VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTELIGENTNÍ REAKTIVNÍ AGENT PRO HRU MS.PACMAN

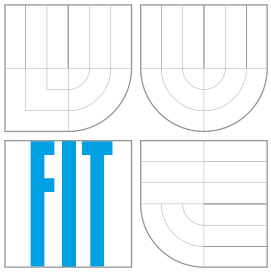
SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE
AUTHOR

BARBORA BLOŽOŇOVÁ

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTELIGENTNÍ REAKTIVNÍ AGENT PRO HRU MS.PACMAN

INTELLIGENT REACTIVE AGENT

FOR THE GAME MS.PACMAN

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE BARBORA BLOŽOŇOVÁ
AUTHOR

VEDOUCÍ PRÁCE Doc. Ing., Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.
SUPERVISOR

BRNO 2016

Abstrakt

Tato práce se zabývá umělou inteligencí pro složitější rozhodovací problémy, jako je hra s neurčitostí Ms. Pacman. Cílem práce je navrhnout inteligentního reaktivního agenta využívající metodu strojového učení, demonstrovat jej ve vizuálním demu Ms.Pacman a srovnat se známými informovanými metodami hraní her (Minimax, Alfa-Beta řezy). Práce je rozdělena primárně na dvě části. V teoretické části řeší problematiku metod hraní her, reaktivitu agenta a možnosti strojového učení (vše v kontextu Ms. Pacman). Druhá část práce je zaměřena na samotný popis návrhu a implementace agenta a na závěr jeho srovnání se zmíněnými známými metodami hraní her, zhodnocení dosažených výsledků a několik návrhů na vylepšení do budoucna.

Abstract

This thesis focuses on Artificial Intelligence for difficult decision problemes such as the game wit uncertainty Ms.Pacman. The aim of this work is to design and implement intelligent reactive agent using a method from the field of reinforcement learning, demonstrate it on visual demo Ms.Pacman and compare with well-known informed methods of playing games (Minimax,AlfaBeta Pruning). The thesis is primarily structured in two parts. The theoretical part deals with adversarial search (in games), reactivity of agent and possibilities of machine learning, all in the context of Ms.Pacman. The second part addresses the design of agent behaviour implementation and finally its comparison to other methods of adversarial search problem, evaluation of results and a few ideas for future improvements.

Klíčová slova

umělá inteligence, hry s nulovým součtem, hry s neurčitostí, reaktivní agent, Markovské rozhodovací procesy, posilované učení, Q-Learning

Keywords

artificial intelligence, zero-sum games, games with uncerntainty, reactive agent, Markov decision processes, reinforcement learning, Q-Learning

Citace

Barbora Bložoňová: Inteligentní reaktivní agent
pro hru Ms.Pacman, semestrální projekt, Brno, FIT VUT v Brně, 2016

Inteligentní reaktivní agent pro hru Ms.Pacman

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením Doc. Ing., Dipl.-Ing. Martina Drahanského, Ph.D. Dále prohlašuji, že jsem uvedla všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Barbora Bložoňová
19. ledna 2016

Poděkování

Ráda bych podělovala Doc. Ing., Dipl.-Ing. Martinu Drahanskému, Ph.D. za jeho odborné vedení, nadhled a celkovou spolupráci.

Obsah

Kapitola 1

Úvod

Svět her je pro umělou inteligenci jedno z velkých témat. Hra by nemohla být zábavná, kdyby by nepřítel nebyl dostatečně chytrý. Umělá inteligence silného oponenta ve složité hře představuje stále otevřeno novým technikám, metodám a možnostem. Výpočetní náročnost takových her staví velkou překážku k vytvoření *ideální* umělé inteligence například pro hru Go. Ačkoliv se již v minulosti podařilo překonat člověka například v hře Šachy, stále je to malý krůček pro tak velký svět zajímavých rozhodovacích problémů jakým je i Ms.Pacman. Ms.Pacman patří k těmto složitým úlohám nejen rozměrností stavů, jež ve hře můžou nastat, ale především náhodností chování svých oponentů, duchů. Nelze tedy přesně stanovit, jak se daný duch bude v danou situaci chovat, lze jen jaksi předpokládat, tudíž je těžší stanovit jednoznačne klasifikovat, jak se má umělá inteligence Ms.Pacman chovat.

Cílem této práce je navrhnout a implementovat dostatečně chytrou umělou inteligenci Ms.Pacman, a to v podobě inteligentního agenta, který bude co njelépe reagovat na své aktuální prostředí a to tak, aby jeho inteligence překonala problematiku náhodnosti a výpočetní náročnosti této hry. Kvalita agenta bude srovnána s již známými metodami řešení podobných problémů- Práce je rozdělena na dvě části, z nichž první se zaměřuje na teoretický základ, popis a klasifikaci samotného problému. Praktická část již nasťinuje samotný návrh možné implementace.

Kapitola 2

Metody hraní her

Tato kapitola si klade za cíl seznámit čtenáře s metodami hraní her. Hry úzce souvisí s metodami řešení úloh, zahrnují tedy snahu dostat se z počátečního stavu do cílového např. za použití posloupnosti nějakých pravidel.

