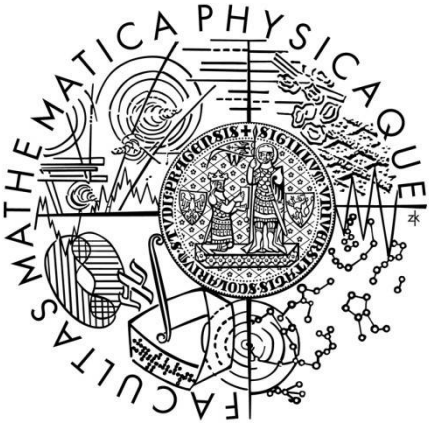
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Michal Preuss

Výpočetní model zvířete určený pro výuku na SŠ

Kabinet software a výuky informatiky

Vedoucí diplomové práce: Mgr. Cyril Brom, Ph.D.

Studijní program:
Informatika

Studijní obor:
Teoretická informatika

Praha

2011

Na tomto místě bych chtěl poděkovat Mgr. Cyrilu Bromovi, Ph.D. za vedení a cenné připomínky. Chtěl bych poděkovat i MUDr. Danielu Klementovi a Františku Šustovi, za poskytnutí odborného náhledu na řešené problémy. Bez přispění všech jmenovaných by tato práce nebyla vznikla.

Předmětem této práce je výpočetní model učení zvířat. Tento model byl následně vyzkoušen na třech virtuálních zvířatech - psovi, lemurovi a papouškovi - a implementován ve výukové počítačové hře Chovatel zvířat. Hra samotná není předmětem této práce, je však přiložena, aby čtenář práce mohl vidět "model v praxi". Výuková hra byla vytvořena Generation Europe, o. s. Za její zapůjčení pro účely této práce rovněž děkuji.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 8.12.2011 Michal Preuss

Název práce: Výpočetní model zvířete určený pro výuku na SŠ

Autor: Michal Preuss

Katedra (ústav): Kabinet software a výuky informatiky

Vedoucí diplomové práce: Mgr. Cyril Brom, Ph.D.

e-mail vedoucího: brom@ksvi.mff.cuni.cz

Abstrakt: V rámci této práce vznikl výpočetní model, jehož pomocí se dá simulovat učení zvířat. Model staví na variantě q-učení a umožňuje modelovat základní typy učení, včetně klasického a operantního podmiňování. Na rozdíl od jiných podobných modelů je popisovaný model vytvořen pro použití ve výukové počítačové simulaci: je speciálně navržen tak, aby učení probíhalo rychle a student mohl během krátké interakce s modelovaným zvířetem pozorovat dopady učení. Těžiště práce spočívá v popisu modelu, analýze jeho chování a porovnání s jinými modely. To se odehrává v kontextu tří virtuálních zvířat: psa, lemura a papouška.

Klíčová slova: virtuální laboratoř, výpočetní model, zpětnovazební učení, operantní podmiňování, shaping, klasické podmiňování

Title: A Computational Model of an Animal Designed for High-School Education

Author: Michal Preuss

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Cyril Brom, Ph.D.

Supervisor's e-mail address: brom@ksvi.mff.cuni.cz

Abstract: This work describes a computational model, which can be used to simulate learning in animals. Model is based on a variation of q-learning and can imitate general types of learning, such as classical and operant conditioning. It differs from similar computational models in that it is designed solely for use in educational software. Therefore the simulation of learning is rather fast, for student to be able to see the consequences of his own actions during a short training session. Main topic of this work lies in description of the model, analysis of its behavior and comparison with some other computational models. All of this takes place in the specific context of three virtual animals: a dog, a lemur and a parrot.

Keywords: virtual laboratory desk, computational model, reinforcement learning, operant conditioning, shaping, classical conditioning

Obsah

Úvod

V dnešní době, kdy jsou školy všech stupňů studia čím dál lépe technicky vybavené, přirozeně přibývá i programů určených k interaktivní výuce studentů. Tvorba takových programů s sebou ovšem přináší specifické problémy, vyplývající z toho, že výuka ve školách má pevný časový rámec, různé školy jsou technicky různě vybavené, učitelé musejí podstoupit školení v používání programu apod., viz např. [10].

Přesto je potenciální užitek podobných programů takový, že se vyplácí jejich tvorbou zabývat [1]. Pomocí specializovaných programů by totiž škola mohla poměrně snadno žákům zprostředkovat zkušenost jinak finančně nebo časově nemožnou, jako například provedení složitějšího fyzikálního, chemického nebo biologického experimentu. Takové programy navíc nemusí mít nutně přímo podobu virtuální laboratoře, jakou je např. program simulující krysu, Sniffy [2], se kterou uživatel může provádět laboratorní pokusy. Výukové programy mohou být i ve formě trenažéru nebo hry, jejíž hraní ovšem cíleně nutí hráče použít specifické znalosti. Hráč si tak tyto znalosti utvrzuje [19].

Ze srovnávací studie [12] ale vyplývá, že hry a instruktážní programy tohoto typu, jsou dodnes ve školství ojedinělé. Nejpatrnější je pak tento nedostatek u středních škol, na které je zaměřen pouhý zlomek podobných programů [6]. Hlavně proto vznikla iniciativa vytvořit program Chovatel zvířat (dále jen program), jehož hraním si uživatel vštěpuje základní techniky používané pro trénování zvířat.

Program je zaměřený na studenty střední školy a je navržen tak, aby ho bylo možné využít během vyučovací hodiny. Studenti v něm mohou za pomoci klasického a operantního [18] podmiňování cvičit virtuální zvířata vykonávat různé úkony. Program dává na výběr několik zvířat, která lze učit paletu cviků, jdoucí od jednotlivých pohybů, po sekvenci komplexních akcí. Obtížnější cviky ale lze zvíře naučit jen využitím reálných trénovacích postupů, které se tak student naučí používat.

Předmětem této práce ovšem není výše zmíněný program, ale návrh, implementace a analýza výpočetního modelu, který v programu simuluje učení zvířat. Netriviálních požadavků, které tento model musí splňovat, je totiž hned několik.

Zaprvé model musí reagovat podobně jako živé zvíře v dané situaci. V programu, ale i při jakémkoli jiném smysluplném použití modelu, je možné trénovat různé druhy zvířat (např. ve stávající verzi programu jde o psa, lemura a papouška). Obecně jde o druhy s velmi rozdílnou kognitivní kapacitou i vzorcem chování. Proto musí být navržený model natolik parametrizovatelný, aby byl schopen v závislosti na simulovaném zvířeti předvádět

odlišné reakce na podobné podněty od uživatele, simulovat typické chování pro dané zvíře, měnit obtížnost trénování atp.

Dále je potřeba, aby se zvíře simulované modelem dalo učit oběma základními typy podmiňování (klasické a operantní podmiňování [18]). Učení přitom musí probíhat rychle a doba, která je potřeba k naučení daného úkonu, musí jít přesně odhadnout. Program, který model využívá, je totiž určen do vyučovací hodiny s pevně daným a omezeným časovým rámcem.

Model také musí být natolik plausibilní, aby zkušenému uživateli programu (např. učiteli) stačil jednoduchý výpis hodnot k zjištění, jak je model naučený a jakým způsobem by se mělo postupovat při dalším učení.

Konečně musí být model parametrizovatelný, nebo jinak funkčně nastavitelný tak, aby realisticky zvládl odbornější tréninkové postupy, jako je „tvarování“ nebo „zachycení“ [14]. Jedině tak se na něm totiž dají předvádět složitější tréninky zvířat.

Cíle práce jsou tedy po řadě: návrh výpočetního modelu s výše popsanými vlastnostmi, jeho implementace a analýza toho, jestli model skutečně dané vlastnosti splňuje, případně do jaké míry je nesplňuje.

Kromě toho je součástí práce také shrnutí výsledků evaluace vlivu hraní s programem na znalosti studentů, kterou jsem provedl ve spolupráci s Mgr. Cyrilem Bromem, PhD. a MUDr. Danielem Klementem. Výsledky tohoto průzkumu vedly i ke vzniku dvou samostatných článků přijatých do impaktovaného žurnálu [6] a na lokální konferenci [15].

První kapitola popisuje vlastní program Chovatel zvířat z hlediska uživatele, typy tréninků, které pomocí něj lze simulovat, a konkrétní postup hry při trénování psa, lemura a papouška.

V druhé kapitole jsou analyzovány požadavky na vlastnosti modelu, které vyplývají ze specifikace hry v první kapitole. V kontextu těchto požadavků jsou pak porovnávány výhody i nedostatky vybraných učících algoritmů. Po této analýze následuje podrobný popis návrhu modelu a jeho implementací ve všech třech případech, používaných v programu (tj. implementace modelu psa, lemura a papouška).

Třetí kapitola obsahuje podrobné experimenty, které analyzují vlastnosti modelu a porovnávají je s požadovanými vlastnostmi, specifikovanými v druhé kapitole.

Ve čtvrté kapitole je popsán způsob evaluace vlivu hry s programem na znalosti studentů střední školy a jsou zde shrnuty její nejzajímavější výsledky.

Závěr práce diskutuje vhodnost použitého návrhu modelu ve světle analýzy jeho chování. Dále hodnotí i přesnost simulace učení jednotlivých zvířat, výsledky evaluace na studentech i možný další vývoj programu Chovatel zvířat nebo jiné užití modelu.

Kapitola 1

Program Chovatel zvířat

Jak už jsem se zmínil výše, hry zaměřené na výuku hráče jsou stále ojedinělým úkazem. To platí v biologii stejně nebo dokonce více, než v ostatních vědních disciplínách (četnost různých aplikací podporujících interaktivní výuku je adresována v [6]). Právě tento nedostatek se stal základním motivem ke vzniku výpočetního modelu, který je předmětem této práce, a na něj navazující aplikace Chovatel zvířat. Metodika použití hry (a z toho plynoucí požadavky na chování modelu) je navržena tak, aby se hráč seznámil nejen se základními typy podmiňování, ale v závislosti na druhu zvířete a úkonu, který má zvíře naučit, i s vybranými specifičtějšími technikami používanými k trénování živých zvířat.

Pro tuto práci je hra Chovatel zvířat zajímavá hlavně z toho důvodu, že na simulování učení virtuálních zvířat využívá autonomní výpočetní model, který jsem navrhl. Ostatní hry tohoto typu (s modelem simulujícím učení zvířete – např. Creatures [11]) jsou ale zaměřeny jen na zábavu hráče. Naopak skutečně výcvikové programy (virtuální laboratoře jako Sniffy [2] atp.) jsou zase pro neodborného uživatele nezáživné a přílišně složité. Obojí z programu Chovatel zvířat dělá zaměřením ve světě jedinečnou aplikaci.

V této kapitole popíši fungování programu z hlediska uživatele a tréninkové postupy, na které je hra s programem (a tedy i model) zaměřena. Z tohoto popisu pak vyplývají požadavky na vlastnosti modelu, které detailně analyzuje kapitola 2.

1.1 Hra s programem

Cílem hry je naučit vybrané zvíře udělat v reakci na stimul určitý úkon. Chovatel zvířat obsahuje 5 různých tréninkových prostředí, ve kterých lze trénovat psa, lemura nebo papouška[1]. V každém tréninkovém prostředí má uživatel k dispozici několik stimulů, jejichž prostřednictvím může interagovat se zvířetem (viz obrázek 1). Kromě stimulu odměny v podobě žrádla jsou všechny stimuly zvukové nebo vizuální a mají omezené trvání. Zvíře tedy lze ve hře odměňovat, ale protože není k dispozici žádný stimul, který by byl zvířeti nepříjemný, zvíře nejde trestat. Jde tedy o učení pozitivní zpětnou vazbou, které je obvyklé i u tréninku živých zvířat.

[1] Pes i papoušek jde trénovat pouze v jednom prostředí, lemur ve třech různých prostředích.

Konkrétní cíl tréninku si uživatel volí sám – může jít například o to, naučit psa zaštěkat na slovní povel, nebo naučit lemura přitáhnout provázek z natažené ruky virtuálního trenéra. Protože je program navržený na užití při výuce, předpokládá se, že cíl tréninku stanovuje učitel, a proto není ve hře pevně daný.

Při tréninku uživatel může s programem jakkoli experimentovat, ale výpočetní model simulující zvířata v programu by se měl učit takovým způsobem, že nejúčinnější postup ve hře vždy koresponduje s reálně používaným postupem. Splněním hry si tak uživatel prohloubí znalosti skutečně fungujících technik učení. Těmto technikám se budu blíže věnovat níže.

Model se také musí učit několikanásobně rychleji, než je obvyklé u živých zvířat. Trénink, který by normálně trval dny až týdny, by tak ve hře mohl trvat jen několik minut a hráč by se mohl rychle přesvědčit, jaké by měl jeho postup následky.

Prostředí, ve kterých se trénink odehrává, jsou velice jednoduchá a model ve většině případů nemusí bez zásahu uživatele s prostředím interagovat. Samostatná interakce modelu s prostředím musí být možná pouze při tréninku s lemurem, který je popsán níže.

Obr. 1: Chovatel zvířat – okno s prostředím pro trénování psa Kvída. Napravo tlačítka ovládající jednotlivé stimuly od uživatele. (c) 2009 GE

1.2 Tréninkové postupy

Hra je navržena tak, že všechna zvířata v ní lze učit stejnými základními technikami. Zde se zaměřím pouze na shrnutí informací, které jsou nutné pro porozumění správnému postupu hráče ve hře. Detailnější informace lze získat např. v [18] (klasické a operantní podmiňování), nebo [14] (specifičtější tréninkové techniky popsané níže).

Základní a nejběžnější typ podmiňování, který se používá ve hře, ale i na živých zvířatech, je operantní podmiňování [5]. Jde o využití zpětné vazby zvířete na odměnu nebo trest. Odměněné zvíře (např. jídlem) si právě vykonanou akci velice pravděpodobně spojí s příchodem odměny a ve snaze dostat další odměnu se frekvence vykonávání dané akce může zvýšit. Pro trest (např. slabý elektrický šok) toto platí obráceně. Ve hře ale není možné zvířata trestat, takže k učení modelu lze využít pouze pozitivní zpětnou vazbu.

Reálně přitom samozřejmě velice záleží na druhu zvířete, na době, která uplyne mezi vykonáním trénované akce a obdržením odměny, ale i na mnoha dalších faktorech. Nejdůležitější z nich musí být implementovány do modelu (více viz kapitola 2 a 3) a hráč by na ně měl pamatovat, pokud chce úspěšně hru dokončit.

Operantní podmiňování hráč ve hře musí být schopen používat v rámci tří tréninkových technik: „zachycení“, „tvarování“ a řetězení akcí [14]. Výpočetní model se musí učit takovým způsobem, aby tyto techniky reprezentovaly nejúčinnější strategii.

1.2.1 Zachycení chování

Nejjednodušší technika využívající operantní podmiňování spočívá prostě v tom, že cvičitel nebo hráč počká, až zvíře udělá danou akci samo od sebe. Když se tak stane, tuto akci odmění a zvíře si tak vštěpuje, že po této akci přichází odměna a pravděpodobně ji začne vykonávat častěji. Takto ale lze trénovat pouze akce, které zvíře alespoň občas dělá spontánně (ať už samo od sebe nebo v reakci na vnější stimul) a jde o tzv. „zachycení“ chování (catching of behavior - [14]).

1.2.2 Postupné učení – tvarování

Pokud chceme zvíře naučit něco, co by samo od sebe prakticky nikdy neudělalo, musíme výše zmíněnou techniku „zachycení“ poněkud upravit. Odměňovat začneme spontánní chování zvířete, které je nejpodobnější nebo má alespoň společné znaky s cílovou akcí. Tím docílíme toho, že zvíře toto chování začne předvádět častěji. Když ho ale náhle odměňovat přestaneme,

zvíře může ve snaze znovu dostat odměnu začít přehánět nebo pozměňovat předchozí pohyby. Z takto vzniklých nových chování si opět vybereme to nejpodobnější s naším cílem a odměnou ho „zachytíme“. Celý trénink končí tehdy, když za pomoci popsaných mezikroků chování zvířete „vytvarujeme“ (shaping of behavior - [14]) do cílové podoby.

Při „tvarování“ je důležité, aby chování trénovaná v po sobě jdoucích mezikrocích byla navzájem dostatečně podobná. Čím podobnější chování, tím je pro zvíře snazší přechod od jednoho ke druhému a tím méně opakování je potřeba, aby se zvíře další krok naučilo. Tak se také snižuje riziko přetrénování chování v nějakém mezikroku, což by mělo neblahé účinky na celý trénink.

1.2.3 Řetězení akcí

V případě, že cílovou akcí je složitý nebo dlouhotrvající pohyb, je učení „tvarováním“ samo o sobě jen těžko použitelné. Pokud ale cílovou akci lze rozdělit na sekvenci několika kratších (a pravděpodobně jednodušších) akcí, můžeme tyto akce učit odděleně. Po naučení všech akcí v cílové sekvenci pomocí „zachycení“ nebo „tvarování“ zbývá tyto akce propojit do jednoho celku. U živých zvířat je nejúčinnější začít řetězení pokud možno od poslední akce v sekvenci (na směru propojování záleží [14]). Na tu napojíme těsně předcházející akci v sekvenci. Jakmile jsou tyto dvě akce dostatečně propojené, přistoupíme k další předcházející akci. Takto postupujeme, až se propojí celá sekvence.

1.2.4 Klasické podmiňování

Kromě technik, využívajících operantní podmiňování, musí být ve hře možné stejně jako na živých zvířatech využít klasické (pavlovovské) podmiňování [18]. Musí tedy jít naučit zvíře, že určitý stimul předchází jinému, tj. tyto stimuly „spárovat“. Konkrétně například, že krátký zvuk předchází obdržení odměny. Takto asociované stimuly navíc pro zvíře nabývají stejného významu (zvuk má prakticky stejný účinek jako sama odměna). Asociace stimulů se ale musí v průběhu tréninku udržovat, takže pokud první ze stimulů již nadále není následován asociovaným stimulem, jejich spojení slábne, až se postupně vytrácí. Proto by v daném příkladu měla po zvuku i po asociaci často následovat skutečná odměna.

Ve hře je ovšem možnost vzniku asociace mezi stimuly v rámci zachování jednoduchosti požadovaná jen mezi zvukovým signálem a odměnou (viz sekce 2.5.1). U živých zvířat by s odměnou šly asociovat všechny stimuly.

1.3 Pes – příklad trénování

Prostředí se psem je navrženo primárně na procvičování tréninku „tvarováním“. Předpokládaným cílem hry je naučit psa na slovní povel zvednout packu do výše očí. Kromě „tvarování“ lze z ostatních výše zmíněných tréninkových postupů na psovi procvičovat ještě „zachycení“ a klasické podmiňování. Trénink řetězení akcí není vzhledem k povaze stimulů a akcí, které pes může vykonávat, možný. Kromě výše popsaných základních technik je dále specificky při tréninku se psem možné použít postupy, jako je přerušení akce odměnou nebo změny spouštěcího stimulu akce.

Hra má dvě obtížnosti – tj. uživatel si může vybrat z trénování dvou psů, Kvída a Huga. Na obrázku 1 je zobrazeno okno hry při trénování rychleji se učícího psa Kvída.

Předpokládaný a ke splnění cíle hry nejsnazší postup je takový, že hráč nejprve asociuje pomocí klasického (pavlovovského) podmiňování odměnu s krátkým zvukovým signálem. Hra sice musí jít úspěšně dohrát i bez tohoto kroku, ale asociací zvuku s odměnou hráč získá možnost rychle označit správně provedenou akci, což významně zefektivňuje trénování. Podávání skutečné odměny totiž trvá dlouho a není tedy jasné, za co přesně ji zvíře dostává. V dalším popisu postupu při hře budu tedy za odměnění považovat zvukový signál (ten je často, ale ne nutně vždy, následovaný odměnou).

Další kroky hry využívají operantního podmiňování a „tvarování“. Předpokládaný nejúčinnější postup učení je takový, že hráč nejprve krok po kroku naučí psa zvedat packu do výše hlavy pomocí stimulu ukázané ruky a pak změní spouštěcí stimul ukázané ruky na stimul slovního povelu.

Prvním krokem v postupném učení zvednutí packy je odměnění jakékoli akce, kterou pes vykoná za použití stimulu 3, dotknutí packy rukou (viz obrázek 1). V druhém kroku pak hráč použije stimul 4 - natažení ruky těsně nad packu psa. V tu chvíli hráč již neodměňuje všechny akce, ale čeká na specifickou akci, kterou odmění a tím posílí - „zachytí“. Touto akcí je mírné zvednutí packy a dotyk s nataženou rukou. Z hlediska živého psa je to jen pokus o to, dostat se k stimulu dotyku packy s rukou, který byl předtím odměňován.

Následující krok přesně odpovídá „tvarování“, protože hráč použije stimul 5 a zvedne ruku o něco výše než v předchozím kroku - psovi do výše hlavy. Potom přestane odměňovat jen mírné zvednutí packy a opět čeká s odměnou pouze na to, až se pes dotkne natažené ruky. Tímto postupem se docílí toho, že pes na stimul natažené ruky zvedne packu do výše hlavy. Pro splnění hry pak zbývá zaměnit spouštěcí stimul natažené ruky za slovní povel.

Záměna stimulů má tři fáze, problém je totiž nejen v tom, že chceme zaměnit spouštěcí stimuly. Pro psa totiž akce „dotknutí se packou natažené ruky“ nemá žádnou spojitost s akcí „zvednutí a držení packy v prostoru“. Proto nejprve oprostíme cílovou akci od fyzického kontaktu s rukou. To uděláme tak, že zdánlivě pokračujeme jako doposud v „tvarování“ a použijeme stimul 6, ruku nataženou vysoko nad psem. Protože je v tomto případě ruka natažená mimo dosah psa, ten nejprve zkusí packu zvednout do původně naučené výšky (tj. do výšky své hlavy). Pokud by v tu chvíli hráč nic neudělal, pes se začne snažit natažené ruky dotknout a vyskočí po ní. Aby tomu hráč zabránil, odmění psa právě ve chvíli, kdy je jeho packa napřažená ve výšce hlavy. Tím přeruší snahu dotknout se ruky (akce již byla odměněna, tím pádem nemá smysl jí dokončovat) a posune spouštěcí stimul od dotyku s rukou pouze k pohledu na nataženou ruku.

