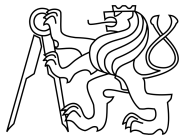
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Neuronová síť pro hraní Pokeru
Student:	Marek Hanuš
Vedoucí:	Ing. Zdeněk Buk, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Implementujte simulátor karetní hry No Limit Hold'em Poker [1]. Dále implementujte a pomocí vytvořeného simulátoru otestujte vybrané typy neuronových sítí pro hraní Pokeru. Rozhodování hráčů ve hře bude tedy řízeno neuronovou sítí a hlavním kritériem pro porovnání úspěšností jednotlivých sítí bude jejich inteligence.

Seznam odborné literatury

[1] Pravidla karetní hry Hold'em Poker: https://en.wikipedia.org/wiki/Texas_hold_%27em

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 6. března 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Neuronové sítě pro hraní pokeru

Marek Hanuš

Vedoucí práce: Ing. Zdeněk Buk, Ph.D.

15. května 2017

Poděkování

Rád bych poděkoval panu Ing. Zdeňkovi Bukovi Ph.D. za trpělivost a odbornou pomoc při psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Hanuš, Marek. *Neuronové sítě pro hraní pokeru.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se zabývá vytvořením umělé inteligence pro karetní hru poker s využitím umělých neuronových sítí. Na rozdíl od ostatních projektů, které se věnují tématice umělé inteligence ve hrách, je v této práci neuronová síť použita jako jediné rozhodovací kritérium a její výstupy přímo odpovídají tahům hráče. Hráči se zlepšují za pomoci metody posilovaného učení připomínající styl, kterým se učí lidé, tedy ze svých zkušeností. Výsledky testování ukázaly, že takto vytvořená neuronová síť hraje na podobné úrovni jako rekreační hráč.

Klíčová slova Poker, Neuronové sítě, Umělá inteligence, Posilované učení, Strojové učení, Q-učení

Abstract

This bachelor thesis deals with creating artificial inteligence for poker using artifical neural networks. Unlike other projects, that focus on artificial inteligence for games, neural network is not just one of many parts in big algorithm, but it decides which moves to take on its own. Players use reinforcement learning to improve their play. Tests show that neural network trained this way plays on similar level as recreation player.

Keywords Poker, Artificial Neural Networks, Artificial intelligence, Reinforcement learning, Machine learning, Q-learning

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Umělá inteligence zažívá v poslední době velmi rychlý rozvoj a patří mezi nejdiskutovanější témata informatiky. Je to zapříčiněno hlavně vývojem v oblasti umělých neuronových sítí. Neuronové sítě nám umožňují řešit problémy, které zatím nebylo možné překonat standardními způsoby programování. Jejich hlavní výhodou je schopnost učit se z předložených dat a hledat v nich obecné vzory, které jim umožňují porozumět i situacím dosud neznámým.

Využíváme je například při převodu řeči na text[1], k rozpoznávání obrazu nebo i u autonomního řízení aut [2] [3]. V těchto oblastech jsou schopné dosahovat podobných a občas i o mnoho lepších výsledků než lidé. Dalším odvětvím, kde neuronové sítě našly své uplatnění, je teorie her. Podařilo se například vytvořit počítačový program, který hraje čínskou deskovou hru Go lépe než nejlepší hráč světa [4] [5], nebo dosáhnout úrovně lidských hráčů v sérii her Atari za pomoci vstupů identických člověku, tedy obrazu z hry [6].

Tato práce se pokusí neuronové sítě využít k vytvoření umělé inteligence pro hru poker. V minulosti proběhlo již několik podobných experimentů. Největší úspěch na tomto poli sklidil pravděpodobně projekt DeepStack [7], který dokázal porazit profesionální hráče pokeru ve variantě No-Limit Heads Up, neboli jeden proti jednomu.

Ve všech zmíněných příkladech byla neuronová síť využita jako podpůrný prostředek dalšího algoritmu. Ve hře Go sloužila jako model předpovídající tahy profesionálních hráčů, které se naučila na reálně odehraných partiích. Poté byl pomocí stromové vyhledávací metody Monte Carlo zjištěn správný tah proti takto namodelovanému soupeři [8].

V této práci je zvolen jiný přístup. Neuronová síť zde není jedním z mnoha stavebních prvků potřebných k určení správného tahu, ale rozhoduje o logice hráče sama o sobě. Je to jediná věc, kterou počítačem řízený hráč využívá a výstupy neuronové sítě přímo odpovídají zvoleným tahům hráče.