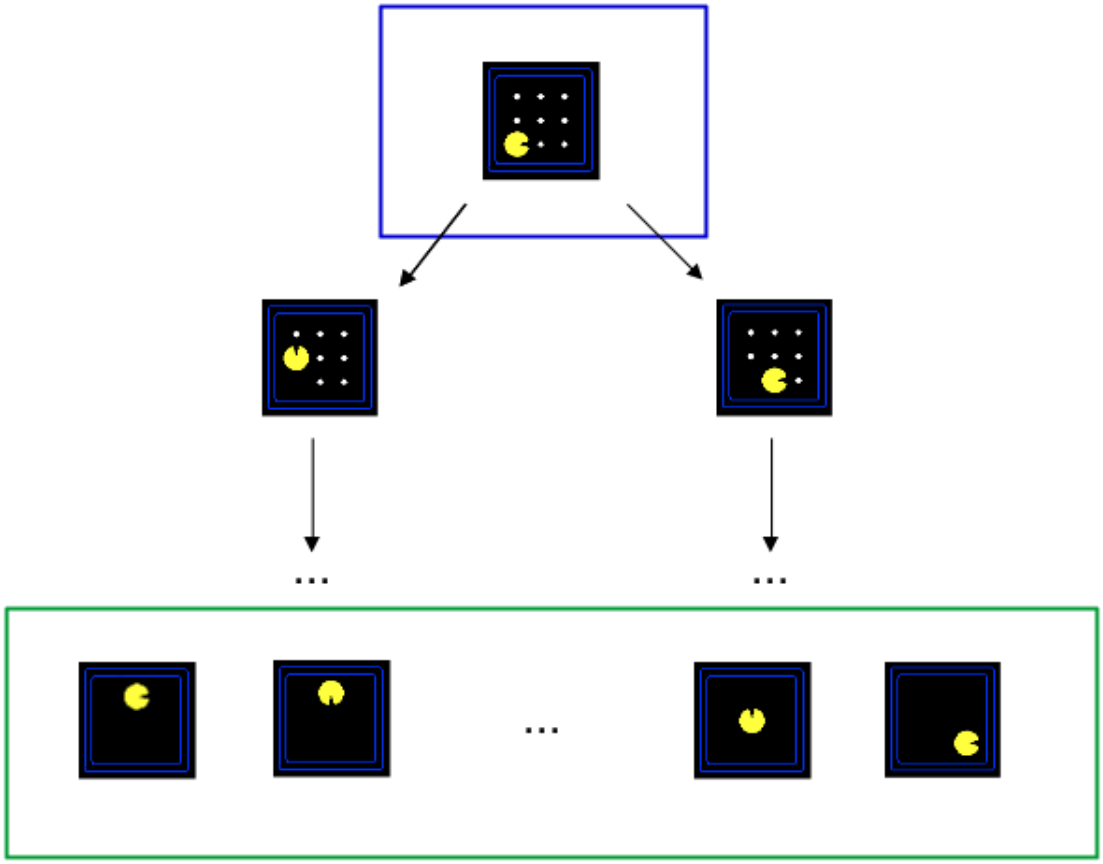
2.1 Základní pojmy

Stavový prostor

Stavový prostor [3] je množina všech stavů, které můžou ve hře nastat. Stavový prostor lze reprezentovat jako orientovaný graf, jehož *uzly* představují jednotlivé pozice ve hře (stavy) a orientované hrany přechody mezi jednotlivými stavy, tedy přípustné tahy ve hře. První uzel se nazývá *kořen*, koncové uzly, *listy*, pak reprezentují koncové pozice ve hře (cíly, stavy). Na obrázku 2.1 lze vidět ukázku omezeného (zjednodušeného) stavového prostoru hry Pacman. Velikost takto velmi omezeného stavového prostoru musí vzít v potaz: rozměry 2D hracího pole, 4 možné směry pohybu Pacmana a fakt, zda 8 kuliček jídla už (ne)bylo sněženo, tedy: $3 \times 3 \times 2^8 \times 4 = 9216$ (!) stavů a to nebyl brán v potaz ani jediný nepřítel, natožpak reálná velikost hracího pole desky.
K dosažení každého uzlu je potřeba vydat nějaké úsilí, **cenu**, která tedy udává nezáporné ohodonocení hrany z jednoho uzlu k druhému. **Hloubka uzlu** udává počet hran na cestě od počátečního uzlu k danému uzlu. Kořen má hloubku 0, jeho následníci mají hloubku 1 atp.

Optimální versus nejlepší řešení

Další důležitou součást metod řešení úloh je **hodnotící funkce** (*successor function*) [3]. Jedná se o funkci vyhodnocující celkovou cenu od jednoho uzlu k druhému, např. součet ceny a *heuristické funkce*. **Heuristická funkce** je funkce založená na empirických znalostech (až odhadech) o hře používaná při prohledávání stavového prostoru. Může to být například vzdálenost Ms.Pacman vůči svému cíli. Podle jejího využití dělíme algoritmy na **informované** a **neinformované**. Proč ale používáme heuristiku místo jistějšího systematického prohledávání stavového prostoru, které by mělo definovat vlastně *nejlepší* řešení? Je to především kvůli velké vypočetní náročnosti toho způsobu prohledávání a vyhodnocování celého stavového prostoru, který u složitějších problémů nabývá příliš velkých rozměrů, aby se vše stihlo provést např. v reálném čase při hraní hry. Pokud je heuristika *optimální*, zaručuje zvýšení efektivity a rychlosti.



Obrázek 2.1: Ukázka stavového prostoru pro hru Pacman. Každý pohyb Pacmana znamená nový stav hry, tedy uzel. Modrý obdelník ohraničuje počáteční stav,kořen, a zelený obdelník ohraničuje možné koncové stavy, listy.

