím uvažujeme jediný).
- **Uzly užitku** (kosočtverce) popisují funkci užitku.

Rozhodovací sítě

vyhodnocovací algoritmus

- Akce se vybírá na základě vyzkoušení všech možných hodnot rozhodovacího uzlu.

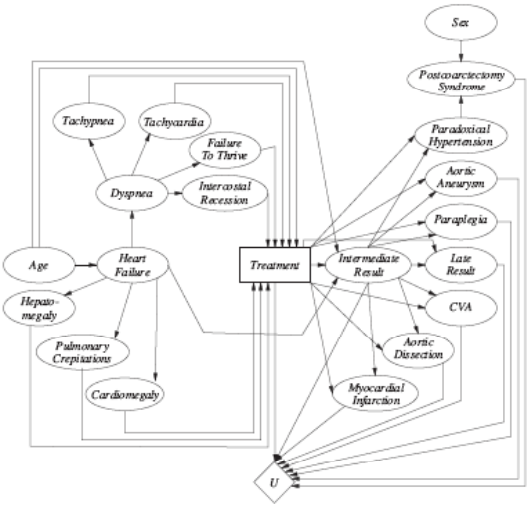
1. nastavíme hodnoty pozorovaných proměnných
2. pro každou možnou hodnotu rozhodovacího uzlu
   a) nastavíme rozhodovací uzel na danou hodnotu
   b) použitím pravděpodobnostní inference vypočteme pravděpodobnosti rodičů uzlu užitku
   c) vypočteme užitek pomocí funkce užitku
3. vybereme akci s největším užitkem

- Pro případy, kde je více uzlů užitku používáme při výběru akce techniky pro **víceatributové funkce užitku.**

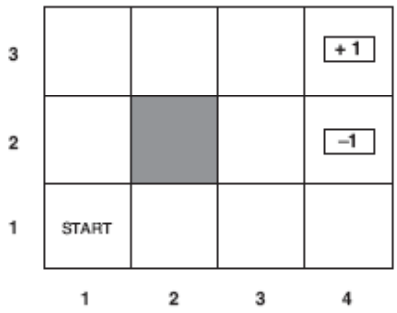
Rozhodovací sítě

konstrukce

- **Vytvoření kauzálního modelu**
  určení symptomů, nemocí, léčby a vztahů mezi nimi
- **Zjednodušení kvalitativního modelu**
  pokud nám jde o léčbu, můžeme odstranit uzly, které s ním nesouvisí.
  některé uzly lze spojit (léčba a její načasování)
- **Přiřazení pravděpodobností**
  vyplnění pravděpodobnostních tabulek jako v Bayesovské síti (dle učebnic, databází pacientů, rozhodnutí experta)
- **Navržení funkce užitku**
  můžeme enumerovat možné výstupy (záleží nejen na lékaři, ale i na pacientovi, který může mít jiné preference)
- **Verifikace a doladění modelu**
  výsledky se porovnají se zlatým standardem, například skupinou lékařů
- **Analýza citlivosti**
  kontrola, zda výsledky systému nejsou citlivé na drobné změny vstupu (malé změny vstupu vedoucí k velkým rozdílům výstupu indikují problém v modelu)

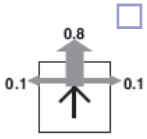


Motivační příklad



- Uvažujme agenta pohybujícího se v prostředí o rozměrech 3×4
- Prostředí je **plně pozorovatelné** (agent vždy ví, kde se nachází)

- Agent může provést akce Up, Down, Left, Right
  - množinu akcí pro stav s značíme A(s)
  - výsledek akce je ale **nejistý**
    - s pravděpodobností 0,8 půjde správným směrem
    - s pravděpodobností 0,1 půjde kolmo k požadovanému směru
  - pravděpodobnost, že [Up,Up,Right,Right,Right] vede ze START do cíle (hodnota +1) je $0.8^5 + 0.1^4 \times 0.8 = 0.32776$.