Samotné zaměnění vizuálního stimulu natažené ruky za zvukový stimul povelu má dvě fáze. Nejprve hráč začne předsazovat zvukový povel každému použití vizuálního stimulu s rukou. Důležité je pořadí stimulů, kdy nový, zvířeti neznámý stimul musí předcházet již známému spouštěcímu stimulu, jinak je asociace stimulů velice obtížná (jedná se vlastně o klasické podmiňování).

Po několika opakováních stimul natažené ruky můžeme vypustit a správně naučený pes by měl pouze na slovní povel zvednout packu. Toto chování „zachytíme“ a posílíme tak, aby byla jeho frekvence co nejvyšší. Tím je trénink se psem úspěšně u konce.

1.4 Lemur – příklad trénování

Trénink s lemurem lze v programu provádět ve třech prostředích, která jsou ovšem propojená a uživatel mezi nimi může libovolně přecházet bez vlivu na simulaci (tlačítka vlevo dole na obrázku 2). V každém z těchto prostředí uživatel disponuje různými stimuly, takže jejich počet je výrazně vyšší než u ostatních zvířat. Lemur navíc může vykonávat více různých akcí, což se hodí pro trénink řetězení akcí.

Hra má opět dvě obtížnosti – tj. uživatel si může vybrat z trénování dvou lemurů, Emila a Kamila. Na obrázku 2 je zobrazeno okno hry při trénování rychleji se učícího lemura Emila v prvním prostředí.

Předpokládaným cílem hry je naučení sekvence akcí, při které lemur vleze do bedny a sám se v ní zavře. Tato sekvence se dá rozdělit na tři nezávislé akce (vlezení do bedny, uchopení provázku připevněného ke dveřím a přitáhnutí provázku). Tedy jde o trénink

využívající řetězení akcí. Sekvence se spouští tím, že uživatel před lemura položí bednu s kladkou.

Umístění bedny před lemura je ovšem v kontextu hry speciální stimul. Bedna totiž na rozdíl od všech ostatních stimulů před lemurem zůstává, dokud se jí uživatel nerozhodne vzít zpět. Proto jde spíš o součást prostředí, se kterou ale lemur může interagovat. Co existence takového stimulu znamená pro výpočetní model, rozebírá kapitola 2.

Trénink je v případě lemura několikrát delší než u psa, a tak ho zde - na rozdíl od tréninku se psem - popíši jen zkráceně. Detailněji je postup tréninku popsán v uživatelské dokumentaci programu Chovatel zvířat, která je přiložena.

Nejúčinnější postup tréninku začíná podobně jako u tréninku se psem asociováním zvukového stimulu s odměnou. Tento krok opět není nutný, ale výrazně ovlivňuje rychlost učení ve všech dalších krocích.

Dále hráč přistupuje k první fázi řetězení, která spočívá v tom, že se lemur naučí odděleně vykonávat všechny akce sekvence. Přitom akce vlezení do bedny se učí pomocí „zachycení“ a akce „uchopení provázku“ a „přitažení provázku k sobě“ se spojí do jedné složitější akce, kterou ale lze naučit pomocí „tvarování“. Právě prostředí, ve kterém se lemur pomocí „tvarování“ učí chytit a přitáhnout provázek, je zobrazeno na obrázku 2.

Po odděleném naučení obou akcí zbývá tyto akce spojit do jedné sekvence. To ale není vůbec jednoduché, protože podle 1.2.3 je řetězení akcí u živých zvířat efektivní jen pokud se na poslední akci sekvence (v našem případě chycení a přitažení provázku z bedny) postupně navazují předcházející akce. To ale není v tomto případě dost dobře možné, protože lemur se nejprve musí do bedny dostat, aby z ní mohl přitáhnout provázek. Navíc jakkoli může být lemur naučený chytit a přitáhnout provázek z ruky trenéra, přitažení provázku v bedně je pro něj natolik odlišné, že to sám od sebe ani nevyzkouší.

Z těchto dvou důvodů spojení akcí do sekvence probíhá tak, že se akce „vlezení do bedny“ rozloží na několik fází. Používá se přitom nejprve průchozí bedna bez dna, dále normální bedna a konečně bedna s kladkou a provázkem. V první fázi (s průchozí bednou) hráč nastaví ruku s provázkem stejně jako při tréninku bez bedny. Protože je ale bedna průchozí a provázek je v ruce hráče, naučený lemur si tuto situaci poměrně snadno asociuje s případem, kdy bedna vůbec není přítomna. Dále už hráč postupuje „tvarováním“ a používá postupně bednu se dnem a bednu s kladkou. Lemur se vlastně znovu učí akci „vlezení do bedny“ s tím rozdílem, že následně v bedně vždy přitahuje nastavený provázek. Toto učení navíc postupuje velice rychle, protože lemur se již předtím učil vlézt do bedny i přitáhnout provázek z ruky hráče.

Jakmile se lemur postupnými kroky naučí vlézt do bedny s provázkem připojeným na kladku, hráč přestává nastavovat ruku držící provázek a jen přikládá prázdnou ruku ke kladce. Lemur si díky přítomnosti ruky hráče všimne provázku a začne ho přitahovat. Když se naučí tento krok, hráč konečně může přestat nastavovat i pouze prázdnou ruku a lemur přitahuje provázek zavěšený na kladce sám od sebe.

Obr. 2: Chovatel zvířat – okno s prvním prostředím pro trénování lemura jménem Emil. Napravo tlačítka ovládající jednotlivé stimuly od uživatele. Vlevo dole přepínání prostředí. (c) 2009 GE

1.5 Papoušek – příklad trénování

Z hlediska počtu různých stimulů, které může hráč při tréninku používat, je nejomezenějším zvířetem v programu papoušek (viz obrázek 3). Proto se trénink s papouškem nehodí pro techniku „tvarování“, která potřebuje pro každý další mezikrok nový stimul, ani řetězení. Papoušek ale umí vykonat mnoho různých akcí, z nichž některé jsou si velice podobné. To z něj dělá ideální zvíře pro trénink „zachycením“.

Hra má stejně jako v předchozích případech dvě obtížnosti, takže uživatel si může vybrat mezi dvěma papoušky – Alexem a Alfonsem. Na obrázku 3 je zobrazeno okno hry při trénování rychleji se učícího papouška Alexe.

Předpokládaným cílem hry je zde naučit papouška vyslovit určitou krátkou větu na slovní povel od uživatele (stimul 4 na obrázku 3) a vyslovit jinou větu na jiný slovní povel (stimul 5 na obrázku 3).

Tento trénink není triviální, protože papoušek často vyslovuje jen části cílových vět, nebo vyslovuje již naučenou větu v reakci na oba slovní povely. Chybné odměnění papouška pak vede k opakování nechtěných akcí a znatelně prodlužuje trénink.

Stejně jako u psa a lemura, i u papouška platí, že pokud hráč před vlastním tréninkem asociuje odměnu se zvukovým signálem pomocí pavlovovského podmiňování, trénink probíhá znatelně rychleji.

Obr. 3: Chovatel zvířat – okno s prostředím pro trénování papouška Alexe. Napravo tlačítka ovládající jednotlivé stimuly od uživatele. (c) 2009 GE

1.6 Zaměření programu

Simulace zvířat v programu je do jisté míry stochastická (více viz kapitola 2), přesto by měla při trénování produkovat opakovatelné výsledky – mělo by se tedy jednat o jistou formu virtuální laboratoře. Přitom ale hra musí být uživatelsky co nejméně náročná, aby jí bez problému mohlo ovládat i dítě. Protože právě na výuku dětí a dospívajících je program primárně zaměřen.

Reakce simulovaných zvířat a postupy při doporučených trénincích zmíněných výše jsou ovšem koncipovány natolik komplexně, že by program měl být vhodný dokonce i jako trenažér pro cvičitele nebo vychovatele zvířat a podobné profese.

Kapitola 2

Model

Výpočetní model, který je předmětem této práce, je speciálně navržen pro aplikaci Chovatel zvířat. V předchozí kapitole je popsána předpokládaná funkce aplikace z hlediska hráče. Z té na model vyplývají určité požadavky. V této kapitole nejprve přesně tyto požadavky na model formuluji.

Následně zmíním několik známých technik strojového učení a budu diskutovat vhodnost jejich použití vzhledem k popsaným požadavkům. Tím nastíním důvody, které mě vedly k návrhu modelu, který jsem nakonec použil.

V sekci 2.3 je můj návrh modelu detailně popsán a v dalších sekcích této kapitoly je podrobně rozebíráno fungování modelu jak abstraktně, tak na několika konkrétních příkladech. Protože je model obecně široce parametrizovatelný, jsou zde také specifikována a diskutována základní nastavení modelu pro všechny druhy zvířat, která jsou simulována v aplikaci Chovatel zvířat.

V následující kapitole pak pomocí experimentů s naimplementovaným modelem zjišťuji, do jaké míry byla schopna vybraná architektura modelu splnit požadavky, na ní kladené.

2.1 Požadavky na model

Hlavní požadavek na model je jeho variabilita. Model musí být schopen dobře simulovat učení všech zvířat v programu. Ta se přitom musejí být schopna realisticky učit všemi způsoby popsanými v kapitole 1. Na druhou stranu ale model nemusí být nijak biologicky plausibilní, protože pro uživatele zůstávají veškeré jeho funkce skryté a tato práce si nedává za cíl výzkum příčin chování zvířat.

Naopak, předností modelu musí být jeho jednoduchost, kdy by podle výpisu několika základních hodnot mělo být možno zjistit příčiny aktuálního chování modelu. To je společně s možností částečného resetování modelu jednou z vlastností, které jsou zásadní pro správný průběh např. vyučování v početnějších třídách.

Další vlastností modelu, která je nutná kvůli použití programu ve vyučovacích hodinách, je možnost odhadnout dobu, nutnou k naučení stanoveného úkonu. Experimenty s modelem musejí být opakovatelné do té míry, že pokud uživatel při tréninku neudělá

žádnou, nebo jen omezený počet chyb, musí být možné předem zhruba[2] odhadnout, jak dlouho bude trénink trvat. Díky tomu bude moci učitel poměrně lehce naplánovat průběh hodiny.

Požadavek na odhadnutelnost celkové doby tréninku je ale příliš obecný a závisí hlavně na zkušenostech uživatele. Navíc k provedení takového odhadu je třeba nejprve odhadnout délku jednotlivých kroků, které jsou v tréninku použity. Proto by model měl při daném nastavení několika parametrů garantovat počet opakování nutných k splnění menších celků učení. Tj. například mezikrok učení „tvarováním“ by měl jít naučit po třech opakováních, „zachycená“ akce po dvou opakováních apod. Příslušné nastavení modelu by pak záviselo na druhu simulovaného zvířete nebo na zvolené obtížnosti tréninku.

Rychlost tréninku také musí být výrazně ovlivněna tím, jestli uživatel používá zvukový stimul asociovaný s odměnou, či nikoli. Běžný trénink ale zároveň musí být možné dokončit i bez použití zvukového stimulu.

Aby měl program co největší výukový potenciál, je dále třeba, aby model výrazně trestal první chybu uživatele. Tak je hráč ihned na svou chybu upozorněn a může si snadno uvědomit, kde ji udělal. Další chyby ve stejném místě (tj. při učení stejného kroku) již naopak nesmí být trestány s takovou razancí, aby i soustavně chybující hráč mohl v tréninku postupovat, pokud mu je vysvětlen správný postup.

Vzhledem k ojedinělosti programu Chovatel zvířat a rozvoji školních počítačových učeben se v budoucnosti mohou stále častěji objevovat podobné programy. Proto je žádoucí, aby navržený model byl snadno rozšiřitelný až do té míry, že by například nový druh zvířete nebo tréninku šel přidat pouze ve formě nového konfiguračního souboru. Odhalit, do jaké míry je toto možné, je dalším z cílů návrhu modelu.

2.2 Výběr architektury

V úloze simulování zvířete, které se učí pouze interakcí s uživatelem, jde o učení bez učitele (unsupervised learning [17]). Pokud za předpoklad vezmeme použití diskrétního času, lze chování výpočetního modelu simulujícího zvíře popsat jako markovův rozhodovací proces [4]. Tedy konkrétně existuje konečný počet stavů, ve kterých se model může nacházet, a konečný počet akcí, které v těchto stavech může model vykonat. V každém časovém kroku t se model nachází ve stavu s_t a z možných akcí se rozhoduje udělat novou akci a_t.

[2] Tedy např. v řádu minut. Jednotlivé tréninky popsané v kapitole 1 při správném postupu trvají průměrně 3-10 minut. Model musí mít jasně definován rozptyl trvání tréninku v závislosti na počtu chyb hráče.

S pravděpodobností $P_a(s, s') = P(\, s_{t+1} = s' \mid s_t = s,\ a_t = a\,)$ se pak model dostává do nového stavu s'a obdrží přitom odměnu $R_a(s, s')$.

O rozhodnutí, jakou akci v daném stavu vykonat, se může starat libovolný algoritmus strojového učení, který k adaptaci používá právě obdrženou odměnu $R_a(s, s')$.

Drobný problém této formalizace spočívá v tom, že zvíře ve hře musí umět vykonávat různě dlouhé akce. Délka akce je ale vždy předem známá, a tak budu tento problém nadále řešit tím způsobem, že k přechodu mezi stavy bude docházet ihned po výběru nové akce, ale další akce se bude vybírat až po úplném vykonání předchozí akce.

2.2.1 Hlavní problémy učení

Největší problémy, se kterými se učící algoritmus modelu musí vyrovnat, plynou z toho, že zvíře v programu neinteraguje s prostředím, ale vlastně pouze s uživatelem. Ten používáním stimulů přímo ovlivňuje aktuální stav zvířete. Navíc i cílová akce se v průběhu tréninku může měnit. Z obou těchto důvodů jsou pravděpodobnosti přechodů mezi stavy $P_a(s, s')$ neznámé, rychle se mění a obsahují šum.

Dalším problémem je integrace klasického podmiňování, kterým lze asociovat zvukový signál s odměnou. Aby mělo užití správně asociovaného zvukového signálu patřičný efekt, je třeba při jeho použití přímo odměnit akci, která právě probíhá. To ovšem znamená, že ani odměny $R_a(s, s')$ nemohou být pevně dané.

Oba tyto problémy naznačují, že jde o problém spadající pod zpětnovazební učení [17].

2.2.2 Použitelné učící algoritmy

Model by neměl vybírat novou akci deterministicky. V programu jsou totiž jednotlivé tréninky relativně omezené, s malým počtem stavů i možných akcí. Pokud by deterministický model byl jednoduchý, uživatel by mohl díky malému počtu stavů chování modelu odhadovat. Navíc simulovaná zvířata by se v takovém případě nechovala realisticky. Naopak použití komplexního deterministického modelu, by zase šlo proti požadavku na jednoduchost a snadnou parametrizovatelnost modelu. Takový model by si musel udržovat například míru koncentrace zvířete, jeho okamžitou motivaci, náladu a další parametry, které by ovlivňovaly výběr akce v daném stavu.

Na druhou stranu jednoduchý stochastický model se chováním v omezeném prostředí s omezeným počtem možných akcí dostatečně přibližuje živému zvířeti [9].

Vzhledem k požadavkům na model formulovaných v sekci 2.1, ale i vzhledem k popisu běžných tréninků v kapitole 1, model nemusí být schopen složitě plánovat daleko dopředu. Proto jsem se rozhodl nepoužít učení časovými rozdíly (temporal difference learning [16]), které se jinak na simulování chování zvířat používá nejčastěji. I přes svoji komplexnost se totiž tento způsob učení stále těžko vyrovnává s větším šumem při obdržování odměny, viz [20].

V zájmu toho, aby byl model co možná nejjednodušší, jsem si nakonec za učící algoritmus zvolil q-učení [22]. To podobně jako učení časovými rozdíly naráží na problém šumu při dostávání odměny, protože odhady odměny za vykonání akce nemohou být přesné. Modifikace q-učení popsaná v [13] se ale za užití technik výše zmíněného učení časovými rozdíly (TD learning) tento problém snaží řešit.

Proto jsem při návrhu svého modelu vycházel právě z modelu popsaného v [13].

Pro účely hry Chovatel zvířat byl ale model [13] třeba v několika ohledech upravit. Nejpoužívanějším typem tréninku ve hře je totiž trénink „tvarováním“, který by původní model nezvládl, ale mnou navržený model ho zvládá díky úpravám popsaným v sekci 2.4.

Dále je model popsaný v [13] pro potřeby hry příliš složitý, protože se snaží simulovat i motivaci zvířat. V tomto ohledu je mnou navržený model značně jednodušší a počítá s neměnnou motivací zvířete během celého tréninku. To je ovšem vzhledem k předpokládané průměrné délce tréninku v programu srovnatelné s reálným tréninkem.

2.3 Popis modelu

Jak bylo ukázáno výše, činnost modelu lze popsat ve smyslu markovova rozhodovacího procesu [4]. Protože je ale pravděpodobnost přechodu mezi stavy neznámá, nepravidelně se mění a obsahuje šum, její hodnoty model nebere v potaz.

Uživatel navíc může ovlivňovat stav modelu pouze prostřednictvím předem definovaných stimulů.

Model tak lze popsat jako uspořádanou šestici (*S, A, Stim, r, C, P*), kde *S* je konečná množina stavů modelu, *A* je konečná množina akcí a *Stim* je konečná množina stimulů. Dále $r\colon S \times A \to [0,1]$ je funkce udávající okamžitou odměnu obdrženou po vykonání určité akce v určitém stavu, *C* je množina konstant ovlivňující vztahy mezi akcemi a efektivitu učení, ke které se v popisu vracím níže. Konečně *P* je matice pravděpodobností vykonání všech akcí ve všech stavech – tedy pokud je $S = \{s_0, s_1, \dots, s_m\}$ množina všech stavů a $A = \{a_0, a_1, \dots, a_n\}$ množina všech akcí, pak je *P* matice typu (*m*+1) *x* (*n*+1) a pro každý stav s_i

platí $\sum_{j=0}^{n} P_{ij} = 1$, tj. $P_{ij} = P(\, a_j \mid s_i \,)$. Akci a_j budu dále nazývat možnou ve stavu s_i, pokud $P_{ij} > 0$. Jinak budu akci a_j ve stavu s_i nazývat nemožnou.

Protože v programu nedochází k interakci simulovaných zvířat navzájem, jediným vstupem modelu jsou stimuly od uživatele a okamžitá hodnota odměny *r*. Samotná činnost modelu pak spočívá v tom, že model v diskrétních časových krocích vyhodnocuje svůj vstup a za použití pravděpodobností z *P* vybírá další akci.

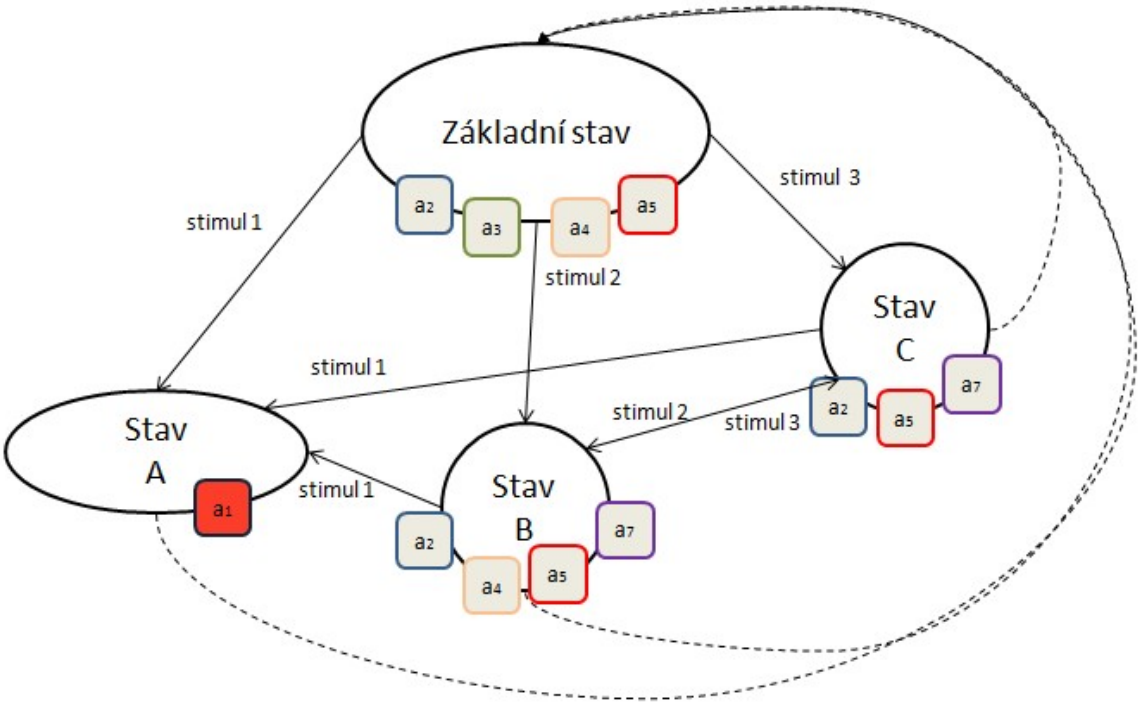
Každá akce $a \in A$ má ale různě dlouhou dobu vykonávání. Model přesto není řízen událostmi skončení akce, protože probíhající akce může být za určitých podmínek přerušena (více o přerušení akce viz níže a 2.5.2).

Jak jsem již zmínil výše, uživatel může do běhu modelu zasahovat prostřednictvím stimulů, které mění jeho aktuální stav a přerušují právě vykonávanou akci. Přesněji je každý ze stimulů asociován právě s jedním stavem (tj. $|S| \geq |Stim|$) a použitím daného stimulu se model uvede do asociovaného stavu, kde se okamžitě vybírá nová akce k vykonání. V zájmu zjednodušení byly ale v programu některé pro trénink důležité akce navrženy tak, že za dobu jejich trvání uživatel nemůže použít stimul a tak je přerušit. Více o možnostech přerušení akce pojednává sekce 2.5.2.

Diagram na obrázku 4 zobrazuje běžné rozložení stavů, možných akcí a stimulů. Stav s_0, ve kterém model začíná, je deklarován jako základní. Z tohoto stavu se model bez zásahu uživatele nemůže dostat a navíc se do něj automaticky vrací po uplynutí určité doby od posledního stimulu (viz obrázek 4). Jaký stav se považuje za základní, se ale může během simulace měnit (viz sekce 2.6.2).