Vlastnosti algoritmů

Rozlišujeme několik následujících vlastností algoritmů:

- **Úplnost** – Pokud existuje řešení, je garantováno, že ho algoritmus vrátí?
- **Optimálnost** – Je garantováno nalezení nejlepší řešení, tedy řešení s nejmenší cenou?
- **Časová složitost** – Kolik uzlů je nutno expandovat? Kolik to zabere času v nejhorším/nejlepším případě/průměrně?
- **Prostorová složitost** – Kolik dat je nutno si uchovávat v paměti v nejhorším/nejlepším případě/průměrně?

2.2 Hry

Hra (také se označuje jako *adversarial serach*) je z matematického hlediska rozhodovací problém, ve kterém figurují 2 a více účastníků, hráčů [3]. Hráči se snaží chovat racionálně, viz sekce 3.2.

Složité hry [3] jsou hry s tak velkým úplným stavovým prostorem, že by jeho neinformované prohledávání nebylo možné výpočetně zvládnout. Například šachy, Go, Ms. Pacman. V takových případech je potřeba omezit strom stavového prostoru pomocí hloubky, tedy maximálního počtu kroků dopředu oproti aktuálnímu stavu, a použít *statickou hodnotící funkci*, která určuje *pravděpodobnost* dosažení cíle z aktuálního stavu. Následující metody hraní her potřebují mít úplnou informaci o stavovém prostoru hry.

2.2.1 Minimax

Metoda [3]prohledávání do hloubky s omezením hloubky prohledávání. Hodnotící funkce nám udává určitou *hodnotu* listů $V(s)$, tedy nejlepší výsledek/užitek, kterého lze dosáhnout z aktuálního stavu. Následně se úrovni hráče při procházení stavového prostoru vybírá **maximum hodnoty:** $V(s) = \max\ V(s')$, kde $s' \in naslednici(s)$.
Na úrovni protihráče **minimum hodnoty:** $V(s') = \min\ V(s)$, kde $s \in naslednici(s')$. Hráči se takto rekurzivně střídají od listů až po finální vypočtení hodnoty kořene.

2.2.2 AlfaBeta Řezy

Metoda pro zmenšení stavového prostoru [3], odříznutím nadbytečných větví. Využívá dvou *mezí*:

- $\boldsymbol{\alpha}$ reprezentující dolní mez ohodnocení uzlu, tedy tah hráče
- $\boldsymbol{\beta}$ reprezentující horní mez ohodnocení uzlu,tedy tak protihráče

Algoritmus si pamatuje hodnoty svých mezí a postupně je aktualizuje porovnáváním meze s následníky uzlu aktuální úrovně a to podle toho, na které úrovni se nachází: (max - aktualizuji α na největší hodnotu následníků, min - aktualizuji β na nejmenší hodnotu následníků). Porovnávání probíhá dokud $\alpha < \beta$, takto se vynechají nadbytečné větve, jejichž procházení již nezmění výslednou cestu. Konečná hodnota kořene je stejná jako u Minimaxu a pokud by se uzly správně seřadily, výrazně se tím změní výpočetní náročnost algoritmu.

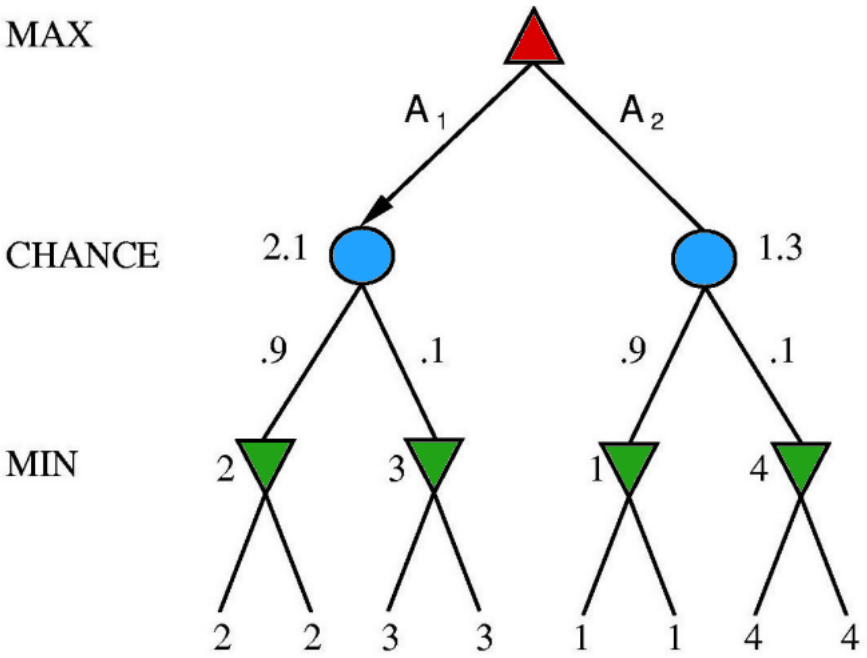
2.2.3 Expectimax

Expectimax [5] je jednou z možností jak řešit **hry s neurčitostí** jako je i Ms.Pacman. Hry s neurčitostí také zahrnují střídání hráčů a mají také úplnou informaci o stavu hry, avšak využívají získání těchto informací *pravděpodobnosti*, které popisují náhodný charakter stavů ve hře. Příkladem jsou hry, kde figuruje házení kostkou nebo např. řízení libovolného robota v reálném světě, kdy se může náhodně pokazit nějaká jeho součástka. Expectimax obohacuje každý tah hráče (jako Minimax) o náhodnost. **Hodnoty stavů** nyní navíc reflektují průměrnou pravděpodobnost stavů:

$V_{\min}(s) = \sum_{i=0} p_i * \min\ V(s')$, kde
$s' \in naslednici(s)$ a p_i je pravděpodobnost daného stavu i-té úrovně.