MDP

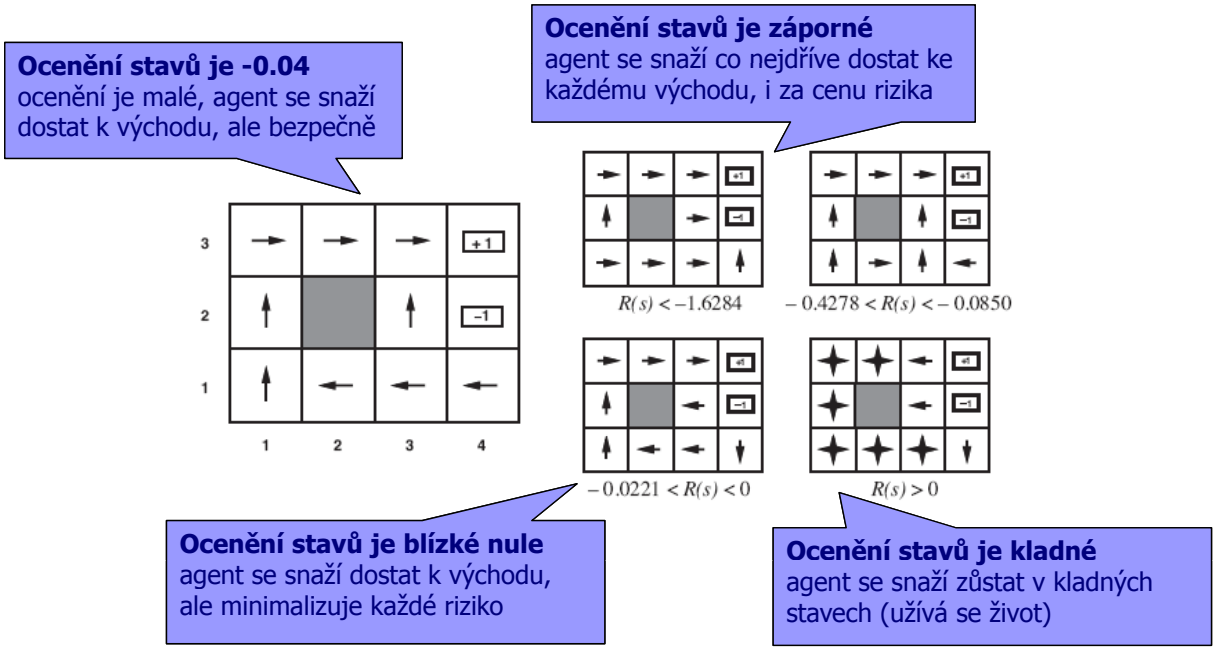
- Pro **popis přechodů** (aplikace akce) použijeme pravděpodobnostní distribuci P(s`|s,a) – pravděpodobnost přechodu z s do s` akcí a.
  - opět uvažujeme Markovský předpoklad (pravděpodobnost přechodu nezávisí na předchozích navštívených stavech)
- **Užitek** tentokrát závisí na prošlých stavech
  - každý stav má přiřazeno **ocenění** (reward) R(s)
    - např. +1 (cíl), -1 (nechtěný stav), -0.04 (ostatní stavy)
  - **funkce užitku** bude (zatím) součet ocenění navštívených stavů
    - funkce užitku je zde podobná jako při hledání nejkratší cesty do cíle (plus máme stochastický charakter akcí)
- **Markovský rozhodovací proces** (Markov Decision Process – MDP)
  - sekvenční rozhodovací problém v plně pozorovatelném stochastickém prostředí s Markovským přechodovým modelem a aditivní funkcí užitku

Řešení MDP

- Ve stochastickém prostředí nemůžeme pracovat s pevným pořadím akcí (plánem)
  - agent se může vydat jinam, než bylo naplánováno
- Protože předem nevíme, kam akce agenta zavede, potřebujeme v každém stavu vědět, co dělat (kam jít).
- **Řešením MDP je strategie** (policy), což je funkce určující pro každý stav doporučenou akci – $\pi(s)$
  - hledáme strategii s největším očekávaným užitkem (**optimální strategie**)
  - strategii může realizovat klasický **reflexní agent**