V příkladě na obrázku 4 odměnu model získává vykonáním akce a_1 ($\forall\, s \in S$: $r(s, a_1) > 0$). Akci a_1 ale lze vykonat jen ve stavu A, což není základní stav, takže se do něj model může dostat pouze za pomoci stimulu od uživatele. Když se ale model nachází v tomto stavu, vykonává pouze odměňovanou akci[3], tj. $P_{A1} = 1$. Díky tomu se ale stav A sám o sobě dá považovat za cílový a ve všech nastaveních modelu popsaných v sekci 2.6 je nazýván stavem „je odměňován“. Za cíl tréninku se tak dá považovat odměňovaný stav, ne odměňovaná akce. K tomu se vztahuje i způsob výpočtu okamžité odměny *r*, který je navržen tak, aby model mohl být odměňován i pouhým zvukovým stimulem klasicky asociovaným se stimulem vedoucím do odměňovaného stavu, viz sekce 2.5.1.

[3] To koresponduje s tím, že živé zvíře při krátkém tréninku žere odměnu kdykoli k tomu má příležitost.



Obr. 4: Diagram znázorňující vztah stavů, stimulů a akcí modelu. Stimuly od uživatele zprostředkovávají transfer mezi stavy (plné šipky; stimul 1 je asociován se stavem A, stimul 2 se stavem B a stimul 3 se stavem C). Ve stavech jsou zobrazeny akce, které v nich lze vykonat s nenulovou pravděpodobností. Vykonání akce ve stavu ale tento stav nezmění. Model se sám od sebe vrací do základního stavu po uplynutí 5 sekund od posledního stimulu (přerušované šipky). Jediná akce, za kterou model přímo dostává nenulovou odměnu je a_1 (červeně zvýrazněna).

Formalizace modelu ukazuje, že jeho důležitou součástí je také množina konstant *C*. Tyto konstanty zásadně ovlivňují způsob i rychlost učení modelu až do té míry, že model může v závislosti na jejich nastavení simulovat různé druhy zvířat s různými vzorci chování. Význam a případné hodnoty jednotlivých konstant jsou podrobně rozebírány níže.

2.4 Učení modelu

Učení modelu probíhá zpětnou vazbou okamžité odměny *r*, což spočívá v úpravách matice *P*. Ve své podstatě se jedná o formu zpětnovazebního učení podobnou známému q-učení [22]. Klasické q-učení, jak je popsané v [22], se ale učí deterministicky a hlavně očekává statické prostředí, kdy se podmínky pro získání odměny nemění a v obdržených odměnách neexistuje šum. Proto jsem čerpal i z návrhu modelu představeného v [13]. Tento model se učí podobným mechanizmem jako q-učení, ale výběr další akce je u něj stochastický. Zde podrobně popíši adaptaci pravděpodobností *P* a zmíním zásadní rozdíly s modelem popsaným v [13].

Model si pro každou dvojici akce-stav uchovává kromě pravděpodobnosti vykonání dané akce v daném stavu i hodnotu, vyjadřující očekávanou odměnu, pokud model zvolí danou akci v daném stavu (tj. tyto hodnoty tvoří matici stejného typu jako *P*). Očekávaná odměna zde přesně koresponduje s očekávanou odměnou, jak je definovaná v klasickém q-učení [22] i v [13] a dále ji budu značit $R_{exp}(i,j,t)$ - pro očekávanou odměnu za akci a_j ve stavu s_i v čase t. Hodnota očekávané odměny všech dvojic akce-stav je po celou dobu činnosti modelu udržována v intervalu [0,1].

V klasickém q-učení by si model další akci vybíral podle největší očekávané odměny v daném stavu. To je ale založeno na předpokladu statického prostředí, kdy je odměňovaná akce neměnná a určitá posloupnost akcí vede vždy do stavu, kde je tato akce možná.

V případě Chovatele zvířat se ale uživatel může rozhodnout dříve odměňovanou akci nadále neodměňovat. Toto je dokonce nutný postup u techniky „tvarování“ (viz 1.2.2), hojně používané u cviků v programu. Proto je nutné, aby model kladl při učení větší důraz na zkoušení nových možností jak získat odměnu. Dokonale naučený model stále opakující jen několik málo osvědčených akcí, by totiž nebyl dobrou simulací chování živého zvířete.

První mechanismus, který zajišťuje větší důraz na zkoušení více různých akcí v daném stavu, je výběr založený na pravděpodobnosti. Díky tomu model v daném stavu i po opakovaném odměnění jedné akce může náhle provést jinou akci, která nikdy předtím odměněná nebyla. Stochastický výběr ale sám o sobě nestačí na případy zdánlivě nesmyslných akcí, které mají přirozeně na začátku učení prakticky nulovou pravděpodobnost vykonání[4].

Proto existují v modelu i nastavitelné podobnosti akcí nebo stavů. Podobné akce navzájem ovlivňují své pravděpodobnosti nejen v aktuálním stavu, ale i v podobných stavech. Tak lze postupným učením a zvedáním pravděpodobností podobných akcí docílit vykonání dříve nemyslitelné akce. Příkladem může být trénink psa popsaný v sekci 1.3, kde je nejprve pravděpodobnost, že pes podá pac do nastavené ruky téměř nulová. Hráč ale postupným odměňováním dotyku ruky s packou psa (podobný stav) a mírného pohybu packy psa (podobná akce) docílí toho, že se pravděpodobnost podání pacu zvedne dostatečně na to, aby to pes skutečně udělal. Konkrétní nárůst pravděpodobnosti v podobných případech řeší speciální experiment v kapitole 4. Nastavení podobností akcí a stavů spadá do množiny konstant C, která je zmíněna ve formalizaci modelu výše.

[4] Jedná se o akce, které by zvíře za normálních okolností nikdy neudělalo, jako např. to, že lemur zaleze do přepravky a sám se v ní zavře. Přesto jsou tyto akce možné a pravděpodobnost těchto akcí je na začátku učení nenulová (je nastavena na 0.001).

Proces úpravy pravděpodobností P přibližuje pseudokód na obrázku 5. Tento pseudokód je spuštěn po každém výběru nové akce (tj. po vykonání akce nebo při přerušení akce) a upravují se při něm jak hodnoty P, tak hodnoty očekávané odměny právě vykonané dvojice akce-stav.

StatesStack je zásobník stavů, ve kterých se model postupně nacházel. Podobně *ActionsStack* je zásobník, do kterého se ukládají vykonané akce. Proměnná s_i tak reprezentuje stav, ve kterém byla vykonána poslední akce modelu, s_k reprezentuje aktuální stav modelu a a_j značí poslední vykonanou akci.

Funkce r na řádku 5 určuje obdrženou odměnu r_0 za vykonání akce a_j ve stavu s_i. Její fungování ovlivňuje nastavení klasického podmiňování v modelu a je detailně popsáno v sekci 2.5.1. Zde postačí, že odměna r_0 je vždy z intervalu [0,1].

Odměna r_0 se pak propaguje do několika posledních dvojic akce-stav (for-cyklus na obrázku 5). Dále budu popisovat pouze jednu iteraci propagace odměny a budu jí brát jako jedinou, která proběhla v diskrétním okamžiku t. Případné další iterace, které používají sníženou hodnotu r_0 (řádek 18 na obrázku 5), probíhají totožně a za výchozí hodnoty všech vstupních veličin používají hodnoty upravené předchozí iterací.

```
01    s_k := StatesStack.pop();              // aktuální stav modelu
02    s_i := StatesStack.pop();              // stav, ve kterém byla vybrána nová akce
03    a_j := ActionsStack.pop();             // poslední vykonaná akce
04
05    r_0 := r(s_i, a_j);          // odměna obdržená během vykonávání akce a_j (ze stavu s_i)
06
07    for ( int it = 0; it < c_iter; it++ ) {
08     R := Vypočti rozdíl obdržené odměny od odhadu odměny, za vykonání akce a_j
       ve stavu s_i (r_0, a_j, s_i, s_k, R_exp);
09
10     Uprav očekávanou odměnu akce a_j ve stavu s_i (R, a_j, s_i);
11
12     Uprav pravděpodobnosti vykonání všech akcí (R, a_j, s_i, A, S, P);
13     Kontrola zachování smyslu u nových hodnot pravděpodobností P;
14
15      s_k := s_i;
16      s_i := StatesStack.pop();
17      a_j := ActionsStack.pop();
18      r_0 := r_0 * c_rew;                  // snížení propagované odměny
19    }
```

Obr. 5: Pseudokód popisující průběh úpravy pravděpodobnosti a očekávané odměny za vykonání akce ve stavu. Zobrazovaný kód se spouští před každým výběrem nové akce (tj. po vykonání akce nebo při přerušení akce).

Úprava pravděpodobností P v čase t po vykonání akce a_j ve stavu s_i probíhá tak, že se nejprve spočte rozdíl odhadu odměny, kterou vykonání akce a_j přinese, od skutečného zisku, který její vykonání opravdu přineslo (řádek 8 na obrázku 5). Tento rozdíl dále nazývám $R(i,j,t)$ a platí:

$$R(i,j,t) = r_0 + \gamma \cdot F(s_k) - R_{exp}(i,j,t). \tag{1}$$

Zde $0 \leq r_0 \leq 1$ je konstanta značící okamžitou odměnu obdrženou za vykonání akce, konstanta $0 < \gamma < 1$ udává míru „důvěryhodnosti“ odhadu odměny, kterou model může obdržet v novém stavu s_k[5], a funkce F pro daný stav tento odhad provádí. Konkrétně funkce $F(s_k)$ vrací prostě aktuální očekávanou odměnu akce s největší pravděpodobností ve stavu s_k.

Z rovnice (1) je patrné, že $R(i,j,t)$ je rozdíl očekávané odměny za vykonání akce a_j ve stavu s_i (tj. $R_{exp}(i,j,t)$) a odhadu skutečně obdržené odměny, reprezentovaného součtem obdržené odměny r_0 s odhadem odměny, která lze dále získat v aktuálním stavu (tj. $\gamma \cdot F(s_k)$).

Pokud je tento rozdíl kladný, byla akce a_j ve stavu s_i nedoceňována a naopak, pokud je rozdíl záporný, byla přeceňována. Cílem učení tak je rozdíl pro všechny dvojice akce-stav minimalizovat. Rovnice (1) je používána v [13] a je běžná i v modelech učení časovými rozdíly (TD learning).

Po vypočtení rozdílu $R(i,j,t)$ se upravuje hodnota očekávané odměny $R_{exp}(i,j)$ a pravděpodobnosti v matici P (řádek 10 na obrázku 5). Aktualizace očekávané odměny je totožná s aktualizací očekávané odměny v klasickém q-učení podle [22] i v modelu [13] a řídí se vztahem

$$R_{exp}(i,j,t+1) = R_{exp}(i,j,t) + \alpha_r \cdot R(i,j,t). \tag{2}$$

Konstanta $0 < \alpha_r < 1$ zde udává míru změn očekávané odměny. Na začátku učení jsou hodnoty očekávané odměny pro všechny stavy a akce nastaveny na nulu. Z rovnice (2) ale vyplývá, že hodnota očekávané odměny pro některé dvojice akce-stav může vyjít záporná (přestože skutečná odměna r_0 je z intervalu [0,1]). V takovém případě je daná očekávaná odměna nastavena na velice malou kladnou hodnotu R_{exp_min}[6]. V opačném případě, kdy hodnota očekávané odměny přeroste 1, je prostě nastavena zpět na 1.

[5] Zpravidla totiž neplatí rovnost $k = i$, tj. stav modelu se za dobu konání akce změní. Nejčastěji je to způsobeno stimulem od uživatele, ale může to být zapříčiněno i návratem modelu do základního stavu po uplynutí 5 s od posledního stimulu.

[6] V programu Chovatel zvířat je tato hodnota 0.001.

Pravděpodobnosti vykonání akce ve stavu se upravují rozdílně od pravděpodobností v [13]. Je to dáno tím, že v našem modelu existují podobnosti stavů a akcí (zmíněné výše). Díky tomu se po vykonání libovolné akce v libovolném stavu mění nejen pravděpodobnost této dané akce v daném stavu, ale i pravděpodobnosti všech podobných akcí ve všech podobných stavech (řádek 12 na obrázku 5). Změny pravděpodobnosti vykonání pro všechny akce a_y ve všech stavech s_x popisuje rovnice (3).

$$P_{xy}^{t+1} = P_{xy}^{t} + \alpha_p \cdot SimS_{xi} \cdot SimA_{yj} \cdot R(i,j,t) \tag{3}$$

Zde $0 < \alpha_p < 1$ značí míru změny pravděpodobnosti v jednom kroku algoritmu, $0 \leq SimS_{xi} \leq 1$ je konstanta udávající podobnost stavů s_x a s_i a $0 \leq SimA_{yj} \leq 1$ je konstanta udávající podobnost akcí a_y a a_j. *R(i,j,t)* je rozdíl očekávaného a skutečného zisku, vypočítaný pomocí (1) po vykonání akce a_j ve stavu s_i.

Počáteční hodnoty pravděpodobností *P* jsou nastaveny tak, aby se zvíře simulované modelem na začátku experimentu chovalo realisticky. Pokud je ale pravděpodobnost pro danou dvojici akce-stav na začátku rovna 0, znamená to, že daná akce v tomto stavu není možná (např. akce a_1 v základním stavu na obrázku 4) a tato pravděpodobnost se v průběhu učení nemůže prostřednictvím (3) zvýšit.

Vztah (3) se postupně aplikuje na všechny dvojice akce a_y a stavu s_x takové, že $SimA_{yj}$ i $SimS_{xi}$ jsou nenulové a akce a_y je možná ve stavu s_x. Po aktualizaci všech takových hodnot pravděpodobností se zajišťuje zachování jejich smyslu (tj. $\forall s_x \in S: \sum_{y=0}^{n} P_{xy} = 1$) způsobem popsaným níže (řádek 13 na obrázku 5).

První problém, který se musí u pravděpodobností řešit, spočívá v tom, že podobně jako u očekávané odměny z rovnice (3) vyplývá, že pravděpodobnost určité dvojice akce-stav by mohla během učení nabýt záporných hodnot (protože rozdíl *R(i,j,t)* může být záporný). V takovém případě je daná pravděpodobnost nastavena na velice malou kladnou hodnotu $Prob_{min}$[7].

Dalším problémem je zachování toho, aby pravděpodobnosti všech možných akcí v daném stavu dohromady dávaly 1. Pokud je po vykonání akce a_j ve stavu s_i rozdíl *R(i,j,t)* ze vztahu (1) kladný, znamená to, že se prostřednictvím (3) zvýšily pravděpodobnosti všech akcí podobných a_j ve všech stavech podobných stavu s_i. Protože se ale pravděpodobnost žádné akce nesnížila, znamená to, že ve všech těchto stavech je suma pravděpodobností všech

[7] U pravděpodobností v programu Chovatel zvířat je tato hodnota 0.01.

možných akcí větší než 1. Aby se zachoval smysl hodnot pravděpodobností, přeškálují se v takovém případě pravděpodobnosti všech akcí v relevantních stavech tak, aby jejich celkový součet byl opět 1. Ve výsledku to znamená, že se nepatrně sníží pravděpodobnosti akcí, které nebyly podobné s aktuálně provedenou akcí.

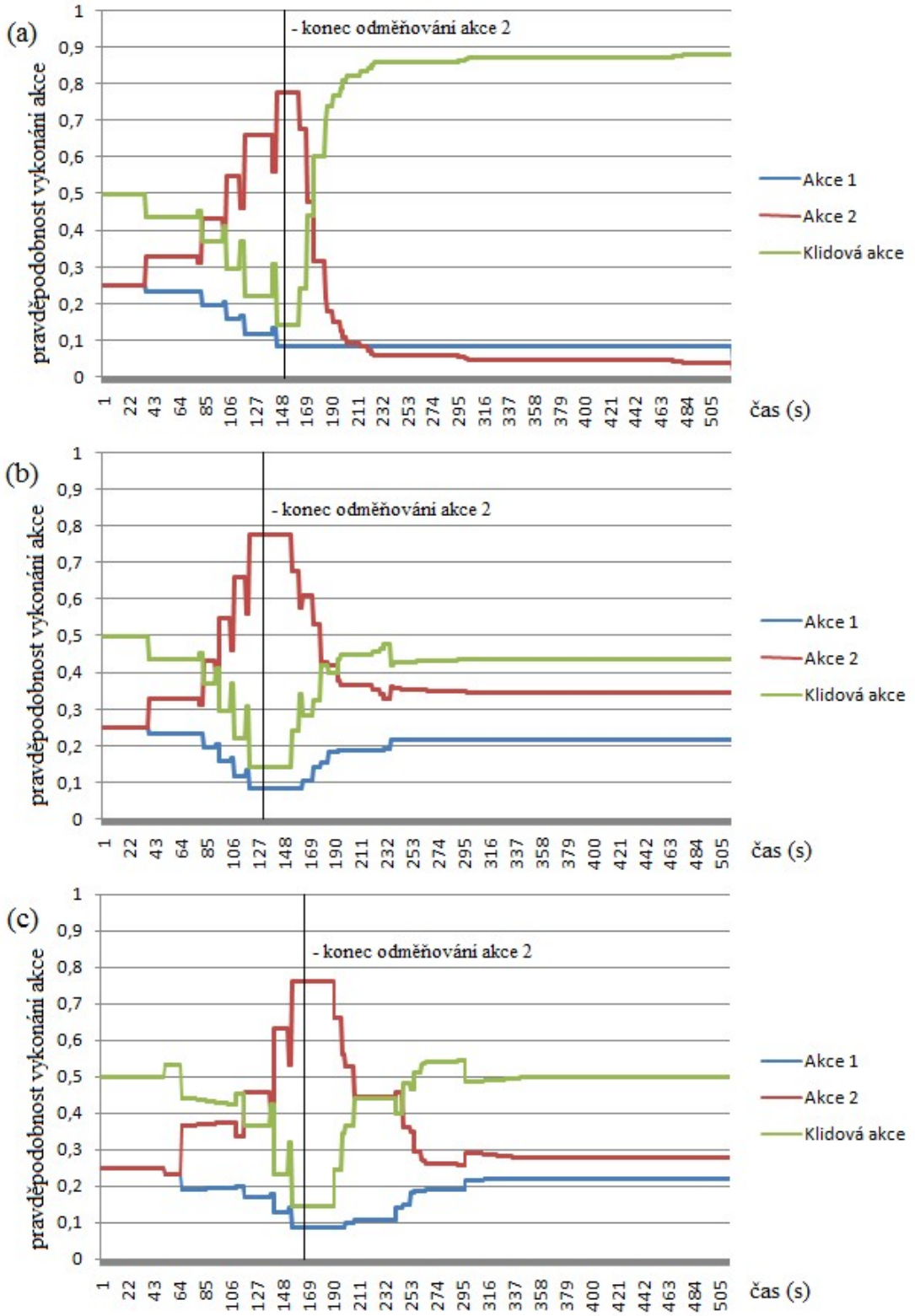
Naopak to ale takto fungovat nemůže. Pokud se totiž aktuální pravděpodobnosti vykonání podobných akcí v podobných stavech sníží, nemohou se všechny přeškálovat, protože by mohla bez zjevného důvodu narůst pravděpodobnost nikdy nepoužitým a velice odlišným akcím (což je navíc zpravidla cílem celého tréninku). Proto by se v tomto případě měla zvýšit pouze pravděpodobnost jedné klidové akce (např. akce „nedělej nic“), aby suma pravděpodobností zůstala rovna 1.

Tento přístup ale dlouhodobě není použitelný. Na obrázku 6 je znázorněn průběh pravděpodobnosti dvou navzájem nepodobných akcí v případě, že jednu z nich uživatel nejprve často odměňuje, a ve chvíli, kdy pravděpodobnost jejího vykonání v daném stavu přesáhne 0.75, odměňovat přestane. V zobrazovaném stavu je kromě těchto dvou akcí ještě možné vykonat klidovou akci.

Obrázek 6a ukazuje průběh pravděpodobností v případě, že se jako kompenzace ke snížení pravděpodobností pouze zvyšuje pravděpodobnost klidové akce. Je patrné, že odměňovaná akce nejprve potlačila pravděpodobnost ostatních akcí, ale když odměňování přestalo, postupně tato akce naprosto vymizí z chování zvířete. To je zapříčiněno tím, že tato akce má v daném stavu nenulovou (i když velmi nízkou) očekávanou odměnu, která ovšem podle (1) a (3) zapříčiňuje to, že rozdíl očekávaného a skutečného zisku je po vykonání této akce vždy záporný. Tak se pravděpodobnosti všech akcí kromě klidové již nadále pouze snižují (v případě vykonání akce 2) nebo nemění (v případě vykonání nikdy neodměněné akce 1 s nulovou očekávanou odměnou, která ani není podobná akci 2). To by ale znamenalo, že zvíře časem v daném stavu bude vykonávat pouze klidovou akci.

To samozřejmě není žádoucí, a proto pokud v daném stavu klesne suma pravděpodobností všech akcí kromě klidové akce pod mez danou konstantou $prob_{lim}$, přeškálují se ty z nich, které mají pravděpodobnost vykonání větší než 0.05 (tj. dostatečně obvyklé akce), aby jejich celková suma pravděpodobností byla rovna $prob_{lim} + 0.1$. Pravděpodobnost zbývající do sumy rovné 1 (tj. $1 - prob_{lim} - 0.1$ – pravděpodobnosti neobvyklých akcí) se přidává klidové akci.

Na obrázku 3b a 3c je vidět vliv tohoto pravidla na ustálení pravděpodobností akcí 1 i 2. Dříve odměňovaná akce se nyní podle očekávání ustálí na pravděpodobnosti o něco větší, než s jakou začínala. Ale i pravděpodobnost akcí, které byly odměňováním potlačeny, se



Obr. 6: Průběh pravděpodobností dvou navzájem nepodobných akcí. Ve všech případech byla akce 2 v daném stavu odměňována, až dosáhla na úkor ostatních akcí pravděpodobnosti vykonání přes 75%. Potom model přestal být odměňován úplně. a) – další vývoj pravděpodobnosti v případě, že jako kompenzace snížení pravděpodobnosti akce 2 roste jen pravděpodobnost klidové akce. b) – pravděpodobnosti v případě, že jsou škálovány, klesne-li suma pravděpodobností akcí kromě klidové pod $prob_{lim} = 0.8$. c) – pravděpodobnosti v případě, že jsou škálovány, klesne-li suma pravděpodobností akcí kromě klidové pod $prob_{lim} = 0.7$.

částečně zvedne (na to má největší vliv hodnota konstanty $prob_{lim}$, viz obrázek 6b a 6c). Navíc díky tomu, že jsou po ustálení pravděpodobnosti akcí škálováním udržovány na stejné úrovni, pravděpodobnost dříve odměňované akce již dále neklesá, přestože očekávaná odměna této akce R_{exp} je nenulová a odměna po jejím vykonání již vůbec nepřichází.