$V_{\max}(s') = \sum_{i=0} p_i * \max\ V(s')$, kde
$s' \in naslednici(s)$ a p_i je pravděpodobnost daného stavu i-té úrovně.

Tyto rovnice udávají tzv. **očekávaný užitek** (*expected utility*) dané úrovně [5]. Na obrázku 2.2 lze vidět použití algoritmu.



Obrázek 2.2: Na obrázku lze vidět ukázku vyhodnocování náhodných uzlů a výslednou cestu $\boldsymbol{A_1}$, kterou si algoritmus posléze zvolí.

Kapitola 3

Učící agent

Tato kapitola se zaměřuje na teoretický podklad hlavního cíle práce, inteligentní formu reaktivního agenta [4].

3.1 Reaktivní agent

Reaktivní agent se dá představit jako autonomní bot, který reaguje na své aktuální okolní prostředí. Nebere tedy v potaz následky svého konání. Matematicky lze zapsat *šesticí* [4] jako: $(S, T, A, vjem, ukon, akce)$, kde:

- S je množina okolních stavů agenta,
- $T, T \subseteq S$ je okolí, které může agent vnímat,
- $vjem : S \to T$ je agentem právě vnímaný stav okolí ($s \in S$)
- A je množina možných úkonů, jež může agent v daný stav vykonat,
- $ukon : A \times A \to S$ je vykonaný úkon, jež má za následek změnu prostředí S
- $akce : T \to A$ je vybraná akce na základě $vjemu$.

Agent představuje účastníka hry (hráče).

3.2 Racionalita

Každý hráč má nějaký cíl a k němu vedoucí strategii. Strategie se skládá z hráčových akcí a voleb, které pro hráče vedou k vlastnímu užitku (*utility*), potažmo výhře. Hráč se potýká s různými stavy ve hře a snaží se z nich vytěžit co nejvíce na základě svých preferencí [2]. Pokud tyto hráčovi preference dodržují axiomy dle [5], zaručují tak hráčův cíl, maximalizaci očekávané hodnoty *užitkové funkce*:

$$U([p_1, S_1; \ldots; p_n, S_n]) = \sum_i p_i U(S_i) p$$

Pro umělou inteligenci hráče je též důležité explicitně stanovit určitá pravidla, ze kterých má hráč vybírat užitek. Předpoklad racionality je hlavní rozdíl teorie her a teorie rozhodování.

3.3 Markovský rozhodovací proces

Agent se bude pohybovat ve *stochastickém* (náhodném) prostředí, je tedy vhodné si definovat Markovský rozhodovací proces [1], dále **MDP** (*Markov decision process*), který řeší takovéto nedeterministické problémy prohledávání, jedná se o čtveřici: (S, A, T, R), kde:

- $s, s \in S$ je množina stavů, která zahrnuje počáteční a koncový stav
- $a, a \in A$ je množina akcí
- $T(s, a, s')$ je **přechodová funkce** (*transition function*), nebo též *model*, tedy pravděpodobnost přechodu ze stavu s do s' ($P(s'|s, a)$)
- $R(s, a, s')$ je **odměnová funkce** (*reward function*), s ní souvisí exponenciální snižování hodnot odměn koeficientem γ (*discount*)

Důležitý rozdíl proti předchozím metodám hraní her je fakt, že výsledná hodnota aktuálního stavu závisí pouze na aktuálním stavu a jeho akci, ne na jeho historii. Další významný rozdíl je ten, že přechozí metody se snažily o nalezení optimálních plánu, nebo sekvence akcí z počátečního stavu až do cíle. MDP hledá **optimální strategii** (*policy*):

$$\pi^* : S \to A$$

π mapuje akci na každý stav a pokud se akce provede, maximalizuje očekávaný užitek.

Užitek je definován jako suma koeficientem γ snížených odměn tak, aby MDP měl větší šanci skončit (pro agenta je lepší vzít blízkou odměnu co nejdříve, tudíž optimálněji sbírá odměny).

Hodnota definuje **optimální očekávaný užitek** (akumulovaný průměr očekávaných výsledků) ze stavu (pro uzly max/min)

3.3.1 Value iteration a Policy iteration

Příkladem přepočtu hodnot je metoda **Value iteration**[1]:

1. začni od úrovně s hodnotou $V_0 = 0$
2. plň vektor optimálních hodnot $V_k(s)$ novými hodnotami vykonáním Expectimaxu pro aktuální úroveň pomocí rovnice:[1]

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_k(s')\right]$$

3. opakuj předchozí krok dokud vektor V_k nekonverguje (což zaručuje optimálnost metody)

Velký rozdíl oproti Expectimaxu je ten, že se nemusí provádět neustálá *rekurze* výpočtu očekávaného užitku, protože je už vypočten ve vektoru hodnot $V_k(s')$. Stále se však metoda potýká s vysokou prostorovou složitostí. Alternativou jsou proto metody **Policy iteration**, které se snaží o přímé nalezení otpimální strategie raději než zdlouhavé výpočty hodnot. O těchto metodách se v souvislosti se strojovým učením zmiňuje další kapitola 3.4.