Optimální strategie
příklad

- optimální strategie maximalizuje očekávaný užitek
  - záleží samozřejmě na ocenění stavů



Užitek v čase
horizont

- Jak obecně definovat funkci užitku pro posloupnost stavů?
  - je to podobné jako pro **víceatributové funkce užitku** $U([s_0,s_1,\ldots,s_n])$ (stavy jsou atributy), ale
  - jaký máme horizont?
- **konečný horizont**
  - máme daný pevný termín N a po něm už na ničem nezáleží $U([s_0,s_1,\ldots,s_{N+k}]) = U([s_0,s_1,\ldots,s_N])$
  - optimální politika záleží na termínu (není stacionární)
    - pro N=3 volí ze stavu (3,1) akci Up
    - pro N=100 volí ze stavu (3,1) bezpečnou akci Left
- **nekonečný horizont**
  - není důvod se ve stejném stavu chovat v různé časy různě
  - **optimální politika je stacionární**
  - to neznamená nutně nekonečné posloupnosti stavu, pouze zde není žádný termín dokončení (deadline)

Užitek v čase

definice funkce

- funkce užitku se chová jako **víceatributová funkce užitku** $U([s_0,s_1,\dots,s_n])$
- Pro získání jednoduchého vzorečku pro výpočet funkce užitku uvažujeme **stacionární preference**
  - $[s_0,s_1,s_2,\dots]$ má preferenci k $[s_0,s`_1,s`_2,\dots]$ stejnou jako $[s_1,s_2,\dots]$ k $[s`_1,s`_2,\dots]$
  - naše preference „zítra a dnes" jsou stejné
- V případě stacionárních preferencí jsou dva způsoby jak rozumně definovat funkci užitku
  - **aditivní funkce užitku**

    $U([s_0,s_1,s_2,\dots]) = R(s_0) + R(s_1) + R(s_2) + \dots$
  - **kumulovaná (discounted) funkce užitku**

    $U([s_0,s_1,s_2,\dots]) = R(s_0) + \gamma\, R(s_1) + \gamma^2\, R(s_2) + \dots$
    - **faktor slevy** $\gamma \in (0,1)$
      - preference mezi aktuálním a budoucím oceněním (blízko 0 = budoucnost není důležitá, blízko 1 = budoucnost je stejně důležitá jako současnost – aditivní funkce užitku)
      - odpovídá úrokové míře $1/\gamma - 1$

Užitek v čase

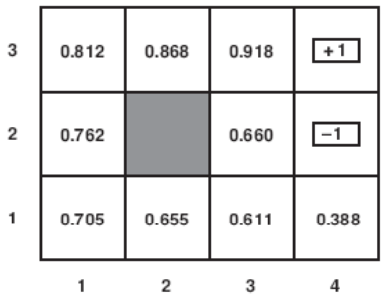
vlastnosti

- Budeme používat **kumulovanou funkci užitku**
  - pro světy bez cílového stavu a nekonečný horizont by aditivní funkce užitku byla problémová (pro nekonečné posloupnosti stavů dává $+\infty$ nebo $-\infty$)
  - **užitek** v kumulované funkci užitku je **konečný** (uvažujeme omezené ocenění s maximem R_{max})

    $U([s_0,s_1,s_2,\dots]) = \Sigma_{i=0,\dots,+\infty}\, \gamma^i\, R(s_i) \le \Sigma_{i=0,\dots,+\infty}\, \gamma^i\, R_{max} = R_{max} / (1-\gamma)$
  - Pokud má prostředí cílový stav, do kterého se agent garantovaně může dostat, nebudeme potřebovat pracovat s nekonečnými posloupnostmi.
    - **řádná (proper) strategie** – garantuje dosažení cílového stavu
      - můžeme používat aditivní funkci užitku
    - strategie, které nejsou řádné (improper), mají nekonečný celkový užitek
  - U nekonečné posloupnosti lze porovnat **průměrné ocenění** (lepší je zůstat ve stavu s oceněním 0.1 než ve stavu s oceněním 0.01)