Nastavením hodnoty $prob_{lim}$ se dále zabývá i experiment zaměřený na trénink „zachycením“ v kapitole 4.

Na obrázku 6 si dále lze povšimnout poklesu pravděpodobnosti akce 2 v době, kdy byla odměňována – vždy těsně před jejím vzrůstem. Tento pokles je zapříčiněn tím, že model očekává odměnu ve formě r_0 okamžitě po ukončení akce 2. Ta ale nepřichází. Odměna se totiž generuje až se stimulem od uživatele, který model odkáže do stavu, kde může vykonat opravdu odměňovanou akci „Žere“, viz obrázek 4 a sekce 2.5.1. To znamená, že nenulová odměna se propaguje až po vykonání odměňované akce, viz postup na obrázku 5.

Nevýhodou vztahu (3), ale i (1) a (2) by se mohlo zdát velké množství konstant, které ovlivňují výpočet nové pravděpodobnosti jednotlivých akcí. Kromě konstant α_p, α_r, a γ totiž průběh učení ovlivňuje také nastavení podobnosti akcí a stavů i hodnota odměny r_0. Tyto podobnosti a hodnota odměny by se ale spíše než za parametry modelu (resp. simulovaného zvířete) daly považovat za konstanty vázané k prostředí a prováděnému cviku. Proto jsem při experimentech s modelem vždy měnil jen parametry α_p, α_r, a γ. Hodnoty podobností stavů a akcí byly ve všech případech nastaveny po konzultaci s expertem na trénování zvířat.

V programu tyto tři konstanty uživatel může nepřímo měnit volbou „obtížnosti“ cviku v podobě jinak barevného zvířete. Hodnoty podobností akcí a stavů i velikost odměny jsou pro každý cvik konstantně nastavené na základě konzultace s expertem.

2.5 Další vlastnosti modelu

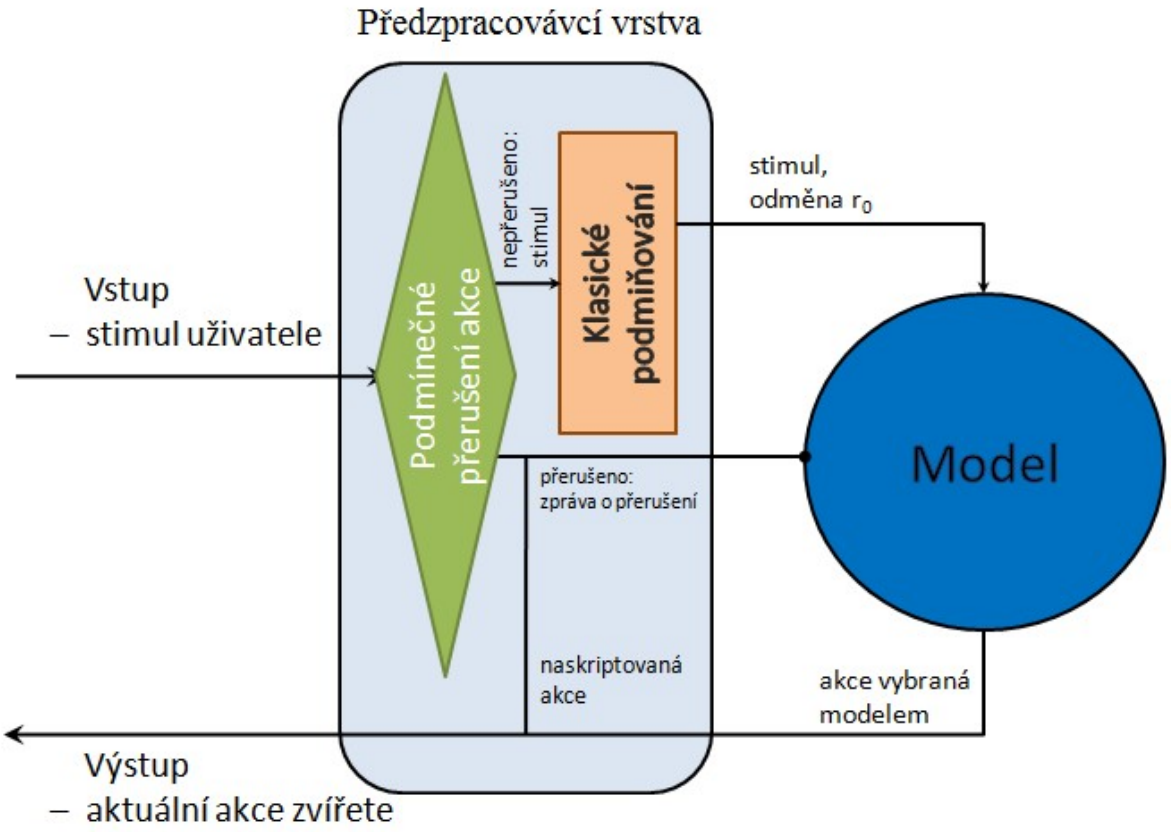
Model se dokáže učit na základě zpětné vazby z okolí, jak je popsáno výše. Podle požadavků na něj kladených by se ale měl model umět učit i pomocí klasického podmiňování (viz 1.2.4 a [18]). Navíc některé tréninky obsažené v programu Chovatel zvířat potřebují k zdárnému a pro hráče poučnému splnění možnost nechat model přerušit vykonávanou akci bez zásahu uživatele nebo učit model vykonat sekvenci akcí.

Klasické podmiňování i arbitrární přerušování akcí je v modelu řešeno vrstvami filtrujícími vstup od uživatele ještě před samotným modelem, který je popsán výše. Diagram na obrázku 7 tak zobrazuje proud informací od uživatele k modelu a zpět. Detaily o fungování

vrstev zpracovávajících vstup jsou popsány v sekcích 2.5.1 (klasické podmiňování) a 2.5.2 (přerušení akce).

Vedle funkce obou vrstev, zpracovávajících vstup od uživatele, je důležitým faktorem učení sekvencí akcí. To je bohužel vzhledem k architektuře modelu prakticky nemožné. Model totiž sám od sebe (bez stimulu od uživatele) nemůže změnit svůj aktuální stav. To znamená, že naučená sekvence akcí, by musela proběhnout beze změny stavu. V takovém případě ale vzhledem k stochastickému výběru akcí neexistuje mechanismus, který by zajišťoval správné pořadí akcí v sekvenci.

Na druhou stranu se v případě sekvence akcí, která potřebuje ve svém průběhu stimuly od uživatele, aby se měnil aktuální stav, nejedná o řetězení akcí ve smyslu zpětnovazebního učení. I když model lze takto naučit, taková sekvence není celek a uživatel by mohl začít sekvenci z libovolného bodu.



Obr. 7: Diagram přenosu dat v modelu. Stimul od uživatele je před vstupem do samotného modelu, popsaného v 2.3 a 2.4, zpracován ještě dvěma dalšími vrstvami. První z nich zajišťuje přerušení akce, kdy model v reakci na předdefinované podmínky může přerušit aktuální akci jinou akcí. V tom případě dostane model pouze informaci, jaká akce bude nově spuštěna, ale nic už dál neovlivňuje. V případě, že k přerušení akce nedojde, předá se nezměněný vstup vrstvě, která zajišťuje klasické podmiňování. Tato vrstva z daného stimulu vypočítá hodnotu r_0 ve smyslu rovnice (1) a předá oboje modelu, který pak vybere novou akci k vykonání. Dotaz na novou akci zvířete přichází v návaznosti na stimul uživatele, ale také po každém přirozeném ukončení akce. Hodnota stimulu tedy nemusí být vůbec nastavena.

Přes tyto problémy je v programu kvůli tréninku s lemurem potřeba, aby model alespoň nějak podporoval učení řetězení akcí. Proto existují akce, které svým vykonáním mění aktuální nebo dokonce základní stav modelu. Detailnější popis těchto akcí a důsledků jejich použití viz popis nastavení modelu při učení s lemurem v sekci 2.6.2.

2.5.1 Klasické podmiňování modelu

Cviky se zvířaty od modelu vyžadují schopnost klasického podmínění jen do jisté míry. Vlastně je jen potřeba, aby model dokázal spárovat (klasicky podmínit, viz [18]) jeden zvolený stimul se stimulem odměny (tj. stimulem vedoucím do odměňovaného stavu). Dovolit párování dvou libovolných stimulů by bylo nejen složitější pro model, ale hlavně by značně ztížilo užívání programu a jeho poučnost pro uživatele.

Z toho důvodu je do modelu přidán mechanismus, který před vlastním modelem zpracovává aktuální stimul od uživatele a určuje hodnotu právě obdržené odměny. Jedná se vlastně o výpočet funkce r ze specifikace modelu v sekci 2.3, která vrací r_0 ze vztahu (1). Tuto hodnotu vrstva klasického podmínění poté společně s aktuálním stimulem předává jako vstup modelu, který tak do jejího výpočtu nezasahuje, viz obrázek 7.

Klasické podmiňování je simulováno tak, že se uchovává hodnota provázání specifikovaného stimulu se stimulem odměny (tj. stimulem vedoucím do odměňovaného stavu). Tuto hodnotu budu dále nazývat r_k a vždy platí, že $0 \leq r_k \leq 1$.

Před popisem výpočtu r_k je ale nutné vysvětlit, v jakých případech se modelu vůbec předává nenulová odměna r_0. Existují dvě možnosti, kdy se tak děje – byl použit buď stimul odměny, nebo párovaný stimul.

Pokud byl použit párovaný stimul, předá se modelu jako r_0 aktuální hodnota r_k. Také se zanese použití párovaného stimulu do krátkodobé paměti modelu, v které si model použití párovaného stimulu určitou dobu udržuje[8].

Pokud byl použit stimul odměny a v krátkodobé paměti modelu není předchozí použití párovaného stimulu, pak se rovnou modelu předává hodnota $r_0 = 0{,}5$. Tato hodnota je záměrně relativně nízká, aby se vyplatilo model odměňovat za pomoci párovaného stimulu.

U reálných zvířat tato metoda sice normálně není významně efektivnější, ale některé typy tréninků se bez jejího použití prakticky neobejdou. Z výukových důvodů se tak model podle požadavků ze sekce 2.1 musí učit efektivněji, pokud je odměna asociována se zvukovým stimulem, čehož se dociluje právě předáním nízké hodnoty r_0.

[8] Krátkodobá paměť modelu byla po konzultaci s expertem nastavena na 20 sekund.

Pokud ale v krátkodobé paměti modelu použití párovaného stimulu je, nepředá se odměna žádná, protože to znamená, že model již byl odměněn.

V obou možných případech se také přepočítává hodnota r_k. Předcházel-li párovaný stimul stimulu odměny, je hodnota r_k zvýšena a naopak, pokud stimul odměny nenásleduje za párovaným stimulem nebo je zaznamenán stimul odměny bez párovaného stimulu v krátkodobé paměti, hodnota r_k se zmenšuje.

Jak konkrétně se hodnota r_k změní, záleží na době, která uplyne mezi párovaným stimulem a stimulem odměny. K vypočtení změny r_k se používá míra párování $M_p(t)$ dvou za sebou jdoucích stimulů, která vychází ze vztahu (4).

$$\mathrm{M_p}(t) = \begin{cases} t \cdot 0.15 + 0.7, & t \leq [0,2] \\ \min\left(1, -\frac{(t-2)^{1.5}}{8} + 1.5\right), & t \in (2,6) \\ 0.5 - (t-6) \cdot \frac{1}{28}, & t \in [6,20] \end{cases} \tag{4}$$

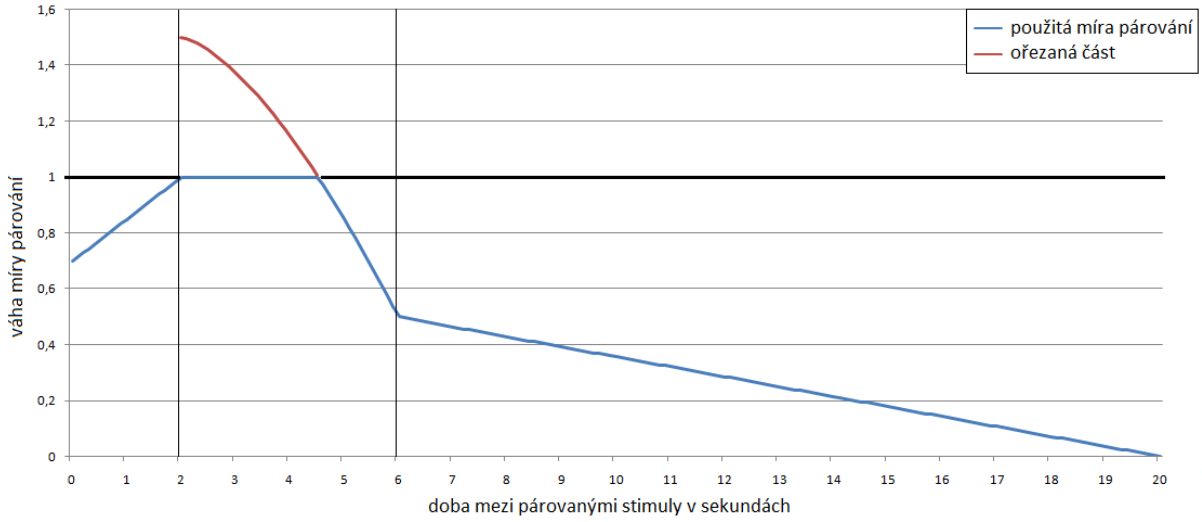
Zde t je doba, která uplynula mezi použitím párovaného stimulu a stimulu odměny v sekundách. Pokud je $t > 20$, znamená to, že párovaný stimul vůbec nebyl v krátkodobé paměti a $M_p(t)$ se nastavuje na nulu. V tom případě se bude hodnota r_k snižovat.

Přes relativní složitost (4) je hodnota míry párování spojitá a v intervalu [0,1]. Pro lepší představu je její průběh zobrazen také na obrázku 8. Model je samozřejmě parametrizovatelný, takže délka krátkodobé paměti, ani časy přechodů (tj. po řadě 20, 2 a 6 sekund) nejsou pevně nastavené. Model po nastavení těchto časů automaticky vygeneruje rovnice (4) tak, aby pro nové nastavení byla opět $M_p(t)$ spojitá po celé délce krátkodobé paměti.

Zde uvádím všechny parametry rovnic (4) jako konstanty, protože uživatel jejich nastavení nemůže změnit a všechna zvířata v programu používají stejné časové a tedy i ostatní hodnoty. Důležitý je zde tvar křivek v jednotlivých mezidobích, který byl konzultován s expertem a vychází i ze základů pavlovovského učení popsaných v [18].

Po spočtení míry párování v daném čase se její hodnota využije k přepočítání r_k. To se zvětšuje lineárně v závislosti na její nenulové velikosti, ale snižuje se, pokud je $M_p(t)$ nulová. Závislost mezi r_k a mírou párování popisuje vztah (5).

$$r_k(t+1) = \begin{cases} r_k(t) + \beta_p \cdot M_p(t), & M_p(t) > 0 \\ \beta_n \cdot r_k(t), & M_p(t) = 0 \end{cases} \tag{5}$$



Obr. 8: Graf hodnoty míry párování M_p v závislosti na době, která uběhla mezi párovaným stimulem a stimulem odměny. Míra párování je shora omezena 1 a zespodu 0. Na intervalu [0,2] míra roste lineárně, další zhruba 2-3 sekundy dosahuje maximální možné hodnoty (tj. jde o ideální dobu mezi párovanými stimuly). Pak ale míra párování počne polynomiálně klesá až do doby 6 sekund mezi stimuly, kdy začne klesat lineárně a pozvolna až k 0. To, že na intervalu [2,6] míra párování klesá polynomiálně a ne lineárně, by se více projevilo při jiném nastavení časových konstant paměti

Zde $\beta_p = 0.6$ a $\beta_n = 0.8$ jsou opět konstanty nastavené po konzultaci s expertem na podmiňování zvířat, které jsou neměnné pro všechna zvířata v programu.

Z (5) vyplývá, že již druhé použití párovaného stimulu může pro model znamenat větší odměnu, než jaké je možné dosáhnout pouze se stimulem odměny. To je samozřejmě odlišné od živých zvířat, ale splňuje to požadavek efektivity při krátkém tréninku ve hře.

V zájmu dalšího zjednodušení párování dvou stimulů se navíc hodnota r_k snižuje podle vztahu (5) až v případě, že je $M_p(t)$ nulová dvakrát za sebou. Tím se dociluje toho, že stejně jako u živých zvířat, stimul spárovaný s odměnou nemusí při tréninku opravdová odměna vždy nutně následovat.

2.5.2 Přerušení akce

Podobně jako klasické podmiňování, i schopnost přerušení právě probíhající akce není vyžadována v mnoha případech a je tak řešena pouze v rámci předzpracování vstupu modelu, viz obrázek 7.

Při trénincích model potřebuje sám od sebe přerušit akci pouze, aby se ukázalo, k čemu vede neobratné nebo nevčasné zacházení se stimuly. Například při tréninku se psem, který je popsán v sekci 1.3, musí uživatel v určitém kroku učení přerušit akci zvedání packy, aby jí pes zastavil ve vzduchu a nesnažil se dotknout vysoko natažené ruky. Pokud to neudělá,

je model psa nastaven tak, že automaticky spustí akci „vyskakuje“, kdy se pes začne skákáním snažit dotknout ruky nad sebou. Spuštění akce „vyskakuje“ v tomto případě demonstruje to, že pes je z předchozího tréninku naučen dotýkat se ruky uživatele, viz 1.3.

Arbitrární přerušení akce se ale děje jen za předem určených podmínek. Přesněji, v konfiguračním souboru modelu lze nastavit, jaká akce, pokud byla vykonána v určitém stavu, bude přerušena jakou akcí. Pokud se pak při tréninku daná akce v daném stavu skutečně vykoná, model po uběhnutí 60% její celkové délky spustí předem naskriptovanou akci. V takovém případě už se modelu předá pouze zpráva o tom, jaká akce aktuálně probíhá, ale jinak už nic do výběru akce nezasahuje, viz obrázek 7.

Je důležité si přitom uvědomit, že každý stimul od uživatele přerušuje aktuální akci. Takové přerušení se ale děje změnou vnitřního stavu modelu a standardním stochastickým výběrem nové akce.

Uživatel tak může stimulem spuštění naskriptované akce přerušit dřív, než podmíněná akce dojde do 60% své délky (akce zpravidla trvají 4-6 sekund, takže na to má času dost). Proto také ve zmíněném příkladu ze sekce 1.3 uživatel odměněním psa zabrání spuštění akce „vyskakovat“.

Funkce samovolného přerušení akce se kromě tréninku se psem používá ještě při tréninku s lemurem, více viz níže.

2.6 Schémata modelů pro různá zvířata

Vzhledem k velkému rozsahu učících technik, které lze modelem simulovat, popíši v této sekci nastavení stavů, akcí a stimulů zvlášť u jednotlivých zvířat v programu. Kromě nastavení konstant podobností ve smyslu vztahu (3) se u některých zvířat používá také naskriptované přerušení akce nebo speciální podpora řetězení akcí.

Všechny modely mají ale několik společných prvků zběžně zmíněných v sekci 2.3. Základní a počáteční stav všech modelů je nazván „Sedí“. Další společné stavy jsou „Je odměňován“ a „Slyší kliker“. Stav „Je odměňován“ je u všech zvířat jediný stav, kde lze provést odměňovanou akci, viz 2.3. Ve všech případech je dále tento odměňovaný stav možné klasicky spárovat (způsobem popsaným v 2.5.1) se stavem „Slyší kliker“[9]. Oba stavy také sdílí společné stimuly od uživatele, které do nich model uvádějí.

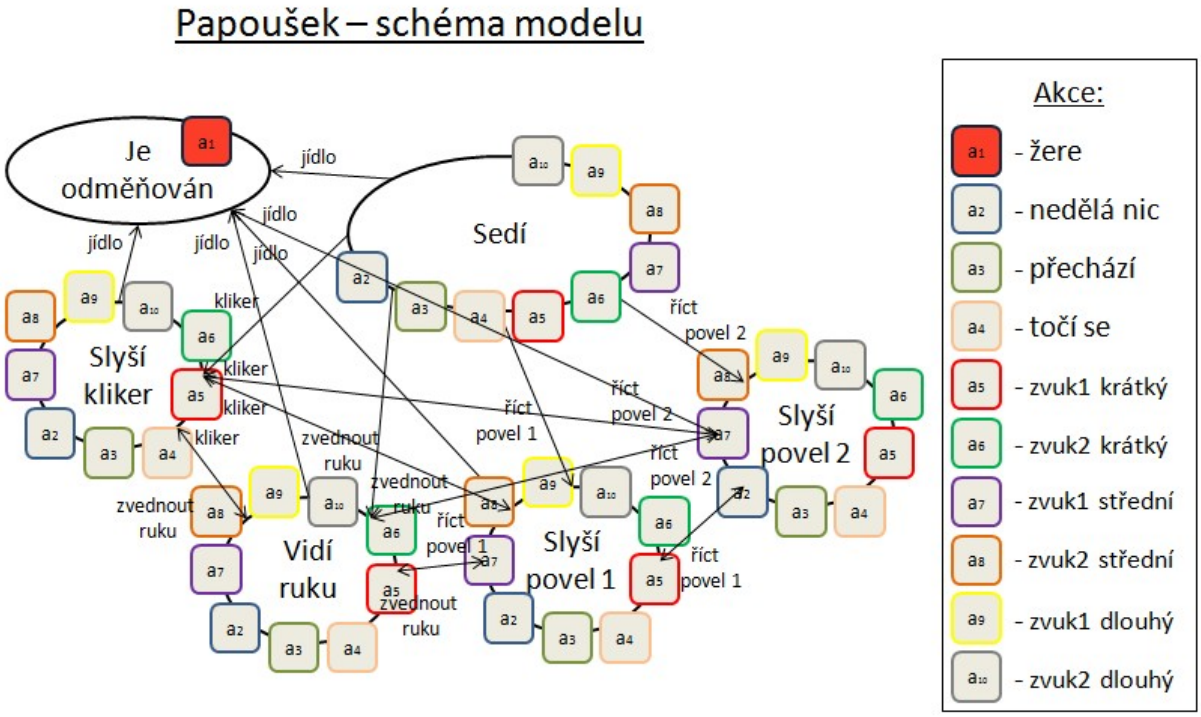
[9] Kliker je nástroj produkující krátký a výrazný zvuk, který používají trenéři zvířat jako stimul spárovaný s odměnou, viz [18].