[1] $\gamma V_k(s')$ je hodnota budoucí odměny

3.3.2 Q-hodnota

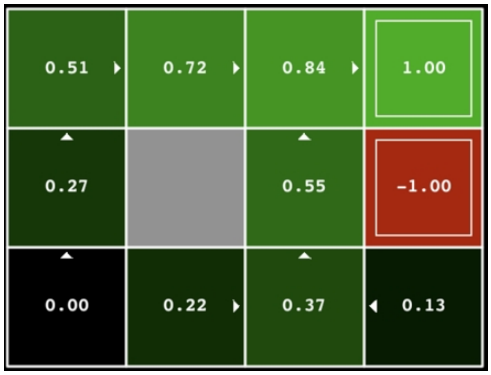
Posledním důležitým pojmem je Q-hodnota, která definuje optimální očekávaný užitek v budoucnosti z uzlu náhodnosti, tzv. *q-stavu*, příklad na obrázku 3.2.

Pro získávání optimálních V-hodnot a Q-honot v MDP tedy platí následující rovnice:[2][3]

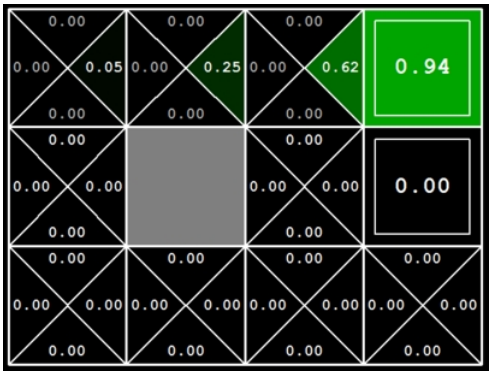
$$V^*(s) = \max_a Q^*(s)(s, a, s')$$

$$Q^*(s) = \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_k(s') \right]$$

Na obrázku 3.1 lze vidět optimální strategii a vypočtené hodnoty pro daný *grid-world* problém.



Obrázek 3.1: Problém představuje pole o velikosti 12 polí, 4 směry možnosti chůze a dva koncové stavy s odměnami ohodnocenými -1 a 1. Již po 5 iteracích je vidět optimální strategie, reprezentovaná šipkou směru (agent začínám v levém spodním rohu), vypočtena pomocí hodnoty V_{100} každého pole.

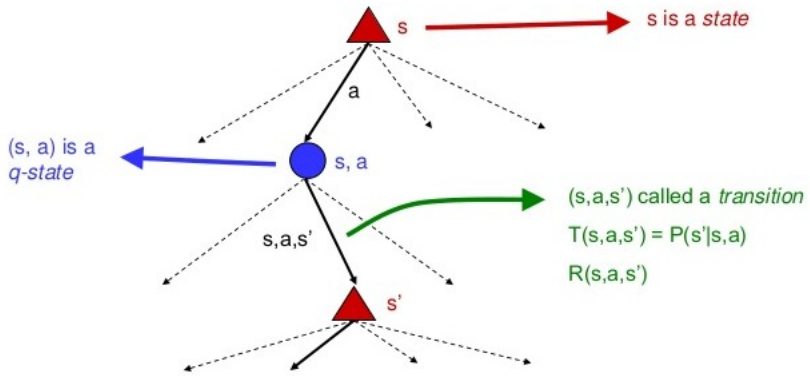


Obrázek 3.2: Ukázka počátečních Q-hodnot pro předešlý grid-world problém. Q-hodnoty se počítají pro všechny možné akce z daného stavu, proto jsou pro jedno políčko na hracím plánu 4 pro každý směr, kam může agent jít.

[2] hvězdička * značí optimální hodnotu

[3] rovnice přímo vychází z Bellmanových rovnic [5]

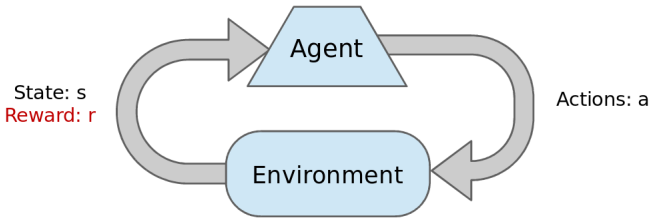
MDP lze též reprezentovat prohledávacím stromem, viz obrázek 3.3, který se velmi podobá Expectimaxu avšak jak již bylo řečeno není potřeba rekurze.



Obrázek 3.3: Na obrázku lze vidět analogii MDP k Expectimaxu.

3.4 Strojové učení

Další důležitou vlastností agenta bude schopnost *učit se* [6]. Agent se učí na pozorovaných výsledcích svých akcí, tzv. **vzorky** (s, a, s', r), kde r je odměna nového stavu. Stále se jedná o MDP, avšak největší rozdíl zde nastává v tom, že **neznáme** odměnovou funkci $\boldsymbol{R}$, ani přechodovou funkci $\boldsymbol{T}$. Agent (viz obrázek 3.4)se tedy musí metodou *pokus-omyl* NAUČIT tyto hodnoty modelu.



Obrázek 3.4: Agent využívající strojové učení.

Metody strojového učení [6]: se dělí dle **závislosti na modelu**

- (*model-based*) metoda chce empiricky vytvořit model, který již lze řešit pomocí MDP - to se děje na základě průběžného výpočtu očekávaného užitku:
  1. spočítej všechny výsledky/vzorky $\boldsymbol{s'}$ pro každé $\boldsymbol{s, a}$
  2. spočítej odhad okamžitého užitku, tedy odhad přechodové funkce $\hat{\boldsymbol{T}}(\boldsymbol{s, a, s'})$
  3. odhadni odměnovou funkci $\hat{\boldsymbol{R}}(\boldsymbol{s, a, s'})$ na základě objevených odměn stavů
- (*model-free*) metoda model nepotřebuje, stačí posbírat vzorky, protože rozdělenení praděpodobnosti pro daný vzorek určuje jeho výskyt (není potřeba počítat očekávaný užitek). Dále se budeme zabývat těmito metodami.