Optimální strategie
vlastnosti

- Hledáme očekávaný užitek strategie π pro počáteční stav s
  $U^{\pi}(s) = E[\Sigma_{i=0,\ldots,+\infty} \gamma^i R(S_i)]$
  - S_i je náhodná proměnná popisující stav v čase i
- **Optimální strategie** pro počáteční stav s
  $\pi^*_s = \text{argmax}_{\pi} U^{\pi}(s)$
- Záleží ale na počátečním stavu?
  - když se strategie π^*_a a π^*_b sejdou v nějakém stavu c není důvod, aby pokračovaly různě
- Můžeme definovat **užitek stavu** $U(s) = U^{\pi^*}(s)$
  - R(s) je „krátkodobé" ocenění, zatímco U(s) je „dlouhodobé" ocenění
  - akce budeme vybírat na základě principu maximalizace očekávaného užitku
    $\pi^*(s) = \text{argmax}_{a \in A(s)} \Sigma_{s'} P(s'|s,a) U(s')$
  - Pozor, to nemusí být akce vedoucí do stavu s největším U(s)!



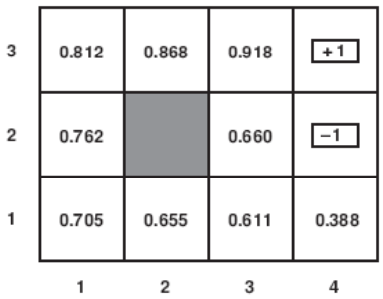
Bellmanova rovnice

- Užitek stavu přímo závisí na užitku okolních stavů (pokud agent volí optimální akci).

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \Sigma_{s'} P(s'|s,a) U(s')$$

- Tomuto vztahu se říká **Bellmanova rovnice**.

$$U((1,1)) = -0.04 + \gamma \max[0.8U((1,2)) + 0.1U((2,1)) + 0.1U((1,1)),\ 0.9U((1,1)) + 0.1U((1,2)),\ 0.9U((1,1)) + 0.1U((2,1)),\ 0.8U((2,1)) + 0.1U((1,2)) + 0.1U((1,1))]$$



Iterace hodnot

algoritmus

- Na základě Bellmanovy rovnice můžeme navrhnout algoritmus pro řešení MDP.
- Problém je, že máme soustavu nelineárních rovnic.
- Použijeme **iterativní postup**
  1. začneme s libovolnými vstupními hodnotami U(s)
  2. U(s) upravíme na základě Bellmanových rovnic (**Bellmanův update**)

$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \Sigma_{s'} P(s'|s,a)\, U_i(s')$

function VALUE-ITERATION(mdp, ϵ) returns a utility function
inputs: mdp, an MDP with states S, transition model **P**, reward function R, discount γ
ϵ, the maximum error allowed in the utility of any state
local variables: U, U', vectors of utilities for states in S, initially zero
δ, the maximum change in the utility of any state in an iteration

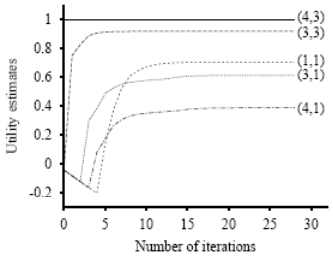
repeat
$U \leftarrow U'$; $\delta \leftarrow 0$
for each state s in S do
$U'[s] \leftarrow R[s] + \gamma \max_a \sum_{s'} \mathbf{P(s'|s,a)}\ U[s']$
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U

Iterace hodnot

konvergence

Jak můžeme garantovat konvergenci iterace hodnot ke skutečné hodnotě užitku stavu?