Kromě odměňované akce „žere“ mají všechna zvířata pouze jednu společnou akci, „nedělá nic“. Tato akce se využívá jako klidová při úpravách pravděpodobností akcí popsaných v 2.4.

2.6.1 Jednoduchá schémata – papoušek a pes

Fungování modelu je navrženo tak, aby nejefektivnější způsob učení bylo učení „zachycením“ a „tvarováním“. Speciálně na „zachycení“ je zaměřen cvik s papouškem, kde jde jen o to, zvýšit pravděpodobnost vykonání akce, kterou model už od začátku tréninku sám od sebe občas vykonává, viz též 1.2.1 a 1.5.

Ve cviku popsaném v sekci 1.5 jde o zvýšení pravděpodobnosti vykonání akce a_9 ve stavu „Slyší povel 1“ a akce a_{10} ve stavu „Slyší povel 2“ vzhledem k popisu nastavení modelu na obrázku 9. Vzhledem k tomu, že model je již na začátku tréninku nastaven tak, že obě cílové akce v cílových stavech provede s pravděpodobností 0.1, jedná se o zdánlivě jednoduchou úlohu.



Obr. 9: Diagram nastavení stavů, akcí a stimulů pro model simulující papouška. Odměňovaná akce a_1 je zvýrazněna. Stav „Sedí“ je základní, tj. po uplynutí 5 sekund od posledního stimulu se do něj model automaticky navrací. V tomto nastavení je model jednoduchý – lze se vždy dostat do libovolného stavu (graf stavů a stimulů je úplný). Cílem tréninku je zvýšit pravděpodobnost akce a_9 ve stavu „Slyší povel 1“ a akce a_{10} ve stavu „Slyší povel 2“.

Ovšem díky tomu, že akce 5-8 jsou cílovým akcím podobné jak ve smyslu učení modelu, tak grafikou použitou v aplikaci, tato úloha není pro hráče nejjednodušší. Chybné odměnění jiné akce vede k drastickému zvýšení její pravděpodobnosti vykonání. Tím se nejen tato akce bude do budoucna opakovat častěji, ale protože jsou v takovém případě přeškálovány všechny pravděpodobnosti, aby jejich suma byla rovna jedné (viz 2.4), výsledný efekt je také ten, že se sníží pravděpodobnost cílové akce. Analýza v kapitole 3 rozebírá vliv podobných druhů chyb na délku tréninku.

Na procvičování techniky „tvarování“ je zaměřen cvik se psem. Postup při tomto cviku je podrobně popsán v sekci 1.3. Na obrázku 10 je zobrazeno nastavení stavů, akcí a stimulů modelu psa. Učení tvarováním využívá podobnost akce 5 a 6 a dále podobnost akce 6 a 7. Odměňování (a tím zvyšování pravděpodobnosti) akce 5 totiž podle (3) zvyšuje i pravděpodobnost vykonání akce 6 a obdobně je to i s akcemi 6 a 7. Navíc stavy, kdy je vidět ruka trenéra, jsou si také podobné a pravděpodobnosti se tedy mohou šířit i mezi nimi.

Trénink psa je ale o něco složitější než trénink papouška, a tak i jeho model využívá některé možnosti, které model papouška nepotřebuje. Například se využívá skript přerušení akce jinou akcí z 2.5.2, kde je dokonce model psa brán jako příklad užití přerušení.

Podobnost akcí 5 a 6 totiž předpokládá, že jejich cíl je dotyk s nastavenou rukou. Ale při přechodu od akce 6, kdy pes dává pac do natažené ruky, k akci 7, kdy pes zvedá pac do prázdna, je tento koncept porušen. Cíl akce 7, by tak ve stavu „Ruka vysoko“ měl být dotyk s nataženou rukou. Proto pokud uživatel akci 7, která končí v polovině vzdálenosti od ruky, v tomto stavu okamžitě neodmění, spustí model akci 8. Tato akce, při níž pes vyskakuje po natažené ruce, dokončuje předpokládaný pohyb, což ale není při tréninku žádoucí. Jde spíš o upozornění, že se uživatel při tréninku dopustil chyby. Akce 8 jako taková ovšem nelze samovolně vykonat v žádném stavu.

Podle popisu tréninku se psem v sekci 1.3 je dále třeba nahradit stimul „Ruka nahoře“ stimulem „Slyší povel“. Stavy, do kterých tyto stimuly vedou (tj. stav „Ruka vysoko“ a stav „Slyší povel“, viz obrázek 10), si nejsou podobné, takže se navzájem vůbec neovlivňují. Proto je do modelu přidán stav „Slyší povel + ruka“. Tento stav je přístupný pouze pokud uživatel použije oba stimuly ve správném pořadí[10] a je si podobný jak se stavem „Slyší povel“, tak se stavem „Ruka vysoko“, viz obrázek 10. Odměňování akce ve stavu „Ruka vysoko“ přímo ovlivňuje pravděpodobnosti ve stavu „Slyší povel + ruka“ a tento stav zase ovlivňuje pravděpodobnosti ve stavu „Slyší povel“.

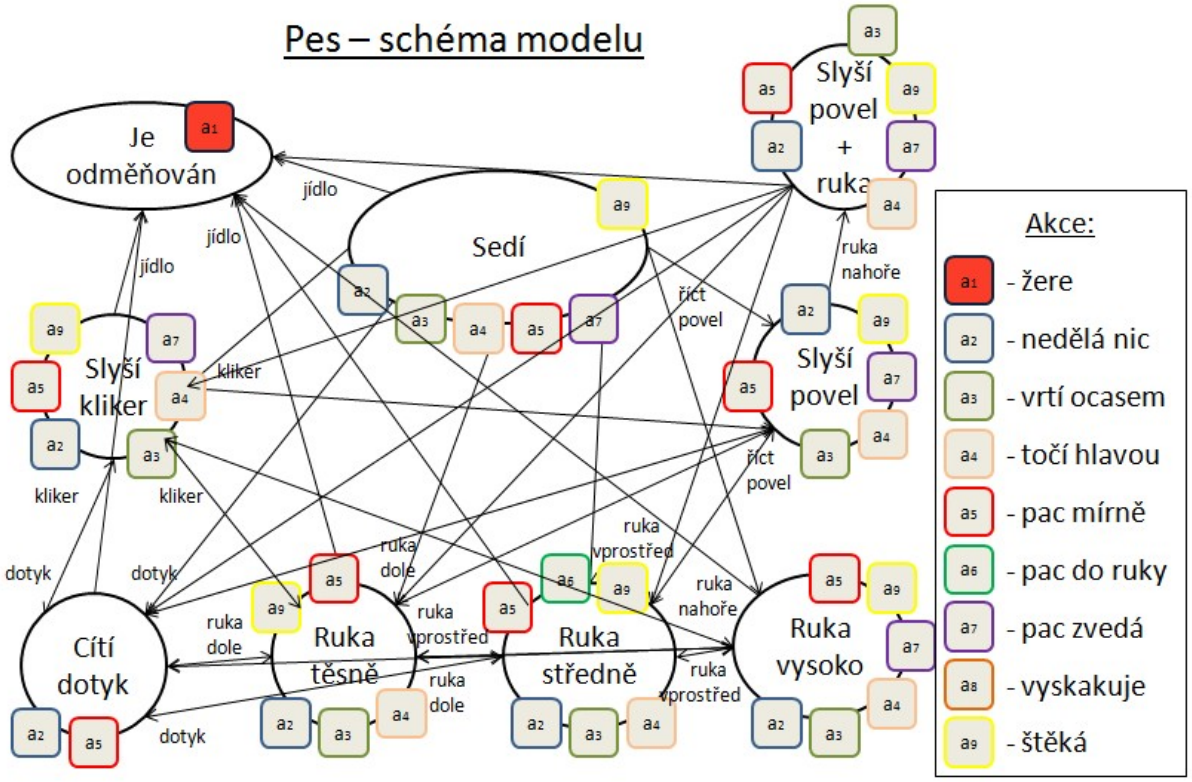
[10] Ve skutečnosti je to tak, že uživatel musí oba stimuly použít do 1 sekundy od sebe. Pak je model bere jako jeden speciální stimul, asociovaný se stavem „Slyší povel + ruka“.

To znamená, že jde o jakýsi přemosťující stav, který dovoluje převést naučené akce mezi dvěma navzájem si nepodobnými stavy. Tento postup lze při návrhu stavů, akcí a stimulů modelu použít obecně, takže lze například místo převedení akce zvedání packy, jak je popsáno v 1.3, převést jakoukoli akci, která je možná ve všech třech zmíněných stavech.

Přitom se zachovává reálnost prostředí, protože podobnosti stavů a akcí nejsou transitivní a takto propojené stavy tedy zůstávají navzájem nepodobné.

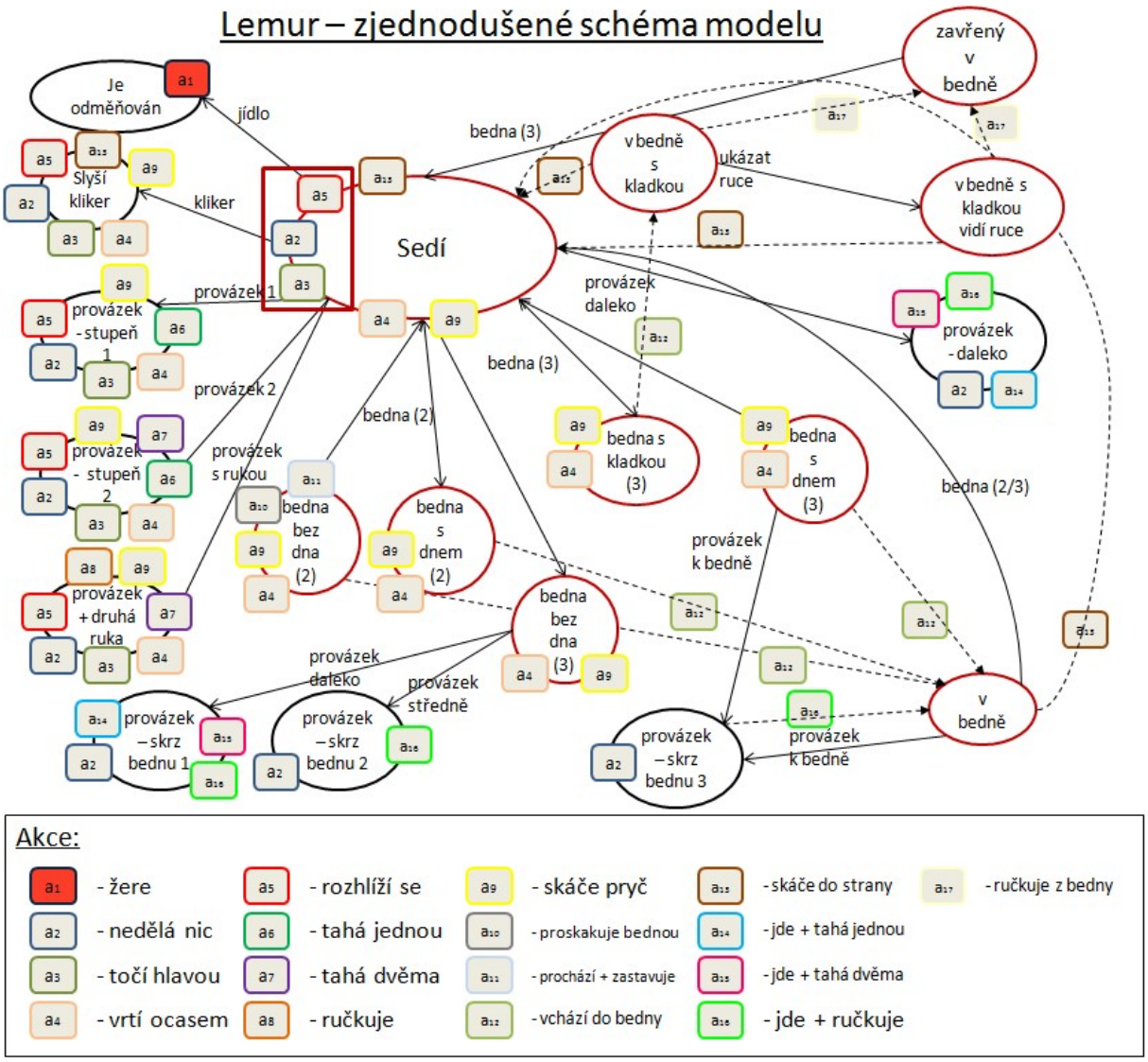
2.6.2 Komplexní model – lemur

Cvik s lemurem je zaměřen na učení řetězení akcí a kromě toho kombinuje jak „tvarování“, tak „zachycení“. Pro svoji komplexnost je na obrázku 11 zobrazen vztah stavů, akcí a stimulů modelu jen zjednodušeně[11].



Obr. 10: Diagram nastavení stavů, akcí a stimulů pro model simulující psa. Odměňovaná akce a_1 je zvýrazněna. Stav „Sedí“ je základní, tj. po uplynutí 5 sekund od posledního stimulu se do něj model automaticky navrací. Od nastavení, které simuluje papouška, se liší tím, že existuje speciální stimul, který uživatel může vytvořit pouze použitím stimulů „říct povel“ a „ruka nahoře“ za sebou. Na diagramu je to znázorněno tak, že graf stavů a stimulů není úplný – do stavu „Slyší povel + ruka“ se nedá dostat jinak, než ze stavu „Slyší povel“. Dále model psa obsahuje i speciální akci a_8, která nemůže být vykonána samovolně v žádném stavu a model jí vykonává pouze jako součást předdefinovaného skriptu.

[11] Akce a_2, a_3 a a_5 (na obrázku 11 jsou ve stavu „Sedí“ označeny červeným obdélníkem) jsou totiž proveditelné ve všech stavech kromě stavu „Je odměňován“ a stimuly klikeru a jídla jsou spustitelné ve všech stavech kromě „V bedně s klatkou“ a „V bedně s klatkou vidí ruce“.



Obr. 11: Zjednodušený diagram nastavení stavů, akcí a stimulů pro model simulující lemura. Odměňovaná akce a_1 je zvýrazněna. Model začíná v základním stavu „Sedí“, ale pokud se dostane do jakéhokoli červeně označeného stavu, stane se z něj nový základní stav. Model dále v tomto nastavení obsahuje akce „vchází do bedny“, „skáče do strany“ a „ručkuje z bedny“, které svým vykonáním mění stav modelu. To je rozlišeno použitím přerušovaných šipek v případě takového přesunu mezi stavy. Akce „nedělá nic“, „točí hlavou“ a „rozhlíží se“ (označené obdélníkem ve stavu „Sedí“) lze provést ve všech stavech kromě stavu „Je odměňován“. V zájmu přehlednosti dále nejsou zobrazeny všechny šipky znázorňující přechod pomocí stimulu „jídlo“ nebo „kliker“.

Trénink zaměřený na řetězení akcí je obecně pro model nejobtížněji zvládnutelný, viz důvody popsané v sekci 2.5. Aby se vůbec model mohl učit vykonat pevnou sekvenci akcí[12], musí být schopen měnit svůj aktuální stav bez závislosti na uživateli. To je zajištěno tak, že existuje kategorie předem specifikovaných akcí v určitých stavech, které svým vykonáním model přesunou do jiného stavu. Na obrázku 11 je takový přechod od přechodu zapříčiněného

[12] V případě modelu jde o zvýšení pravděpodobnosti určitých akcí ve stavech, které v pevné sekvenci následují po sobě. Jedná se tedy spíše o řetězení stavů než akcí.

stimulem odlišený tím, že je značen přerušovanou šipkou. Pokud se ale model takto přesune do jiného stavu, nový stav se navíc nastaví jako základní (tj. model se z něj už sám nedostane). Proč je toto nutné, jasně vyplývá z příkladu akce, která mění stav modelu. Tím může být akce „přechází do bedny“, při které lemur zaleze do připravené bedny. V bedně potom již zůstává, je to ale přirozeně jiná pozice s jinými možnostmi než mimo bednu, a proto i jiný stav.

Akci, která mění stav modelu, může uživatel odměňováním zvýšit pravděpodobnost vykonání v jednom stavu a následně dalším odměňováním zvýšit pravděpodobnost vykonání jiné akce ve stavu, kam předchozí akce model uvádí. Pokud mají obě akce dostatečně vysokou pravděpodobnost vykonání, model pak obě akce často vykonává za sebou. Takto jde navíc zapojit do sekvence více než dvě akce, pokud všechny akce kromě poslední zároveň mění stav modelu. V kapitole 3 je na příkladech analyzována schopnost modelu tvořit sekvence akcí.

Některé stimuly od uživatele také mohou měnit základní stav (např. zvednutí bedny v případě, že lemur se nachází v ní). Všechny stavy, které se mohou za tréninku stát základními (tj. stanou se základní, pokud se do nich model dostane), jsou na obrázku 11 zvýrazněny červeně.

V dalším fungování se již model simulující lemura od modelů papouška a psa neliší. Vzhledem k jeho obsáhlosti zde ale nebudu popisovat, kdy je v tréninku konkrétně potřeba uplatnit jakou techniku, jak jsem to výše udělal u papouška nebo psa.

Kapitola 3

Analýza chování modelu

V této kapitole analyzuji chování modelu vzhledem k požadavkům formulovaným v kapitole 2. Protože je model stochastický, v rámci každého bodu analýzy provádím minimálně 5 tréninků a vždy uvádím pouze průměrná data ze všech těchto pokusů.

Hlavními body analýzy jsou experimenty testující schopnost modelu učit se „zachycením“ i „tvarováním“ a schopnost řetězit akce. Dalšími pokusy jsem se zaměřil na odhadnutelnost doby nutné k ukončení tréninku, vliv chyb při tréninku na celkovou dobu tréninku a vliv klasického podmiňování (či spíše jeho absence) na rychlost tréninku. Analyzuji i vliv nastavení obtížnosti tréninku pomocí konstant α_p, α_r, a γ, viz sekce 2.4.

V závěru této kapitoly diskutuji výsledky provedených pokusů vzhledem k jednotlivým požadavkům na model.

3.1 Formát experimentů

Všechny experimenty v této kapitole používají stejné nastavení konstant klasického podmiňování, β_c = 0.6 a ω = 0.8. Konstanta ovlivňující škálování pravděpodobností akcí $prob_{lim}$ je nastavena na 0.7, pokud není řečeno jinak.

U experimentů se po stanovení přesného cíle, spočívajícího v hodnotě pravděpodobnosti vykonání dané akce či akcí v daných stavech, zaznamenává doba trvání a případné další vlastnosti, jako je počet chyb při tréninku a jejich druhy, procento stimulů odměny následující za zvukovým signálem, poměr použití samotného stimulu odměny apod. Každá sada experimentů zaměřených na jednu výše zmíněnou vlastnost obsahuje vždy minimálně 5 tréninků.

Průběh pravděpodobností akcí při experimentech se obecně nezaznamenává, protože jejich průměrování by nedávalo smysl (stochastické experimenty generují změny pravděpodobností v místech časové osy, která napříč experimenty nesouvisí).

V rámci každého tréninku, kde se k označení odměny využívá zvukový stimul, je tento stimul nejprve třeba spárovat s odměnou, což zabírá průměrně 10 sekund.

V některých případech je ovšem průběh jednoho konkrétního tréninku uveden jako názorný příklad obecné vlastnosti, která byla společná pro všechny tréninky odpovídající sady.

Výsledky experimentů jsou ihned diskutovány v jednotlivých sekcích a základní poznatky plynoucí z experimentů stejně jako obecné připomínky k experimentům jsou shrnuty v diskuzi v závěru kapitoly.

3.2 Učící techniky

Nejprve je třeba se přesvědčit, jestli se model zvládá učit všemi technikami, popsanými v první kapitole. To je základní požadavek, který musí model bezpodmínečně splňovat, aby byl použitelný pro simulování učení v aplikaci Chovatel zvířat.

3.2.1 Učení zachycením

Technika „zachycení“ je popsána v sekci 1.2.1 a jedná se o nejjednodušší způsob učení modelu. Jde o zvyšování pravděpodobnosti vykonání akce v daném stavu prostřednictvím odměňování právě této akce v daném stavu. To je ale příliš triviální. Navíc je potřeba zjistit, jaký má „zachycení“ vliv na pravděpodobnosti akcí v podobných stavech. Proto je v tomto případě cílem experimentu zvýšit pravděpodobnost vykonání určité akce v jednom stavu a zároveň v nějakém podobném stavu zvýšit pravděpodobnost vykonání jiné akce, která je pokud možno první akci nepodobná. To ovšem odpovídá cviku s papouškem, který je popsaný v první kapitole. Proto k tomuto experimentu použiji nastavení modelu pro papouška.

Cíl experimentu

Cílem tréninku je v modelu papouška zvýšit pravděpodobnost vykonání akce a_9 (tj. „zvuk dlouhý 1“) ve stavu „slyší povel 1“ a pravděpodobnost vykonání akce a_{10} (tj. „zvuk dlouhý 2“) ve stavu „slyší povel 2“. K tomu se používá zvukový stimul spárovaný se stimulem odměny. Cíl je splněn, pokud obě řečené pravděpodobnosti přesáhnou hodnotu 0.6.

Nastavení modelu

Použitý model je papoušek Alex, tj. $\alpha_p = 0.5$, $\alpha_r = 0.2$ a $\gamma = 0.7$. Podobnost akce a_9 s akcí a_{10} je 0.1. Podobnost stavů „slyší povel 1“ a „slyší povel 2“ je 0.1. Počáteční pravděpodobnost vykonání akce a_9 ve stavu „slyší povel 1“ je 0.1, stejně jako počáteční pravděpodobnost

vykonání akce a_{10} ve stavu „slyší povel 2“. Počet možných akcí v obou zmíněných stavech je 9. Z toho zajímavé jsou akce a_7 a a_8, protože podobnost akce a_7 s akcí a_9 i podobnost akce a_8 s akcí a_{10} je 0.35.

Výsledek experimentu

Cíle experimentu bylo ve všech případech dosaženo. Průměrná doba tréninku bez chyb, která byla nutná k dosažení cíle, byla 202.3 sekund. Směrodatná odchylka délky bezchybného pokusu byla 25.307 sekund. V případě tréninků, kdy byla před dosažením cíle jednou odměněna nesprávná akce, nebo nebylo odměněno vykonání správné akce, byla průměrná doba nutná k dosažení cíle 254.25 sekund. V případě dvou chyb 299 sekund. Směrodatná odchylka pokusů s jednou chybou byla 35.42 sekund a pokusů se dvěma chybami 69.521 sekund.