Dalším dělícím kritériem je **řízení akcí**:

- *pasivní* metody - je daná statická strategie $\boldsymbol{\pi(s)}$, kterou se agent pevně řídí a získává z ní vzorky, ze kterých se učí hodnoty stavů $\boldsymbol{V(s')}$ pro daný stav $\boldsymbol{s}$.
  Ukázka postupu metody přímého vyhodnocení (**direct evaluation**):
  1. následuj $\boldsymbol{\pi}$
  2. pro každý navštívený stav vypočti sumu koeficientem $\boldsymbol{\gamma}$ budoucích snížených hodnot
  3. spočítej průměr sumy

  Není potřeba $\boldsymbol{T}$, ani $\boldsymbol{R}$, avšak je zde přílišná abstrakce (plýtvání informací) a vyhodnocení stavů díky tomu má příliš velkou časovou náročnost.
  Další možností je **Time-Difference value learning**[6], dále TD učení, které se místo vyhodnocování hodnoty (Value iteration) snaží vyhodnotit přímo nové strategie na základě hodnot (**Policy iteration**). Vyhodnocení probíhá *po každé akci*, protože nelze zaručit, že se strategie bude vracet znovu do již vyhodnoceného stavu, aby opět přehodnotila svou hodnotu.

1. následuj $\boldsymbol{\pi}$
2. aktualizuj $\boldsymbol{V(s)}$ pokaždé, když narazíš na přechod pro vzorek $\boldsymbol{(s, a, s', r)}$ **Aktualizace** (*update*) spočívá v tom, že se vezme aktuální hodnota stavu a přičte se k ní o koeficient $\boldsymbol{\alpha}$ zmenšený rozdíl mezi očekávaným stavem a reálným vzorkem. $\boldsymbol{\alpha}$ reguluje fakt, že nové odměny budou mít větší váhu než staré.

$$\text{Vzorek } \boldsymbol{V(s)}: \quad \boldsymbol{vzorek = R(s, \pi(s), s') + \gamma V^{\pi}(s')}$$

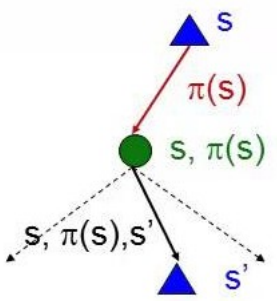
$$\text{Update } \boldsymbol{V(s)}: \quad \boldsymbol{V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(vzorek - V^{\pi}(s))}$$

TD učení je pasivní metoda, která je schopná získat ohodnocení hodnot $\boldsymbol{V}$, avšak není schopná aktivně přeměnit hodnoty na novou strategii $\boldsymbol{\pi(s)}$ dle rovnic MDP pro Policy iteration (viz obrázek 3.5):

$$\boldsymbol{\pi(s) = arg \max_{a} Q(s, a)}$$

$$\boldsymbol{Q(s, a) = \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V(s') \right]}$$

Problém je opět chybějící přechodová a odměnová funkce $\boldsymbol{(T, R)}$, to řeší až aktivní metody strojového učení.



Obrázek 3.5: Policy iteration metoda s potřebnými proměnnými.

- *aktivní* metody - je daná statická strategie $\boldsymbol{\pi(s)}$, ale agent provádí vlastní akce a získává z ní vzorky, ze kterých se učí optimální strategie a hodnoty stavů $\boldsymbol{V(s')}$. Mezi aktivní metody patří **Q-Learning**.

3.4.1 Q-Learning

Dosud se primárně vycházelo z $\boldsymbol{V}$-hodnot, avšak Q-Learning staví na vyhodnocování akcí, tedy využití $\boldsymbol{Q}$-hodnot, takže jediné co je potřeba vědět je aktuální stav a jeho možné akce:

$$\boldsymbol{vzorek = \left[R(s, a(s), s') + \gamma \max_{a'} Q_k(s', a') \right]}$$

Následně se provede aktualizace (update) TD učení:

$$\boldsymbol{Q(s, a) \leftarrow Q(s, a) + \alpha(vzorek - Q(s, a))}$$

$\boldsymbol{\alpha}$ nyní představuje rychlost učení (*learning rate*).

Q-Learning postup

1. navštiv nový uzel a ze vzorku (s, a, s', r) vypočti novou hodnotu jeho odhadu $Q_{k+1}(s, a)$ jako:
   průměr očekavných užitků akcí stavu $T *$ (okamžitá odměna + nejlepší možná Q-hodnota následníka), tedy:

   $$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

   průměr T a hodnoty odměn se postupně počítají na základě akcí, algoritmus se tedy blíží následující rovnici:

   $$Q(s, a) \approx r + \gamma \max_{a'} Q_k(s', a')$$

   r je přímá odměna ze stavu

2. udělej update TD učení (průměr odhadu vůči vzorku)

   $$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') \right]$$

Q-Learning konverguje k optimální strategii, i když následuje posloupnost neoptimálních akcí. Tento jev se nazýva *off-policy learning*. Jak ale agent vybírá, kterou akci provede, aby maximalizoval svůj užitek z tréninku? Je nutné zvolit dobrý poměr mezi zjišťováním terénu (*exporation*) a využívání již zjištěných dobrých strategií (*expoitation*). Dobrý poměr lze řešit několika způsoby:

- zavedení koeficientu $\varepsilon - greedy$ pro výběr náhodných akcí. Výhodné je ovšem koeficient časem zmenšovat, aby nedocházelo k nelogičnostem akcí.
- využití explorační funkce, aby se prozkoumaly stavy, jejichž špatná hodnota ještě není úplně jasná (málo výskytů) a nakonec s tímto skončit, výsledný Q-Update potom vypadá:

  $$Q(s, a) \leftarrow_{\alpha} R(s, a, s') + \gamma \max_{a'} f(Q(s', a'), N(s', a'))$$

  Q-Update se následně propaguje dál.