- Bellmanův update splňuje **vlastnost kontrakce** ($|f(x) - f(x')| < c|x - x'|$, kde $0 \le c < 1$)
  - má jediný pevný bod
  - po každé iteraci se přiblížíme k pevném bodu
- U funkce U měříme vzdálenost jako maximální vzdálenost odpovídajících prvků vektoru.
  $|U| = \max_s |U(s)|$
- Nechť BU je Bellmanův update vektoru U, potom
  $|BU_i - BU'_i| \le \gamma\, |U_i - U'_i|$
  $|BU_i - U| \le \gamma\, |U_i - U|$, kde U je hledaný užitek stavu (pevný bod)

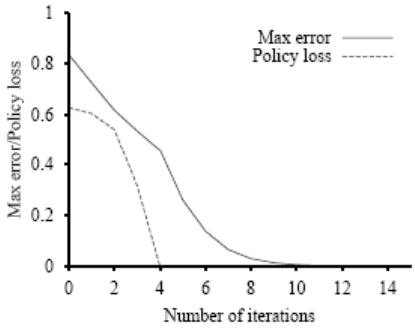


- **Počet kroků** potřebný pro přiblížení se k U na vzdálenost ε
  $|BU_0 - U| \le 2R_{max} / (1-\gamma)$ maximální vzdálenost od U na začátku
  $|BU_N - U| \le \gamma^N 2R_{max} / (1-\gamma)$ v každém kroku se přiblížíme o γ
  $N = \lceil \log(2R_{max} / \varepsilon(1-\gamma))/\log(1/\gamma) \rceil$ počet kroků do chyby ε
- **Ukončovací podmínka** pro dosažení vzdálenosti ε od U
  $|BU_{i+1} - BU_i| < \varepsilon\,(1-\gamma) / \gamma$

Iterace hodnot

ztráta strategie

- Agenta fakticky zajímá, jak dobrá je strategie π_i dle aktuální hodnoty U_i vzhledem k optimální strategii π^*.
  - definujme **ztrátu strategie** π_i jako $|U^{\pi_i} - U|$
  - pokud $|U_i - U| < \varepsilon$ potom $|U^{\pi_i} - U| < \varepsilon \gamma / (1 - \gamma)$
- V praxi tedy strategie konverguje k optimální strategii rychleji než funkce užitku a pevného bodu dosáhne dříve.



Iterace strategie

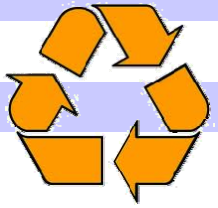
princip

- Optimální strategii můžeme získat, i když je funkce užitku ještě nepřesná.
  - Pokud je jedna akce jasně lepší než ostatní, potom užitek stavu nemusí být přesný.
- To můžeme využít u jiného algoritmu řešení MDP, tzv. **iterace strategie**, který je založený opakování následujících kroků:
  - **evaluace strategie** – výpočet U^{π_i} pro strategii π_i
    - protože známe akce, řešíme zjednodušené Bellmanovy rovnice

      $U^{\pi_i}(s) = R(s) + \gamma \Sigma_{s'} P(s'|s, \pi_i(s)) U^{\pi_i}(s')$
    - pro malý počet stavů n lze použít přesný výpočet metodami lineární algebry – časová složitost $O(n^3)$
    - pro větší počet stavů použijeme několik kroků zjednodušené iterace hodnot

      $U_{j+1}(s) \leftarrow R(s) + \gamma \Sigma_{s'} P(s'|s, \pi_i(s)) U_j(s')$
  - **zlepšení strategie**

    $\pi_{i+1}(s) \leftarrow \text{argmax}_{a \in A(s)} \Sigma_{s'} P(s'|s,a) U^{\pi_i}(s')$



Iterace strategie
algoritmus

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an **MDP** with states S, transition model **P**
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{\mathbf{a}} \sum_{s'} \mathbf{P(s'|s,a)}\ U[s'] > \sum_{s'} \mathbf{P(s'|s,\pi(s))}\ U[s']$ **then**
        $\pi[s] \leftarrow \text{argmax}_{\mathbf{a}} \sum_{s'} \mathbf{P(s'|s,a)}\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

- Strategií je konečně mnoho a při každém kroku strategii zlepšujeme, proto algoritmus musí skončit.