Diskuze

To, že se model umí učit „zachycením“, není překvapivé. Nepředpokládané není ani to, že v případě bezchybně provedených tréninků je směrodatná odchylka doby trvání pokusů výrazně menší, než když trénink obsahoval chyby.

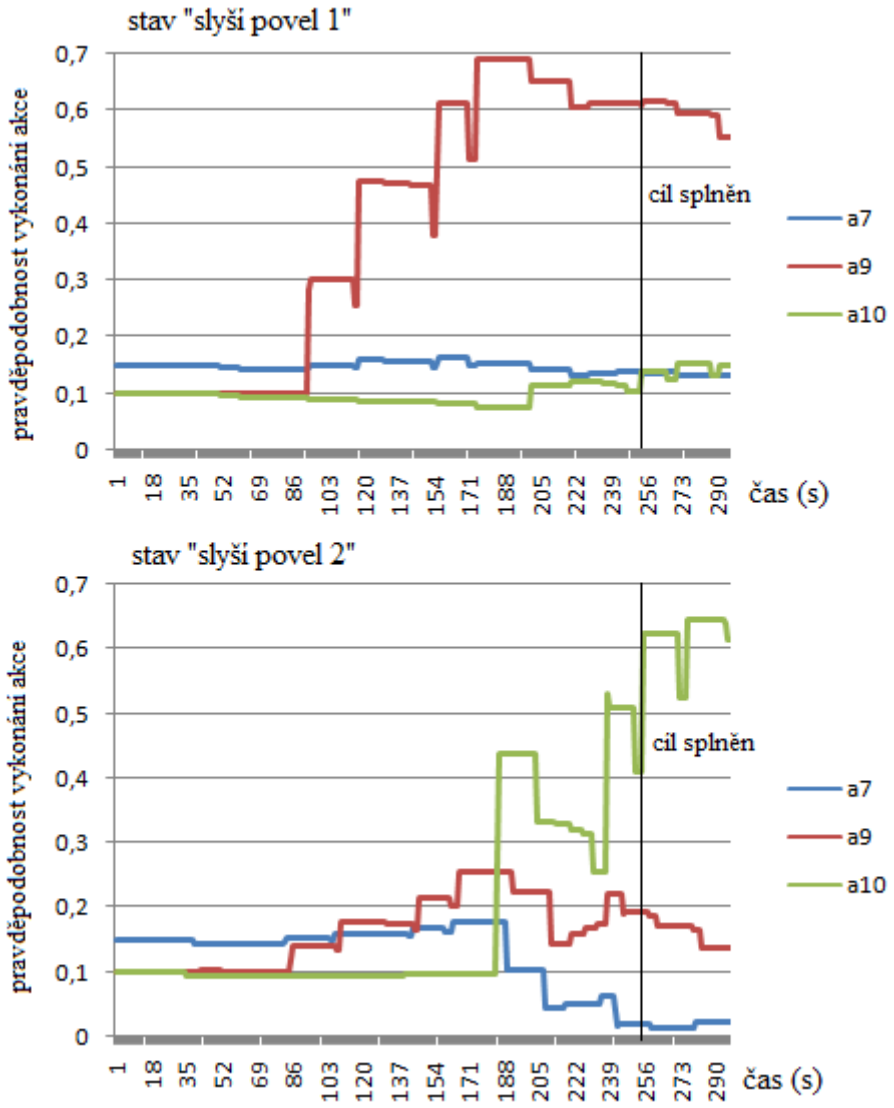
Obě cílové akce byly navzájem trochu podobné, takže by se mohlo zdát, že trénování jedné akce pomáhalo i tréninku druhé akce v podobném stavu. Jak je vidět na obrázku 12, který zobrazuje průběh jednoho z provedených tréninků, nebylo tomu tak. Na začátku tréninku učení akce a_9 ve stavu „slyší povel 1“ prakticky nezvyšovalo pravděpodobnost akce a_{10} ve stavu „slyší povel 2“. Naopak, po splnění cíle tréninku jsem dále odměňoval akci a_{10} ve stavu „slyší povel 2“. To zvyšovalo její pravděpodobnost vykonání i v podobném stavu „slyší povel 1“. Ovšem suma pravděpodobností v tomto stavu tak přesáhla 1 a pravděpodobnosti všech akcí byly přeškálovány (viz 2.4), což ve výsledku vedlo ke snížení pravděpodobnosti vykonání akce a_9, protože poměrově se více zvedla pravděpodobnost samotné akce a_{10}. Tomu by nepomohla ani vyšší podobnost akcí než 0.1, protože akce a_9 má ve stavu „slyší povel 1“ velmi vysokou pravděpodobnost vykonání, a tak by odměňování akce a_{10} vždy poměrově daleko více posílilo právě pravděpodobnost samotné akce a_{10}. Ta by po přeškálování rostla hlavně na úkor akce a_9 (protože všechny ostatní akce mají v daném stavu již tak malou pravděpodobnost vykonání).

Z tohoto důvodu by mělo být splnění cíle tohoto experimentu obtížnější s rostoucí podobností vybraných stavů (protože právě trénovaná akce by rostla v obou stavech najednou). A skutečně, po nastavení podobnosti obou stavů na 0.3 jsem provedl ještě jednu

sadu pokusů a k dosažení cíle experimentu bylo průměrně potřeba 312.5 sekund tréninku bez chyb.

Dalším poznatkem plynoucím z vykonání tohoto experimentu je to, že chybné odměnění akce podobné cílové akci (např. a_7) nemělo výrazně odlišný vliv na dobu trvání experimentu od chybného odměnění akce nepodobné cílové akci.

Zajímavé také je, že jinak závažná chyba neodměnění vykonání cílové akce má v případě tréninku „zachycením“ v porovnání s chybou odměnění nesprávné akce velice malý dopad na délku tréninku (na obrázku 12 je ve 205. sekundě stavu „slyší povel 2“ vidět pokles pravděpodobnosti akce v případě jejího neodměnění).



Obr. 12: Ukázkový průběh experimentu učení „zachycením“, učení akce a_9 ve stavu „slyší povel 1“ a akce a_{10} ve stavu „slyší povel 2“. Pokus obsahuje pouze jednu chybu – neodměnění akce a_{10} v 205. sekundě běhu pokusu.

3.2.2 Učení tvarováním

Další základní technika, kterou model musí být schopen simulovat, je učení „tvarováním“, viz sekce 1.2.2. Právě kvůli této technice byl do modelu zaveden mechanismus podobných akcí a stavů. Zde otestuji schopnost modelu vykonat jeden mezikrok této techniky, tj. učením akce v jednom stavu zvednutí pravděpodobnosti vykonání podobné akce v podobném stavu. Přitom tato podobná akce musí začínat s velice malou pravděpodobností vykonání.

Při „tvarování“ se hráč může dopustit několika různých chyb, které mohou mít různě velký dopad na trénink. V reálném tréninku jsou podle experta nejzávažnější tyto tři chyby: neodměnění prvního vykonání cílové akce mezikroku, odměnění nesprávné akce a neodměnění vykonání cílové akce mezikroku (potom, co již byla tato akce alespoň jednou odměněna). Neodměnění prvního vykonání cílové akce je bráno jako zvláštní chyba, protože při reálném tréninku má o něco větší dopad na délku tréninku, než pozdější neodměnění cílové akce. Vliv odměnění nesprávné akce by měl být srovnatelný s neodměněním prvního vykonání cílové akce.

Cílem tohoto experimentu je tedy kromě samotného učení „tvarováním“ také zjistit vliv jednotlivých druhů chyb na délku tréninku. K experimentu byl použit model psa.

Cíl experimentu

Cílem tréninku je v modelu psa zvýšit pravděpodobnost vykonání akce a_6 (tj. „pac do ruky“) ve stavu „ruka středně“ pomocí odměňování akce a_5 (tj. „pac mírně“) ve stavu „ruka těsně“ nebo odměňováním samotné akce a_6. Přitom ale akce a_6 začíná s velice nízkou pravděpodobností vykonání, takže se musí začít odměňováním akce a_5. K odměňování se používá zvukový stimul spárovaný se stimulem odměny. Cíl je splněn, pokud pravděpodobnost vykonání akce a_6 ve stavu „ruka středně“ přesáhne hodnotu 0.6.

Nastavení modelu

Použitý model je pes Kvído, tj. $\alpha_p = 0.6$, $\alpha_r = 0.2$ a $\gamma = 0.8$. Podobnost akce a_5 s akcí a_6 je 0.3. Podobnost stavů „ruka těsně“ a „ruka středně“ je 0.5. Model začíná v nastavení, kdy je naučený ve stavu „ruka těsně“ vykonávat akci a_5 – pravděpodobnost vykonání této akce je zde 0.81. Počáteční pravděpodobnost vykonání akce a_6 ve stavu „ruka středně“ je 0.03, pravděpodobnost vykonání akce a_5 ve stavu „ruka středně“ je 0.43.

Výsledek experimentu

Cíle experimentu bylo ve všech případech dosaženo. Ovšem odměňováním akce a_5 ve stavu „ruka těsně“ bylo ve všech případech dosaženo zvýšení pravděpodobnosti vykonání cílové akce a_6 pouze na hodnotu 0.1247. Dál se muselo postupovat přímým odměňováním akce a_6 ve stavu „ruka středně“.

Průměrná doba tréninku bez chyb, která byla nutná k dosažení cíle, byla 95.4 sekund. Směrodatná odchylka délky bezchybného pokusu byla 8.04 sekund. V případě tréninků, kdy byla před dosažením cíle jednou odměněna nesprávná akce, byla průměrná doba nutná k dosažení cíle 124.3 sekund a směrodatná odchylka délky pokusu byla 14.99 sekund. Pokud nebylo odměněno první vykonání akce a_6 ve stavu „ruka středně“, byla průměrná délka pokusů před dosažením cíle 130.6 sekundy a směrodatná odchylka pokusu 6.42 sekund. Pokud nebylo odměněno jedno pozdější vykonání cílové akce, byla průměrná délka tréninku 156.7 sekundy a směrodatná odchylka pokusu 6.53 sekund.

Diskuze

Část tréninku „tvarováním“, kdy odměňováním podobné akce v podobném stavu roste pravděpodobnost vykonání cílové akce, koresponduje s problémem zmíněným v předchozím experimentu. Proto není překvapivé, že pravděpodobnost akce a_6 odměňováním jiné akce nepřesáhla hodnotu 0.1247. Tato hodnota je ovšem postačující k tomu, aby bylo možné akci a_6 odměňovat přímo (tj. vlastně „zachytit“), protože se vykonává dostatečně často.

Zajímavý je vliv různých chyb na dobu tréninku. Chyba neodměnění prvního vykonání cílové akce a chyba odměnění nesprávné akce průměrně prodloužily dobu tréninku zhruba stejně, takže se jejich vážnost dá (i přes rozdílné střední odchylky délky tréninků) považovat za srovnatelnou.

Jinak je tomu u chyby neodměnění již dříve odměněné cílové akce. Ta v průměru prodloužila trénink dvojnásobně v porovnání s předchozími chybami. Je to dáno tím, že jednou odměněná akce má nastavenu nenulovou očekávanou odměnu. Čím je tato očekávaná odměna vyšší, tím více klesne pravděpodobnost této akce v případě, že její vykonání není odměněno. Tím se znehodnotí např. jedno předchozí odměnění cílové akce.

Na druhou stranu u chyby neodměnění prvního výskytu cílové akce je její očekávaná odměna stále nulová, takže její pravděpodobnost se nezmění a uživatel může pokračovat, jako by se nic nestalo.

Model tedy sice lze učit „tvarováním“, ale požadavky na vážnost jednotlivých druhů chyb nejsou úplně splněny. Předpokládá se totiž, že neodměnění prvního vykonání cílové

akce je závažnější než neodměnění následného nevykonání této akce. Tento problém ovšem není zásadní, protože důsledky zmíněné chyby jsou *větší* než předpoklad, tj. tato chyba neztrácí pro hráče hry poučný význam, ale naopak.

3.2.3 Učení sekvence akcí

Poslední ze tří základních technik učení modelu je řetězení akcí, viz sekce 1.2.3. Jak bylo popsáno v 2.5 a 2.6.2, aby model umožňoval řetězení akcí, je potřeba speciální nastavení některých akcí. Takové akce obsahuje pouze model lemura (ostatní zvířata v programu Chovatel zvířat sekvence akcí učit nelze). Proto budu řetězení akcí demonstrovat na modelu lemura, kterého naučím vykonat sekvenci dvou akcí. Protože tento trénink je komplexnější než předcházející experimenty, uvádím i popis postupu při jednotlivých trénincích.

Cíl experimentu

Cílem tréninku je v modelu lemura zvýšit pravděpodobnost vykonání akce a_{12} (tj. „vchází do bedny“) ve stavu „bedna se dnem“ a dále zvýšit pravděpodobnost akce a_3 (tj. „vrtí hlavou“) ve stavu „v bedně“. K odměňování se používá zvukový stimul spárovaný se stimulem odměny. Cíl je splněn, pokud pravděpodobnost vykonání akce a_{12} ve stavu „bedna se dnem“ i pravděpodobnost vykonání akce a_3 ve stavu „v bedně“ přesáhnou hodnotu 0.6.

Nastavení modelu

Použitý model je lemur Emil, tj. $\alpha_p = 0.6$, $\alpha_r = 0.3$ a $\gamma = 0.7$. Nutný předpoklad ke splnění tréninku je ten, že vykonání akce a_{12} ve stavu „bedna se dnem“ přesune model do stavu „v bedně“. Podobnost stavu „sedí“ a stavu „v bedně“ je 0.1, stavu „bedna se dnem“ a „v bedně“ 0.15. Akce a_3 a a_{12} si navzájem nejsou podobné. Akce a_{12} je navíc nemožná ve stavu „sedí“ i ve stavu „v bedně“. Počáteční pravděpodobnost vykonání akce a_{12} ve stavu „bedna se dnem“ je 0.01, pravděpodobnost vykonání akce a_3 ve stavu „v bedně“ je 0.1.

Postup při tréninku

Uživatel nejprve zvýší pravděpodobnost vykonávání akce a_3 ve stavu „sedí“ a tím i v podobném stavu „v bedně“ (tento krok není zásadní, protože již počáteční pravděpodobnost akce a_3 ve stavu „v bedně“ je dostatečně vysoká pro trénink „zachycením“). Dále pomocí učení „tvarováním“ uživatel zvedne pravděpodobnost vykonání akce a_{12} ve stavu „bedna se dnem“.

To odpovídá postupnému naučení jednotlivých akcí sekvence, což je první krok řetězení z 1.2.3. Zbytek tréninku už spočívá jen ve zvětšování pravděpodobností obou akcí prostřednictvím „zachycení“. Výsledkem je, že lemur s velkou pravděpodobností vleze do bedny a tam zavrtí hlavou.

Výsledek experimentu

Cíle experimentu bylo ve všech případech dosaženo. Průměrná délka bezchybného tréninku byla 159.2 sekund a směrodatná odchylka 40.54 sekund.

Diskuze

Trénink kombinuje techniky „tvarování“ a „zachycení“, přičemž hlavně „zachycení“, které je často závislé na stochastickém výběru akce s malou pravděpodobností, ovlivňuje velikost odchylky jednotlivých tréninků od průměrné délky experimentu. To je ovšem v souladu s výsledky experimentu z 3.2.1.

Sekvence dvou akcí naučená v tomto experimentu se s vysokou pravděpodobností provádí jako celek (jakmile se vykoná první akce (a_{12}), je šance na vykonání celé sekvence rovna pravděpodobnosti vykonání druhé akce (a_3), která je vysoká). Problematické by bylo až řetězení tří a více akcí. Pravděpodobnost vykonání jedné akce v daném stavu totiž velice obtížně dosahuje hodnot větších, než např. 0.8[13]. To je zapříčiněno tím, že ostatní akce v daném stavu dosahují velmi malých pravděpodobností. Když se pak znovu odměňuje akce s vysokou pravděpodobností vykonání, celková suma pravděpodobností v daném stavu přesáhne 1. Při škálování (viz 2.4) se ale hodnoty pravděpodobnosti zmenšují v poměru k jejich velikosti, tj. velmi málo pro ostatní akce a hodně pro akci s vysokou pravděpodobností, která tak ve výsledku roste čím dál tím pomaleji.

Sekvence tří akcí se tak vykoná jako celek v lepším případě s pravděpodobností zhruba 0.64, ale v případě, že jsou jednotlivé akce natrénovány např. na pravděpodobnost 0.6 jako v experimentech této kapitoly, celá sekvence se provede s pravděpodobností zhruba 0.36.

Pro potřeby programu Chovatel zvířat ale stávající řetězení akcí postačuje, protože ve hře není třeba tvořit delší sekvence akcí. Navíc pokud sekvenci akcí přeruší např. 1 sekundu trvající akce „nedělá nic“ nebo jiná krátká klidová akce, není to závažné, protože hráč sleduje pouze grafický výstup programu. Sekvence akcí, které občas přeruší klidová akce, jsou naopak předností proti nepřirozeným sekvencím chovajícím se jako nepřerušitelný celek.

[13] Je to samozřejmě ovlivněno počtem možných akcí ve stavu, jejich vzájemných podobností a nastavením podobných stavů. V této kapitole se ale jako vysoká pravděpodobnost vykonání bere již hodnota 0.6.

3.3 Další experimenty

Kromě základních požadavků na schopnost učení specifickými technikami bylo v kapitole 2 formulováno i několik dalších požadavků na vlastnosti modelu. Mezi ně patří například možnost odhadnout přibližnou dobu, kterou bude daný trénink trvat, vliv použití klasického podmiňování na efektivitu trénování nebo možnost nastavení obtížnosti tréninku.

3.3.1 Odhad doby tréninku

Model je navržen pro použití ve hře, zaměřené na využití v rámci vyučovací hodiny. Ta je ale časově striktně omezená, a tak si učitel musí být případné použití hry schopen naplánovat. Proto by učení modelu mělo probíhat s garantovanou rychlostí postupu (za předpokladu tréninku bez, nebo s několika málo chybami). Přestože je model stochastický, délka experimentů s učením „tvarování“ měla standardní odchylku 6-15 sekund, při průměrné délce celého tréninku zhruba 2 minuty.

Větší variabilitu měly tréninky používající učení „zachycením“, které při průměrné délce tréninku 3-5 minut měly standardní odchylku v rozmezí 25-70 sekund (v závislosti na počtu chyb při tréninku).

Otázkou je, do jaké míry je malá variabilita doby trvání experimentů s „tvarováním“ zapříčiněna jejich malou průměrnou délkou a jestli se u komplexnějšího tréninku projeví učení „zachycením“ jinak.

Proto jsem testoval celý cvik se psem, jehož postup je popsaný v sekci 1.3. Popsaný trénink obsahuje pět mezikroků „tvarování“ a následně zvyšování pravděpodobnosti vykonání akce zvednutí packy „zachycením“.

Pokud bych chtěl předem odhadnout dobu, kterou trénink bude trvat, vezmu v úvahu výsledky experimentů na jednom mezikroku „tvarování“ z 3.2.2. U nich byla průměrná délka bezchybného tréninku 95.4 sekund. Ovšem jeden trénink v tom případě zahrnoval i spárování zvukového stimulu se stimulem odměny, které se u delšího tréninku musí dělat jen jednou. Trénink také zahrnoval „zachycení“ cílové akce (protože cílová pravděpodobnost byla vysoká). To se opět v delším tréninku dělá jen jednou. Samotné naučení jednoho mezikroku „tvarování“ tak v těchto trénincích trvalo odhadem jen necelou minutu. Takže za předpokladu pěti mezikroků učení „tvarováním“, asociace odměny se zvukem a zvyšování pravděpodobnosti cílové akce docházím k odhadu 4*60 + 95.4 = 335.4 sekund pro dobu trvání bezchybného tréninku se psem. Protože je potřeba pouze hrubý odhad, mohuu tento předpoklad zaokrouhlit na 5-6 minut.

Stejným způsobem mohu dojít k odhadu délky tréninku, při kterém uživatel udělá jednu chybu. V takovém případě je odhad 4*60 + 130 = 370 sekund (chyba pouze v jednom mezikroku). Tedy o málo více než 6 minut.

Cíl experimentu

Cílem tréninku je v modelu psa zvýšit pravděpodobnost vykonání akce a_7 (tj. „zvedá packu“) ve stavu „slyší povel“. K odměňování se používá zvukový stimul spárovaný se stimulem odměny. Cíl je splněn, pokud pravděpodobnost vykonání akce a_7 ve stavu „slyší povel“ přesáhne hodnotu 0.6.

Nastavení modelu

Použitý model je pes Kvído, tj. $\alpha_p = 0.6$, $\alpha_r = 0.2$ a $\gamma = 0.8$. Stav „slyší povel“ je podobný stavu „slyší povel + vidí ruku“ (0.5), ten je zase podobný stavu „ruka vysoko“ (0.5). Stav „ruka vysoko“ je podobný stavu „ruka středně“ (0.4), který je podobný stavu „ruka těsně“ (0.5). Stav „ruka těsně“ je podobný stavu „cítí dotyk“ (0.6). Akce „nedělá nic“ (a_2) je podobná akci „pac mírně“ (a_5), která je podobná akci „pac do ruky“ (a_6) a ta je podobná akci „zvedá packu“ (a_7). Podobnost všech těchto akcí je 0.6 a žádná z nich není podobná ostatním akcím modelu. Podobnost akce a_5 a a_7 je 0.3. Počáteční pravděpodobnost vykonání akce a_5, a_6 i a_7 ve všech stavech, kde jsou tyto akce možné, je 0.01.

Výsledek experimentu

Cíle experimentu bylo ve všech případech dosaženo. Průměrná délka bezchybného tréninku byla 282.35 sekund a směrodatná odchylka 73.79 sekund. V případě jednoho odměnění nesprávné akce nebo neodměnění správné akce byla průměrná délka experimentu 323.9 sekund a standardní odchylka 69.23 sekund. V případě dvou chyb byla průměrná délka experimentu 395.5 sekund a směrodatná odchylka 91.32 sekund. Nakonec v případě pěti chyb byla průměrná délka experimentu 580 sekund a směrodatná odchylka 79.4 sekund.