Nový pojmem pro metody posilovaného učení je **lítost** (*regret*) jako míra toho, kolik mě stály všechny chyby během tréninku (rozdíl mezi očekávanými odměnami a optimálními odměnami). Je snaha tuto míru minimalizovat, tedy se vlastně optimálně učit optimální řešení daného problému.

3.4.2 Approximate Q-Learning

Základní Q-Learning pomohl zbavit ze zbytečného přepočítávání hodnot, ale stále si musí udržovat tabulku všech Q-hodnot. Pro složitou úlohu jakou je Ms.Pacman je tato velká prostorová složitost stavového prostoru stále nevhodná. Z toho důvodu je potřeba **generalizovat**: naučit se q-stavy malého množství tréninkových stavů a hodnoty generalizovat na podobných situacích. To je důležitá myšlenka metody aproximační Q learning a potažmo i strojového učení [1]:

popsání stavu pomocí vektoru vlastností (*properties*), vlastosti jsou funkce ze stavý do reálných čísel(např. 0,1), které zachycuji důležité vlastnosti stavu. Následně lze tyto vlastnosti ohonodnotit **váhou** (*weight*) a sesavit lineární rovnice:

$$\boldsymbol{V(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s)}$$

$$\boldsymbol{Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)}$$

Nevýhoda takové generalizace je ale fakt, že je potřeba stanovit co nejlepší vektor vlastností, tak aby se hodnoty rozdílných stavů nepodobaly.

Aproximační Q-learning postup

1. navštiv nový uzel a ze vzorku $\boldsymbol{(s, a, s', r)}$ a vypočti rozdíl odhadu Q-hodnoty vůči vzorku:
$$\boldsymbol{rozdil = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s, a)}$$

2. udělej update TD učení (průměr odhadu vůči vzorku)
$$\boldsymbol{Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a')\right]}$$

3. následně místo updatu jedné $\boldsymbol{Q}$-hodnoty (základní Q-Learning), aproximuj $\boldsymbol{Q}$-hodnoty pomocí změny váhy $\boldsymbol{w_i}$ vůči vektoru vlastností:
$$\boldsymbol{w_i(s) \leftarrow w_i + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right] f_i(s, a)}$$
kde:
   - $\boldsymbol{\alpha\,[r + \gamma \max_{a'} Q(s', a')]}$ reprezentuje cíl (maximální), kterého se snaží daný stav dosáhnout
   - $\boldsymbol{\alpha\,[Q(s', a')]}$ předpokládaná Q-hodnota dané akce $\boldsymbol{a}$ daného stavu $\boldsymbol{s}$
   - $\boldsymbol{f_i(s, a)}$ reprezentuje lineární funkci, jejíž rozdíl je aproximován[4], což napomáhá možnosti využití i nelineárních funkcí ve vektoru vlastností.
   Zjednodušeně řečeno např. u Ms.Pacman: Pokud Ms.Pacman narazí do ducha, zvětší se váha situace ohrožení Ms.Pacman duchem. Pokud se tedy stane nějaká pozitivní akce, pozitivní váha zvětší hodnoty vlastností ve vektoru vlastností a naopak.

[4] K aproximaci se používá obecná **Metoda nejmenších čtverců, tzv. lineární regrese**

Kapitola 4

Ms.Pacman

4.1 O hře a důvod jejího výběru

Ms.Pacman je arkádová hra z roku 1982, která vylepšuje původní koncept hry Pacman. Hra je obohacena o nový design hlavní postavy, mapy a především implementuje lepší umělou inteligenci nepřátel. Hlavní postavou ovládanou hráčem je nyní Ms. Pacman, nepřátelé zůstávají v podobě 4 barevných duchů. Cílem hry je dosáhnout co největšího skóre pojídáním kuliček, duchů (časově omezené; po snězení speciální větší kuličky tzv. power-upu), nebo ovoce. Zvýšení obtížnosti oproti Pacmanovi spočívá především v **opuštění deterministického chování duchů**, čímž pádem nelze předpovídat jejich chování. Tento fakt spolu se složitostí hry jsou hlavními faktory proč se na hru zaměřit z hlediska strojového učení.

4.2 Pravidla

Pro účely této práce se bere v potaz zjednodušení původní hry na vizuální demo tak, aby byla dobře vidět účinnost jednotlivých algoritmů a příliš vysoké nároky na výpočetní výkon. Demo není rozděleno na levely a je založeno na umělé inteligenci obou stran (Ms. Pacman vs 2 duchové). Cíl Ms.Pacman je maximalizovat skóre na aktuální mapě, zatímco cíl nepřátel je její skóre minimalizovat. Stochastické chování nepřátel je simulováno tak, aby co nejvíce odpovídalo svému vzoru (zdrojové kódy hry nejsou veřejné, tudíž nelze simulovat chování duchů naprosto stejně jako v originálu). Duchové se snaží chytit a zabít Ms.Pacman. Pokud se jim to povede, hra končí. Skóre Ms.Pacman se navyšuje jezením kuliček, power-upů a následným jezením duchů po omezený čas trvání power-upu. Bonusová ovoce se neberou v potaz. Skóre se snižuje po dobu chození Ms.Pacman po bludišti bez jezení ovoce/duchů. Mapy se též liší, jsou menší a obsahují pro umělou inteligenci Ms.Pacman problémová zákoutí, jako větší neohraničené plochy atp. Ms.Pacman a duchové mají stejnou rychlost s výjimkou již zmíněnené situace, kdy Ms.Pacman sní power-up a může po omezený čas jíst duchy. Tato zjednošení slouží především ke snížení vypočetní náročnosti algorimu na běžně dostupný hardware.