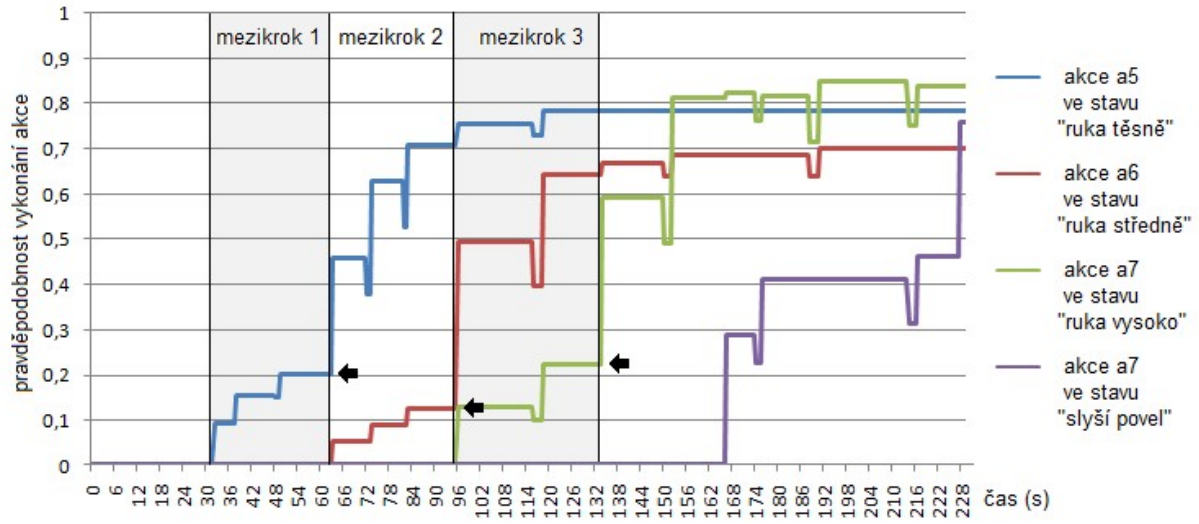
Diskuze

Výsledky experimentu ukazují, že model se v případě bezchybného tréninku učil o necelou minutu rychleji, než byl můj odhad výše. O zhruba stejný rozdíl se učil rychleji i v případě jedné chyby uživatele. To ale není způsobeno tím, že by se pravděpodobnost a_7 začala při tréninku zvyšovat již o jeden mezikrok napřed, viz obrázek 13. Přestože jsou si totiž akce a_5 a a_7 trochu podobné, stavy „ruka těsně“ a „ruka vysoko“ si podobné nejsou. Na obrázku 13 jsou

znázorněny pravděpodobnosti akcí, které jsou trénovány ve třech po sobě jdoucích mezikrocích učení „tvarováním“, tj. jde o zvyšování pravděpodobnosti akce pomocí odměňování podobné akce v jiném stavu. V experimentu z 3.2.2 se učí právě druhý zobrazený mezikrok – pomocí akce a_5 ve stavu „ruka těsně“ zvyšovat pravděpodobnost vykonání akce a_6 ve stavu „ruka středně“. Z obrázku 13 je ale patrné, že při tomto mezikroku roste pravděpodobnost trénované akce pomaleji než v ostatních zobrazených případech. To je dáno tím, že stavy „ruka těsně“ a „ruka středně“ mají podobnost pouze 0.4 a všechny ostatní mezikroky jsou mezi stavy, které si jsou podobnější[14], viz nastavení experimentu. Proto lze ostatní mezikroky v průměru splnit rychleji než mezikrok z experimentu 3.2.2.

Rozdílná doba potřebná k splnění mezikroku je v souladu s tvrzením v popisu učení „tvarováním“, že kratší mezikroky – obsahující podobnější cílové akce v podobnějších stavech – se učí efektivněji, viz 1.2.2.

Ze dvou případů odhadu délky tréninku ale pořád nemohu usuzovat, že ji lze obecně odhadnout. Hlavní poznatek vyplývající z tohoto experimentu tak je, že jedna chyba v rámci delšího tréninku měla přibližně stejný vliv na prodloužení tréninku jako v případě krátkého experimentu z 3.2.2.



Obr. 13: Ukázkový průběh tréninku psa. Cílem celého tréninku bylo dosáhnout pravděpodobnosti 0.6 u akce a_7 ve stavu „slyší povel“. Na obrázku jsou dále znázorněny pravděpodobnosti tří akcí ve třech stavech, které v postupu tréninku tvoří tři po sobě jdoucí mezikroky učení „tvarováním“. Mezikrok 1: akce a_5 je učena odměňováním nezobrazené akce; mezikrok 2: akce a_6 je učena odměňováním akce a_5; mezikrok 3: akce a_7 je učena odměňováním akce a_6. Šipky označují pravděpodobnost trénované akce, kterou dosáhla prostřednictvím odměňování podobné akce v jiném stavu. Nejmenší je tato pravděpodobnost v případě mezikroku 2, kdy je prostřednictvím akce a_5 trénována akce a_6.

[14] Podobnosti stavů byly nastaveny po konzultaci s profesionálním trenérem zvířat.

Dalším zajímavým poznatkem může být to, že vliv jediné chyby během celého tréninku prodloužil tento trénink zhruba o 40 sekund, ale další chyba prodloužila trénink zhruba o 70 sekund (trénink se dvěma chybami). To ovšem může být jen nedostatkem pokusů (5 je na delší trénink pravděpodobně málo, také všechny směrodatné odchylky jsou velké). Navíc jak bylo ukázáno v 3.2.2, různé druhy chyb prodlužují trénink různě dlouho a při provádění tohoto experimentu jsem druhy chyb nerozlišoval.

3.3.2 Vliv klasického podmiňování na trénink

Jedním z požadavků na model je i to, aby odměňování prostřednictvím zvukového stimulu bylo při tréninku efektivnější až do té míry, že ho hráč hry bude užívat prakticky vždy. Jaký vliv má odměňování pomocí zvukového stimulu, budu zjišťovat následující sadou experimentů.

Trénovat budu opět psa zvednout packu (stejně jako v předchozím experimentu a v sekci 1.3). Tento trénink nejprve provedu bez použití zvukového stimulu asociovaného s odměnou (tedy jen s opravdovým stimulem odměny)[15].

Efektivita trénování, při kterém se používá zvukový stimul asociovaný s odměnou, je ale závislá na síle této asociace, tj. na tom, jak často po tomto stimulu skutečně přijde odměna. Proto provedu několik sad experimentů, které budou mít různé frekvence případů, v nichž je zvukový stimul skutečně následován odměnou.

Výsledky obou typů experimentů navzájem porovnám. Předpoklad ale je, že odměňování zvukovým stimulem, který následuje skutečná odměna alespoň v 50% případů, bude nejefektivnější.

Cíl experimentu

Cíl experimentu je stejný jako v sekci 3.3.1. Tj. cíl je splněn, pokud pravděpodobnost vykonání akce a_7 ve stavu „slyší povel“ přesáhne hodnotu 0.6.

Nastavení modelu

Použitý model je pes Kvído, tj. $\alpha_p = 0.6$, $\alpha_r = 0.2$ a $\gamma = 0.8$. Podobně i všechna ostatní nastavení jsou stejná jako v sekci 3.3.1.

[15] Fáze tréninku se psem, kdy je potřeba zastavit psí packu ve výšce jeho hlavy ovšem nelze splnit bez použití zvukového stimulu asociovaného s odměnou. Proto je v tomto mezikroku tento zvukový stimul používán. V rámci celého tréninku jde zhruba o 15% všech případů, kdy je třeba psa odměnit.

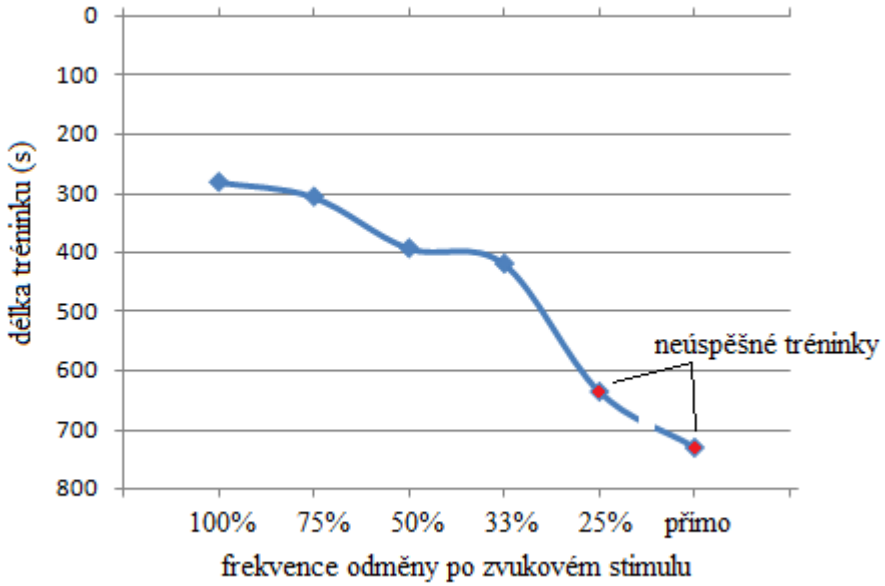
Výsledek experimentu

Cíle experimentu bylo dosaženo v případě používání zvukového stimulu asociovaného s odměnou, který byl alespoň v 33% případů následován skutečným stimulem odměny. Konkrétně trénink průměrně trval 420 sekund v případě 33% skutečné odměny, 394 sekund v případě 50% odměny a 307 sekund v případě 75% odměny.

Pokud byl zvukový stimul následován stimulem odměny pouze v 25% případech, nepřesáhla maximální pravděpodobnost cílové akce ve všech trénincích 0.25. Do toho stavu se trénink průměrně dostal po 636 sekundách.

V případě, že nebyl používán zvukový stimul asociovaný s odměnou, ale přímo stimul odměny, nepřesáhla maximální pravděpodobnost cílové akce ve všech trénincích 0.5. Do toho stavu se trénink průměrně dostal po 730.5 sekundách.

Tyto výsledky společně s výsledky s předchozího experimentu (kde po zvukovém stimulu následovala skutečná odměna vždy) znázorňuje graf na obrázku 14.



Obr. 14: Průměrné délky experimentů při tréninku psa v závislosti na frekvenci s jakou následoval po zvukovém stimulu stimul odměny. Úplně vpravo průměrná doba tréninků, při kterých byl používán pouze stimul odměny. V těchto trénincích a v trénincích, kdy odměna za zvukovým stimulem následovala pouze v 25% případů, nebylo možné „zachycením“ dostatečně zvýšit pravděpodobnost vykonání cílové akce.

Diskuze

Výsledky tohoto experimentu dopadly podle předpokladů dobře v případě, že se používá zvukový stimul k označení odměny, který více než v 50% případů následuje i skutečný stimul odměny. Vzhledem k tomu, že hodnota r_k (viz popis klasického podmiňování modelu v 2.5.1)

se nastavuje v případě přímého odměňování stimulem odměny na 0.5, není překvapivé ani to, že pokud při takových trénincích pravděpodobnost cílové akce nepřekročila hodnotu 0.5. Kromě malé hodnoty okamžité odměny je to způsobeno také tím, že uživatel odměňuje pouze stimulem odměny, která ale díky tomu přichází pozdě a do cílového stavu a akce se pouze propaguje, viz algoritmus na obrázku 5.

Právě malá hodnota okamžité odměny zapříčinila i neúspěch tréninku, kdy zvukový stimul následovala odměna pouze ve 25% případů. Zde je patrný poměrně velký rozdíl mezi tímto případem a případem, kdy odměna následovala ve 33% případů. Tento rozdíl je ale zapříčiněn tím, že první neodměnění skutečnou odměnou nesnižuje hodnotu r_k (viz sekce 2.5.1), takže její hodnota byla v obou případech velmi rozdílná.

3.3.3 Vliv změny obtížnosti na trénink

Obtížnost učení modelu nejzásadněji ovlivňují konstanty α_p, α_r a γ (podobnosti akcí a stavů jsou totiž u jednotlivých tréninků nastaveny za pomoci profesionálního trenéra zvířat a odrážejí jejich fyzické, ne kognitivní, možnosti).

Abych přesně demonstroval vliv konkrétního nastavení těchto tří parametrů, provedl jsem několik sad experimentů s jejich různým nastavením. Vzhledem k tomu, že kombinací nastavení může být mnoho, vždy jsem měnil pouze jeden z těchto parametrů a zbylé zafixoval na hodnotách použitých v experimentech výše. Kromě toho jsem ale provedl i experiment s nastavením používaným v programu Chovatel zvířat pro obtížnější verzi psa (kde jsou najednou změněny dva z těchto parametrů).

Cíl experimentu

Cíl experimentu jsem zvolil stejný jako v sekci 3.2.2. Tedy zvýšit pravděpodobnost vykonání akce a_6 (tj. „pac do ruky“) ve stavu „ruka středně“ pomocí odměňování akce a_5 (tj. „pac mírně“) ve stavu „ruka těsně“ nebo odměňováním samotné akce a_6. Cíl je splněn, pokud pravděpodobnost vykonání akce a_6 ve stavu „ruka středně“ přesáhne hodnotu 0.6.

Nastavení modelu

Použitý model ve většině případů neodpovídá žádnému nastavení ze hry Chovatel zvířat (pokud ale není řečeno jinak, používá se nastavení modelu jako u psa Kvída, tj. $\alpha_p = 0.6$, $\alpha_r = 0.2$ a $\gamma = 0.8$). Pouze jedna sada experimentů používá model v nastavení psa Huga, tj. $\alpha_p = 0.4$, $\alpha_r = 0.2$ a $\gamma = 0.4$.

Jiná použitá nastavení jednotlivých parametrů jsou: $\alpha_p = 0.1$; $\alpha_p = 0.3$; $\alpha_p = 0.9$; $\alpha_r = 0.1$; $\alpha_r = 0.5$; $\alpha_r = 0.8$; $\gamma = 0.1$; $\gamma = 0.4$.

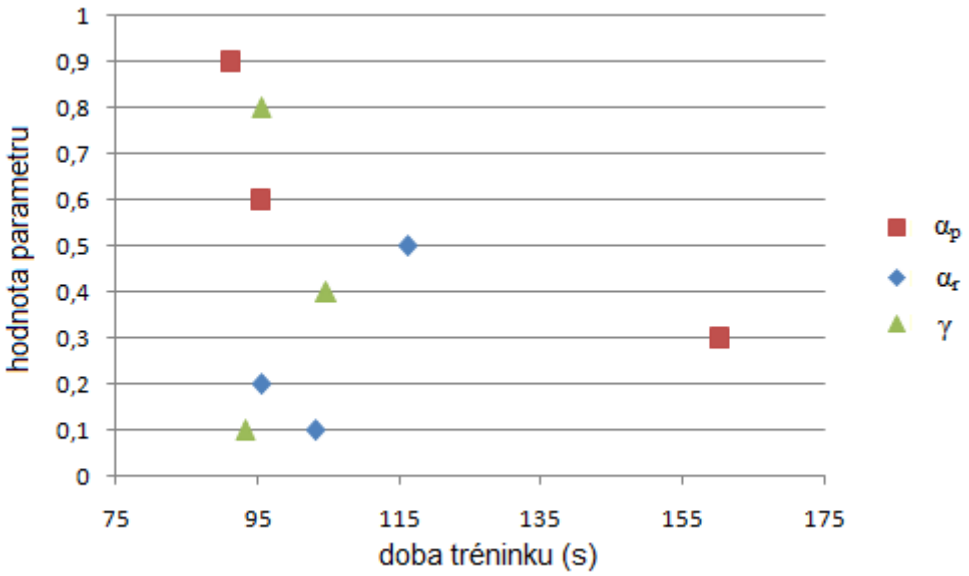
Podobnost akce a_5 s akcí a_6 je 0.3. Podobnost stavů „ruka těsně“ a „ruka středně“ je 0.5. Model začíná v nastavení, kdy je naučený ve stavu „ruka těsně“ vykonávat akci a_5 – pravděpodobnost vykonání této akce je zde 0.81. Počáteční pravděpodobnost vykonání akce a_6 ve stavu „ruka středně“ je 0.03, pravděpodobnost vykonání akce a_5 ve stavu „ruka středně“ je 0.43.

Výsledek experimentu

Cíle experimentu bylo dosaženo ve všech případech kromě nastavení $\alpha_p = 0.1$ a $\alpha_r = 0.8$. V těchto dvou případech byly všechny experimenty ukončeny po 5 minutách, kdy se pravděpodobnost cílové akce již prakticky neměnila a byla nižší než 0.6.

Průměrná délka ostatních experimentů byla v případě $\alpha_p = 0.3$ 160.2 sekund, $\alpha_p = 0.9$ 91.2 sekund, $\alpha_r = 0.1$ 103.1 sekund, $\alpha_r = 0.5$ 116.2 sekund, $\gamma = 0.1$ 93.1 sekund a $\gamma = 0.4$ 104.7 sekund. Průmětné doby tréninků znázorňuje graf na obrázku 15.

V případě nastavení modelu na psa Huga ($\alpha_p = 0.4$, $\alpha_r = 0.2$ a $\gamma = 0.4$) byla průměrná doba tréninku 102.5 sekund.



Obr. 15: Průměrné délky experimentů při tréninku psa v závislosti na změně jednoho ze tří parametrů α_p, α_r nebo γ. Hodnota změněného parametru je znázorněna na svislé ose grafu.

Diskuze

Velký vliv parametru α_p na délku tréninku zde není překvapivý. Tento parametr totiž udává podle vztahu (3) míru změny pravděpodobnosti akce v jednom učícím kroku modelu. Pokud

je jeho hodnota nízká, je při trénincích potřeba více odměnění k dosažení stejné pravděpodobnosti a tím se doba tréninku značně prodlužuje. Naopak pokud je tato hodnota příliš vysoká, změny pravděpodobnosti jsou moc velké a i když je průběh bezchybného trénink rychlý, jediná chyba může vést k drastickému snížení pravděpodobnosti a tím k jeho neúměrnému prodloužení. Vývoje pravděpodobností na obrázcích 12 a 13 ukazují, že pokud $\alpha_p = 0.5$, lze dosáhnout vysoké (větší než 0.6) pravděpodobnosti vykonání akce po třech odměněních (obrázek 12) a pokud $\alpha_p = 0.6$, stačí ke stejnému výsledku pouze dvě odměnění (obrázek 13). V případě $\alpha_p = 0.9$ ale stačilo v průměru 1.4 odměnění.

Na druhou stranu když bylo $\alpha_p = 0.3$, bylo třeba 8 odměnění k dosažení stejného výsledku a v případě $\alpha_p = 0.1$ byl po 10 odměnění další přírůstek pravděpodobnosti menší než 0.01 za jedno odměnění. Očekávaná odměna za vykonání odměňované akce totiž dosáhla maximální hodnoty 1 a rozdíl očekávaného zisku, který počítá s pesimistickým odhadem v aktuálním stavu, byl díky tomu vždy velice malý.

Změny v nastavení konstanty α_r neměly velký vliv na dobu tréninku, pokud zůstaly menší než 0.5. Ovšem v případě, že hodnota této konstanty byla 0.5, celkovou dobu tréninku to značně prodlužovalo. Při nastavení $\alpha_r = 0.8$ pak dokonce trénink nebylo možné dokončit, protože se pravděpodobnost cílové akce ustálila na nízké hodnotě. K vysvětlení tohoto chování přispívá následující rozbor.

Parametr α_r podle vztahu (2) ovlivňuje velikost změn očekávané odměny právě vykonané akce. Pokud je jeho hodnota velká, očekávaná odměna akce po každém odměnění významně vzroste. To ale zmenšuje následující odhady rozdílu zisku z vykonání této akce počítané podle vztahu (1), protože se použije právě zvětšená hodnota očekávané odměny. Ve svém důsledku to tedy znamená menší nárůst pravděpodobnosti vykonání akce po jejím opakovaném odměnění a naopak při jednom neodměnění to znamená velký negativní rozdíl odhadu zisků a velké snížení pravděpodobností (protože odhad odměny F je pesimistický a nemůže nabývat stejných hodnot jako očekávaná odměna – většinou platí $\gamma < 1$).

Poměrně překvapivý je malý vliv parametru γ na dobu tréninku. Tento parametr ovlivňuje do jaké míry je brán v potaz odhad F ve vztahu (1) – tj. jak moc záleží na budoucím stavu modelu při odhadování obdrženého zisku. To, že na budoucím stavu modelu moc nezáleží a dokonce trénink s jeho nižší hodnotou v průměru trvá kratší dobu, se dá vysvětlit tím, že model většinou při výpočtu rozdílu očekávaných odměn skončí v základním stavu, kde je odhad F velmi malý sám o sobě.

3.4 Diskuze výsledků experimentů

Experimenty popsané v této kapitole ukazují, že model se umí učit všemi třemi požadovanými technikami způsobem postačujícím pro využití v aplikaci Chovatel zvířat. Experiment 3.3.1 a hodnoty středních odchylek všech tréninků také naznačují, že ačkoli doba nutná k provedení tréninku není jednotná, naprostá většina tréninků s malým počtem chyb nepřesáhne dvojnásobek průměrné doby ukončení daného tréninku bez chyb.

Zde se dá namítnout, že každý hráč používá stimuly s jinou rychlostí a exaktnější metoda měření délky experimentu tak je spíše záznam počtu použitých stimulů (resp. kliknutí uživatele). Ovšem protože jsem zde všechny experimenty prováděl osobně, rychlost klikání nebyla v jednotlivých trénincích výrazně odlišná. Je ale pravdou, že jiný uživatel programu může dosahovat výrazně odlišných časů (i když poměry časů mezi jednotlivými experimenty by měly zůstat stejné).

V experimentech jsem dále neřešil případy, kdy uživatel při tréninku udělá více než 5 závažných chyb. V takových případech se délka tréninku může velice prodlužovat a student by ztratil možnost trénink dokončit. Řešení podobných situací je plně v režii učitele, který může model např. resetovat nebo ho nastavit do předem naučené fáze tréninku.

Kapitola 4

Evaluace učení s programem

Vliv hraní Chovatele zvířat na znalosti studentů byl evaluován na čtyřech gymnáziích v Praze a okolí ve spolupráci s Mgr. Cyrilem Bromem, PhD. a MUDr. Kamilem Klementem. Protože jsem evaluaci nedělal samostatně, uvádím v této práci pouze stručná fakta a závěry, které z této evaluace vzešly. Výsledky i průběh evaluace jsou detailně popsány v [6].

Evaluace byla provedena na čtyřech gymnáziích v Praze a okolí a celkem se jí účastnilo 134 studentů ve věku od 15 do 18 let. Každá testovaná třída byla rozdělena zhruba na poloviny, které odděleně podstoupily 90-ti minutovou přednášku (tj. dvě vyučovací hodiny – standardní praktika).