4.3 Klasifikace dema Ms.Pacman z hlediska Teorie her

Ms. Pacman je hra, tzn. strategická interakce mezi 2 a více hráči, kteří chtějí dosáhnout optimálního výsledku ve hře. Vizuální demo Ms. Pacman lze specifikovat podle několika

kritérií:

- **dle informovanosti hráče o hře:** *hra s neúplnou informací* (*game with uncertainty*)- duchové mají stochastické chování
- **dle počtu tahů:** *hra strategická* – hráči provádějí souběžná rozhodnutí
- **dle míry konkurence/typu výhry:** *hra s nulovým součtem* (*zero-sum game*) - jeden hráč maximalizuje svou výhru a druhý minimalizuje ztrátu (míra výhry jednoho značí míru prohry druhého hráče)
- **dle počtu hráčů:** Ms. Pacman je maximizér a 2 duchové jsou minimizéři
- **dle komunikace hráčů:** *nekooperativní hra* – Ms.Pacman a duchové spolu nekomunikují a souběžně hrají proti sobě (striktně kompetitivní hra)
- **dle zdrojů:** *antagonistický konflikt* – Ms. Pacman a duchové sdílí jedno pevné maximální skóre dle herního plánu: *grid-world game* – vše probíhá diskrétně, po polích na mřížce v herním bludišti, jež mohou nabývat více stavů: prázdná cesta, zeď, prázdná cesta s hráčem, cesta s kuličkou, cesta s power-upem

Kapitola 5

Návrh a plán práce

Tato kapitola se věnuje praktické části práce a to především **návrhu chování agenta**. Po nastudování teorie je důležité poupravit cíl práce: místo porovnávání metod Minimax a Alfa-Beta Řezů s učicím agentem, bude lepší agenta provnávat s Expectimaxem, který bere v potaz stochastičnost akcí duchů.

5.1 Aproximační Q-Learning

Následující sekce řeší rozvrh práce pro návrh algoritmu učíčícho agenta. První důležitou částí návrhu řešení je kvalitně definovat **vektor vlastností**, který by měl korektně předpovídat $\boldsymbol{Q - hodnoty}$ jednotlivých stavů:

- vzdálenost nejbližšího ducha
- vzdálenost nejbližší kuličky jídla
- vzdálenost nejbližšího powerupu/vzdálenost nejbližšího ducha
- počet duchů
- je Ms.Pacman v rohu?
- je Ms.Pacman v tunelu?
- lze ducha sníst a je do vzdálenosti 4 políček?

Následně se stanoví návrh aproximační funkce vektoru na základě váhy každé vlastnosti.

Poté bude především potřeba nalézt co nejvíce **optimální strategii**, která nebude pouze založená na již zmíněné metodě Aproximační Q-learning:

1. aproximačně najdi model Q-hodnot
2. předpokládej pouze Q-hodnoty akcí z daného stavu
3. opakuj předchozí body dokud nebude dost vzorků, jinak pokračuj na další bod
4. poměňuj váhy vlastností a testuj ompimalitu nové strategie

POměňování váh vlastnutí takto může **maximalizovat odměny** a dojít tak lepší strategii, něž byla doposud použita.

Kapitola 6

Závěr

Cílem toho semestrálního projektu bylo nastudovat problematiku metod hraní her ve vztahu k učícímu agentovi pro hru Ms.Pacman a navrhnout metodu řešení problému. Ms.Pacman. Nastudování teorie pomohlo především k definici a klasifikaci samotné hry jako, a následném stanovení návrhu řešení problematiky metodou Aproximační Q-Learning. Tato metoda řeší stochastické chování duchů a pomůže výrazně eliminovat problém velké výpočetní náročnosti stavového prostoru Ms.Pacman. Jak si metoda s takto složitou problematikou poradí bude nejlépe vidět při srovnání její efektivity například s metodou Expectimax. Práce mě naučila především nebát se složitějších matematických vzorců a problému a též určitý vhled do možností strojového učení.

Literatura

[1] Busoniu, L.; Babuška, R.; Schutter, B. D.; aj.: *Reinforcement learning and dynamic programming using function approximators*. CRC Press, 2010, iSBN 978-1439821084.

[2] KLein, D.; Abbeel, P.; col.: BerkeleyX: CS188x_1 Artificial Intelligence[online]. https://courses.edx.org/courses/BerkeleyX/CS188x_1/1T2013/, 2016-01-15 [cit. 2015-01-17].

[3] Mařík, V.; Štěpánková, O.; Lažanský, J.; aj.: *Umělá inteligence (1)*. Academia, 1993, iSBN 80-200-0496-3.

[4] Mařík, V.; Štěpánková, O.; Lažanský, J.; aj.: *Umělá inteligence (3)*. Academia, 2001, iSBN 80-200-0472-6.

[5] Shoham, Y.; Leyton-Brown, K.: *MULTIAGENT SYSTEMS: Algorithmic, Game Theoretic, and Logical Foundations*. Cambridge University Press, 2008, iSBN 978-0-521-89943-7.

[6] Sutton, R. S.; Barto, A. G.: *Reinforcement Learning: An Introduction*. The MIT Press Cambridge, 1998, iSBN 978-0262193986.