Předmětem přednášek byl úvod do problematiky učení zvířat doplněný konkrétním případem postupu, kterým lze naučit psa zvedat packu na slovní příkaz. Jednalo se o trénink popsaný v sekci 1.3.

4.1 Obsah přednášek

První vyučovací hodina obou přednášek byla pro obě skupiny v zásadě podobná a zabývala se obecnými fakty ohledně druhů podmiňování, etologii, behaviorismu atp. Tato hodina byla také u obou polovin oživena promítáním tematických videí, ale jinak se jednalo prakticky o klasickou výkladovou přednášku.

Ve druhé hodině u první poloviny studentů pokračoval výklad zhruba ve stejném duchu jako doposud. Studentům byly vysvětleny podrobnosti a zmíněny zajímavosti u určitých specifičtějších pojmů, jako je habituace, vtištění, sensitizace atp. (více viz např. [21]). Výklad byl opět doprovázen pouštěním videí a ke konci přednášky následoval podrobný postup příkladu, jak lze psa naučit na slovní povel zvednout přední packu. Tento postup je totožný s postupem, na který je designován cvik se psem v Chovateli zvířat a je přesně popsaný v sekci 1.3.

Druhá skupina studentů na rozdíl od první začala svou druhou hodinu rovnou vysvětlením cviku se psem. Čas spotřebovaný na vysvětlování postupu při cviku byl ale v případě této skupiny zhruba poloviční a studenti si ihned mohli vysvětlený cvik vyzkoušet hraním Chovatele zvířat. S programem pak pod dohledem učitele experimentovali až téměř do konce hodiny.

Posledních zhruba 15 minut bylo u obou skupin věnováno vyplňování dotazníků.

4.2 Metodika analýzy dat

Analýza vlivu práce s programem na získané znalosti byla prováděna prostřednictvím dvou dotazníků. První z nich studenti vyplňovali ihned po skončení přednášky a druhý po uplynutí jednoho měsíce od přednášky.

Dotazníky obsahovaly tři kategorie otázek. První kategorie byla zaměřena na pocity studenta z přednášky, jeho zkušenosti s prací na počítači atp. Vyhodnocení tohoto typu otázek je detailně diskutováno v [6] a zde jej dále zmíním pouze okrajově. Dalších devět otázek pak bylo zaměřeno na znalosti nabyté během přednášky. Tyto otázky tvořily v dotaznících jeden celek, ale ve skutečnosti byly zaměřením rozděleny na obecné otázky z vykládaných znalostí (dále G1) a na otázky týkající se pozitivní zpětné vazby při tréninku zvířat, tj. na téma, které má procvičovat hra (dále G2). Přesné znění otázek všech kategorií u obou dotazníků viz [15].

Anonymní odpovědi studentů na otázky kategorií G1 a G2 hodnotili dva nezávislí specialisté na trénování zvířat ve stupnici 1 až 3 (1 – špatně, 2 – částečně dobře, 3 – správně). Ovšem Spearmanův koeficient korelace hodnocení obou expertů byl vysoký, přesněji 0.83. Navíc většina (79.5%) odpovědí byla ohodnocena stejně oběma hodnotiteli. Proto byla obě jejich hodnocení pro účely evaluace u každé odpovědi zprůměrována.

V této práci prezentuji výsledky získané z dotazníků od 100 studentů (z celkového počtu 134 zúčastněných). Třída studentů z gymnázia v Kladně totiž podstoupila pouze jednohodinovou přednášku. Proto tito studenti nemohli odpovídat na otázky typu G1 v obou dotaznících. Dále nebyli zahrnuti studenti, kteří byli nepřítomní po měsíci od přednášky, a proto nevyplnili druhý dotazník.

Příslušnost studentů do různých tříd nebo věkových kategorií byla nadále zanedbávána. Dotazníky od studentů byly pouze rozděleny na dvě skupiny podle toho, jestli si při přednášce hráli s programem, nebo nikoli. Skupina, která podstoupila pouze výkladovou přednášku tak měla 49 studentů a skupina s hrou 51 studentů, viz tabulka 1. U takto vytvořených skupin pak byly prostřednictvím analýzy rozptylu (ANOVA, viz [7]) porovnávány rozdíly v odpovědích u otázek kategorie G1 a G2 (tj. oznámkovaných faktických znalostí). Analýza rozptylu byla použita namísto t-testu proto, že se navíc testovaly i rozdíly v odpovědích vzhledem k pohlaví studenta.

Na analýzu odpovědí na otázky ohledně pocitů z přednášky byl použit neparametrický Wilcoxnův test, protože na rozdíl od ostatních otázek nebyly v tomto případě odpovědi

studentů rozděleny normálně [3]. Výsledky této analýzy ovšem mají spíše pedagogický význam a jako takové je zmíním pouze v diskuzi k výsledkům evaluace.

„Velikost změny“ (effect size [x]) mezi skupinami u odpovědí na otázky kategorie G1 a G2 byla počítána jako Cohenovo d (viz [8]) a klasifikována jako zanedbatelná (Cohenovo $d < 0.2$), malá (Cohenovo $d < 0.5$), střední (Cohenovo $d < 0.8$) a velká (Cohenovo $d \geq 0.8$).

	Pouze výklad látky	Výklad základů + hraní hry	Celkem
Dívky	30	29	59
Chlapci	19	22	41
Celkem	49	51	100

Tabulka 1: Počty studentů v porovnávaných skupinách.

4.3 Výsledky evaluace

Protože jsou výsledky evaluace již obšírně popsány v [6], zde pouze shrnu výsledky analýzy rozptylu odpovědí na otázky kategorie G1 a G2 u obou dotazníků.

Analýza rozptylu otázek G1 (zaměřených na obecná fakta) neprokázala žádný rozdíl mezi skupinou s výkladovou přednáškou a skupinou, která hrála hru ($F[1,96] = 1.4060$, $p = 0.2387$). Stejně tak nebyl prokázán rozdíl mezi výsledky studentů různého pohlaví ($F[1,96] = 1.3190$, $p = 0.2536$). Rozdíl byl prokázán jen u výsledků odpovědí z bezprostředního dotazníku a dotazníku vyplňovaném po měsíci ($F[1,96]$ ¼ 111.5130, $p < 0.0001$).

Žádná z iterací nebyla signifikantní (pro všechna p platilo $p > 0.6542$). Tím pádem obě skupiny studentů dopadly v případě otázek kategorie G1 stejně, přestože analýza „velikosti změny“ (effect size) ukazovala v případě bezprostředních dotazníků malý příklon ke skupině, která podstoupila pouze výklad, viz tabulka 2.

Studenti z obou skupin dosahovali lepších výsledků bezprostředně po přednášce než po měsíci od ní, viz graf na obrázku 16 a tabulka 4.

Analýza rozptylu u otázek G2 (procvičovaných hrou) prokázala signifikantní rozdíl mezi oběma skupinami ($F[1,96] = 6.4206$, $p = 0.0129$), ale žádný rozdíl mezi pohlavími ($F[1,96] = 1.0305$, $p = 0.3126$) nebo mezi bezprostředním a zpožděným dotazníkem ($F[1,96] = 1.0903$, $p = 0.2990$).

Opět nebyla žádná z iterací signifikantní (pro všechna p platilo p > 0.1428). Hlavním poznatkem tak bylo, že skupina, která hrála hru, dopadla při otázkách kategorie G2 lépe u obou dotazníků.

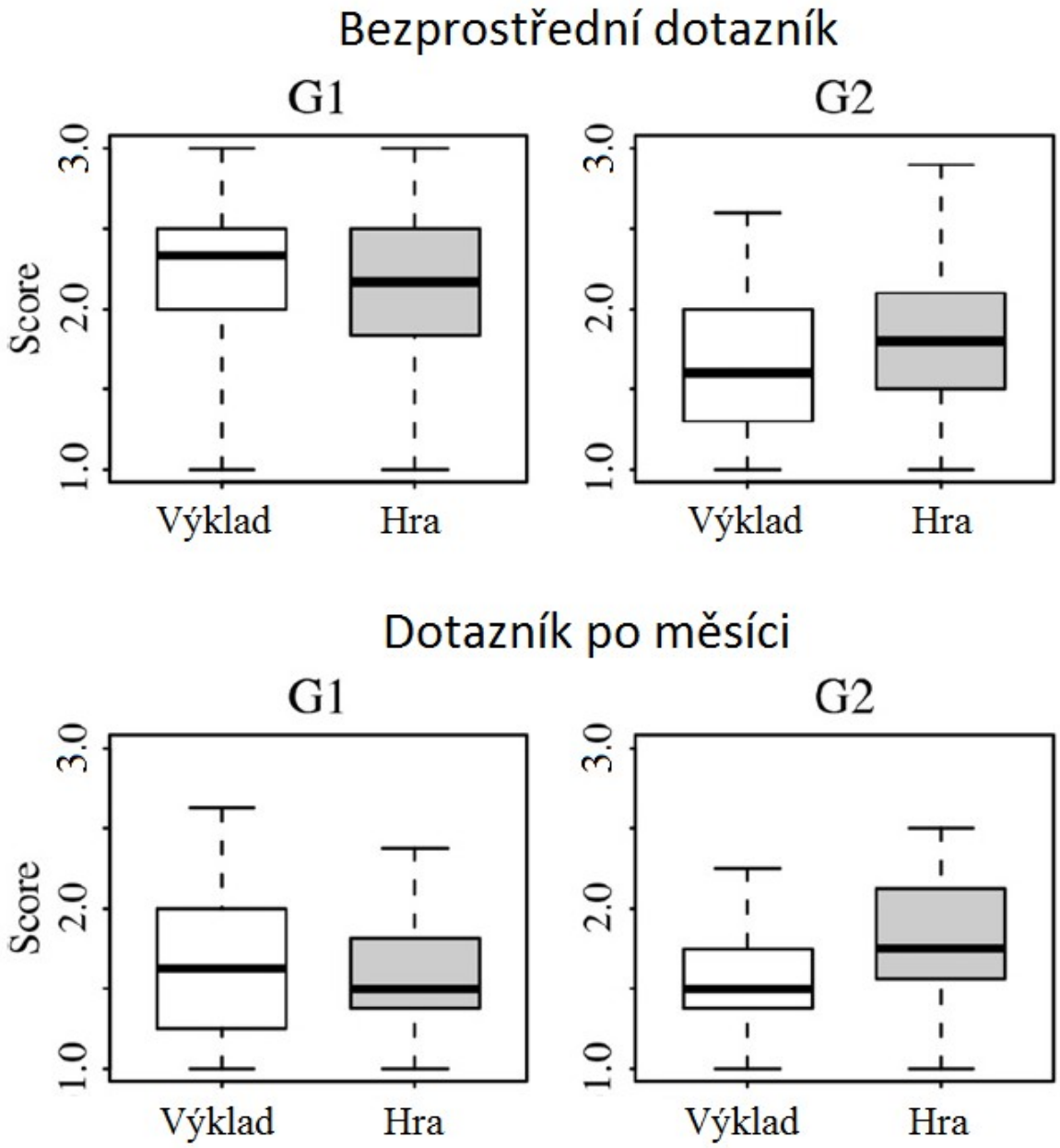
Analýza „velikosti změny“ (effect size) navíc ukázala, že rozdíl mezi skupinami byl větší v případě dotazníků vyplňovaných po měsíci od přednášky, viz tabulka 3.

		G1				G2		
		Chlapci	Dívky	Všichni		Chlapci	Dívky	Všichni
Pouze výklad látky	E	2.27	2.16	2.23		1.62	1.7	1.65
	SD	0.52	0.55	0.53		0.48	0.48	0.48
Výklad + hraní hry	E	2.15	2.05	2.11		1.86	1.7	1.79
	SD	0.51	0.57	0.53		0.43	0.46	0.45
Velikost změny (effect size)	Cohenovo d	-0.24	-0.19	-0.23		0.54	-0.01	0.3
		malý	zanedbatelný	malý		střední	zanedbatelný	malý

Tabulka 2: Střední hodnoty (E), směrodatné odchylky (SD) a „velikosti změny“ (effect size) v podobě Cohenova d [8] u otázek G1 a G2 bezprostředního dotazníku. Tabulka je převzata z [6].

		G1				G2		
		Chlapci	Dívky	Všichni		Chlapci	Dívky	Všichni
Pouze výklad látky	E	1.67	1.57	1.63		1.61	1.52	1.57
	SD	0.53	0.36	0.47		0.29	0.27	0.28
Výklad + hraní hry	E	1.59	1.52	1.56		1.83	1.73	1.78
	SD	0.41	0.36	0.39		0.37	0.32	0.35
Velikost změny (effect size)	Cohenovo d	-0.16	-0.16	-0.17		0.68	0.71	0.67
		zanedbatelný	zanedbatelný	zanedbatelný		střední	střední	střední

Tabulka 3: Střední hodnoty (E), směrodatné odchylky (SD) a „velikosti změny“ (effect size) v podobě Cohenova d [8] u otázek G1 a G2 dotazníku vyplňovaného po měsíci od přednášky. Tabulka je převzata z [6].



Obr. 16: Známky u odpovědí na otázky kategorie G1 a G2 pro oba dotazníky a obě skupiny studentů. Zobrazeny jsou mediány a první a třetí kvartily. Obrázek je přejatý z [6].

4.4 Diskuze výsledků

Porovnání dotazníků skupiny studentů, která podstoupila klasickou výkladovou přednášku, a skupiny, která měla výklad doplněn o hraní hry Chovatel zvířat, přináší několik závěrů.

Hlavním z nich je to, že studenti, kteří hráli hru (bez závislosti na pohlaví), si lépe dlouhodobě zapamatovali vědomosti, které hra procvičuje (viz obrázek 16 a tabulka 3). Skupina, která hrála hru, po měsíci na otázky odpovídala se signifikantně lepším skóre. Přitom ihned po přednášce dosahovaly obě skupiny v otázkách zaměřených na vědomosti posilované hrou výsledků, které nebyly signifikantně rozlišitelné.

Dalším zajímavým výsledkem evaluace je, že obě skupiny dosahovaly nerozlišitelných výsledků i v otázkách zaměřených na obecná fakta, které hra přímo neposilovala (výklad v první hodině praktik). Studenti, kteří druhou hodinu hráli hru, sice u těchto otázek dosahovali menšího skóre, ale rozdíl mezi skupinami nebyl signifikantní.

V rámci evaluace byl také hodnocen názor studentů na obsah přednášky. Zde je poměrně překvapivé, že neexistuje signifikantní rozdíl mezi jejich hodnocením výkladové přednášky a přednášky s hrou, pokud jde o zábavu ani výukovou hodnotu. Názory na výukovou hodnotu jsou nicméně trochu (ale ne signifikantně) v neprospěch přednášky s hrou.

Detailnější rozbor všech výsledků, které evaluace přinesla (např. závislost znalostí na pohlaví studenta atp.), lze nalézt v [6].

		G1				G2		
		Chlapci	Dívky	Všichni		Chlapci	Dívky	Všichni
Pouze výklad látky	E	-0.61	-0.59	-0.6		-0.01	-0.18	-0.08
	SD	0.51	0.47	0.49		0.41	0.44	0.43
Velikost změny (effect size)	Cohenovo d	-1.17	-1.3	-1.21		-0.03	-0.48	-0.2
		velký	velký	velký		zanedbatelný	malý	malý
Výklad + hraní hry	E	-0.56	-0.54	-0.55		-0.03	0.03	-0.01
	SD	0.57	0.61	0.58		0.36	0.35	0.35
Velikost změny (effect size)	Cohenovo d	-1.23	-1.15	-1.19		-0.09	0.08	-0.01
		velký	velký	velký		zanedbatelný	zanedbatelný	zanedbatelný

Tabulka 4: Střední hodnoty (E), směrodatné odchylky (SD) a „velikosti změny“ (effect size) v podobě Cohenova d [8] u rozdílu skóre z otázek G1 a G2 obou dotazníků. Tabulka je převzata z [6].

Závěr

V této práci byl popsán výpočetní model, který simuluje chování různých druhů zvířat při jednoduchých cvicích s trenérem. Model je součástí výukové aplikace Chovatel zvířat, kde se pomocí něj simuluje chování psa, lemura a papouška. Simulovaná zvířata se dokážou učit několika obvyklými trénovacími technikami a podle profesionálního trenéra zvířat jsou jejich reakce srovnatelné s reakcemi zvířat skutečných.

Model je snadno parametrizovatelný a jeho fungování natolik plausibilní, že se dá jeho další chování úspěšně odhadnout jen na základě výpisu několika číselných hodnot (pravděpodobností vykonání akcí).

Dále byly diskutovány i důvody, proč bylo nutné navrhnout originální model, a proč nebyl použit nějaký ze známých postupů strojového učení. Přitom se jako hlavní problém zmíněných zpětnovazebních modelů ukázala variabilita, která je potřeba k simulování různých druhů zvířat a schopnost generalizace nutná k učení „tvarováním“. Dalším problémem známějších modelů zpětnovazebního učení byla jejich přílišná komplexnost. Pokud je takový model používán v programu, kde uživatel nemá možnost zobrazovat jeho interní stav (jako je Chovatel zvířat), je velice těžké předvídat jeho další chování. Při výuce na středních školách se ale předpokládá, že na studenty dohlíží učitel, který podstoupil pouze krátké školení o užívání programu a neví přesně, jak model funguje. Z tohoto hlediska je navržený model vhodnější než zmiňované modely.

Ovšem model je navržen speciálně pro simulace jednoduchých tréninků se zvířaty v aplikaci Chovatel zvířat a v případě složitějšího prostředí, se kterým by virtuální zvíře mohlo interagovat, nebo v případě větší nutnosti řetězit akce, by mnou navržený model byl nevhodný.

Účel aplikace Chovatel zvířat, využívající model, je utvrzení znalostí nabytých výkladem ve škole. Výsledky evaluace vlivu programu na znalosti studentů ukázaly, že studenti hrající hru si z hodiny odnesou zhruba stejné penzum vědomostí, jako studenti, kteří pouze poslouchají výklad na stejné téma. Zajímavější je ovšem to, že studenti si hraním nabyté vědomosti dlouhodobě uchovávají lépe než v případě, že podstoupili pouze standardní výklad.

Program se kromě středních škol dá díky variabilitě tréninků rovněž využít i na školách základních. Díky realističnosti učení zvířat a designu tréninků ale může program sloužit dokonce i jako trenažér pro trenéry zvířat a podobné profese.

Literatura

[1] Aitkin, A. L. (2004). "Playing at Reality: Exploring the potential of the digital game as a medium for science communication", PhD thesis, Australian National University.

[2] Alloway, T.; Wilson, G.; Graham, J. (2005). "Sniffy The Virtual Rat", Wadsworth/Thomson Learning, Belmont.

[3] Anděl, J. (2007). "Statistické metody" (4th ed.). Matfyz Press.

[4] Bellman, R. (1957). "Markovian Decision Process." Indiana University Mathematical Journal 6 No. 4, 679–684

[5] Blackham, D. (1974). "Operant Conditioning: An Experimental Analysis of Behavior", Methuen & Ltd. London.

[6] Brom, C.; Preuss, M.; Klement, D. (2011) "Are Educational Computer Micro-Games Engaging And Effective For Knowledge Acquisition At High-Schools? A Quasi-Experimental Study. " In: Computers & Education 57

[7] Casella, G.; Berger, R.L. (2001). "Statistical Inference". Duxbury Press.

[8] Cohen, J (1988). "Statistical power analysis for the behavioral sciences" (2nd ed.). Lawrence Erlbaum Associates.

[9] Dignum, F.; Bradshaw, J.; Silverman, B.G.; Doesburg, W. (2009). "Agents for Games and Simulations", Springer.

[10] Egenfeldt-Nielsen, S. (2005) "Beyond edutainment: Exploring the educational potential of computer games", PhD thesis, University of Copenhagen.

[11] Grand, S.; Cliff, D.; Malhotra, A. (1997). "Creatures: Artificial Life Autonomous Software Agents for Home Entertainment". In: The First International Conference on Autonomous Agents (Agents '97), pp. 22-29.

[12] Hays, R. T. (2005). "The effectiveness of instructional games: A literature review and discussion." technical report 2005-004. Orlando: Naval Air Warfare Center Training Systems Division.

[13] La Camera G.; Richmond B.J. (2008) "Modeling the Violation of Reward Maximization and Invariance in Reinforcement Schedules". PLoS Comput Biol 4(8): e1000131.

[14] Powell, R.A.; Symbaluk, D.G.; Lynne Honey, P. (2008) "Introduction to Learning and Behavior." Cengage Learning.

[15] Preuss, M.; Brom, C. (2010) "Orbis Pictus Bestialis: Virtuální učení zvířat." In: Kognice a umělý život X, pp. 295-302

[16] Sutton, Richard S. (1988). "Learning to predict by the method of temporal differences". Machine Learning (Springer) 3: 9–44.

[17] Sutton, R.S.; Barto, A.G. (1998). "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA.

[18] Sweatt, J.D. (2003). "Mechanisms of Memory", Academic Press.

[19] Thomas, R., & Hooper, E. (1991). "Simulations: an opportunity we are missing. Journal of Research on Computing in Education", 23(4), 497–513.

[20] Tokic, Michel; Günther Palm (2011). "Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax". KI 2011: Advances in Artificial Intelligence. Lecture Notes in Computer Science. 7006. Springer Berlin / Heidelberg. pp. 335-346.

[21] Veselovský, Z. (2005). "Etologie. Biologie chování zvířat", Academia, Praha.

[22] Watkins, C.J.C.H. (1989). "Learning from Delayed Rewards", PhD thesis, Cambridge University, Cambridge.

Seznam konstant

Vyjmenované konstanty se při běhu programu Chovatel zvířat nedají měnit.

α_p	-	míra změny pravděpodobnosti akce ve stavu
α_r	-	míra změny očekávané odměny akce ve stavu
β_n	-	parametr regulující kladné změny klasického spárování stimulů
β_p	-	parametr regulující záporné změny klasického spárování stimulů
γ	-	míra vlivu budoucího stavu na odhad odměny
c_{iter}	-	počet iterací při propagaci okamžité odměny
c_{rew}	-	faktor snížení odměny při její propagaci
R_{exp_min}	-	minimální hodnota očekávané odměny akce ve stavu
$prob_{lim}$	-	minimální suma pravděpodobností všech akcí kromě klidové ve stavu
$Prob_{min}$	-	minimální pravděpodobnost vykonání akce ve stavu