Rozdílné je v tomto programu také využívání reálně odehraných partií pro trénink virtuálního hráče. V tomto případě se neuronová síť učí sama při simulovaném hraní. Začíná hrát zcela náhodně a postupně zjišťuje, které tahy jsou

pro ni výhodné a které nikoli. Tímto přístupem je zamezeno potenciálnímu naučení se špatných herních návyků a je možné otestovat, zda se počítačový hráč bez externích zásahů naučí hrát podobně jako lidé.

Vygenerováno bylo několik neuronových sítí a ta nejlepší byla otestována krátkou partií proti reálnému hráči pokeru, který poté pomohl zhodnotit přibližnou kvalitu takto vytvořené umělé inteligence.

V první kapitole je představen poker a jeho pravidla. Dále je také ukázáno, proč představuje vytvoření umělé inteligence pro hraní pokeru takový problém a důvody zabraňující jeho vyřešení.

Druhá kapitola seznámí čtenáře s neuronovými sítěmi a principy, na kterých jsou založeny.

Ve třetí budou následovat potřebná omezení hry, aby bylo možné neuronové sítě využít. Zde je také nastíněn algoritmus, pomocí kterého jsou generováni stále lepší hráči.

Čtvrtá kapitola se zabývá Q-učením. Algoritmus je postupně vylepšen několika způsoby, které stabilizují celý proces a zamezují předčasné konvergenci. Zakomponováno bylo využití předchozích zkušeností, cílové sítě nebo dvojitého učení.

Pátá a šestá kapitola se veňují vstupům neuronové sítě. Představí se všechny informace, které je možné ve hře sledovat a poté je zjištěno jejich nejlepší kódování. Následně je zvětšován počet informací, které má neuronová síť k dispozici.

Zbytek práce se zabývá detaily implementace a výsledky různých experimentů.

Kapitola 1

Cíl práce

Hlavním cílem této práce je vytvořit umělou inteligenci pro hru No-Limit Holdem Poker, kde proti sobě hraje najednou 6 hráčů. K tomu je potřeba splnit několik podúkolů:

- Implementovat pokerový simulátor, který poslouží jako prostředí, kde budou neuronové sítě hrát a zlepšovat se. Bude také prostředkem pro zjištění kvality jednotlivých umělých neuronových sítí.
- Propojit neuronové sítě s pokerovým simulátorem. Vytvořit způsob získávání informací ze hry a následně jejich převedení do formátu využitelného sítěmi.
- Naprogramovat posilované učení za pomocí Q-učení s využitím cílové sítě, dvojitého učení či přehrávání zkušeností.
- Vytvořit grafické prostředí pro aplikaci, kde bude pro uživatele možné poměřit síly s nejlepšími neuronovými sítěmi.

Kapitola 2

Analýza a návrh

2.1 Poker

Poker patří do rodiny karetních her, které kombinují aspekty hazardu, strategie a štěstí. Všechny pokerové varianty zahrnují sázení jako hlavní část hry a rozhodují o vítězi handy (jedna celá hra od rozdání karet po zvolení vítěze) podle kombinace hráčových karet, ze kterých alespoň některé zůstávají skryté soupeřům až do konce hry. Jednotlivé varianty se od sebe liší počtem karet rozdaných, společných a skrytých, nebo strukturou sázení. [9]

Pokerových variant je několik. Mezi ty nejznámější patří [10]:

- Texas Hold'em
- Omaha High
- Omaha Hi Lo
- Seven Card Stud
- Razz
- Five Card Draw

Dále můžeme poker rozdělit podle povolených pravidel sázení. Zde máme celkem tři varianty [11]:

- **Limit:** Velikosti sázek jsou předem dané s ohledem na fázi handy a hráči je nemohou měnit.
- **Pot limit:** Maximální velikost sázky je omezena počtem žetonů v banku. Hráči nemohou vsadit více, než je zde žetonů.
- **No limit:** Jediné omezení je zde počet žetonů, které hráč vlastní. Není možné vsadit více než má.

Tato práce se věnuje pouze variantě No Limit Texas Hold'em, kde proti sobě hraje 6 hráčů.

2.1.1 Pravidla pokerové varianty Texas Hold'em

Texas Hold'em je hra pro 2-10 hráčů, kde každý účastník na začátku obdrží dvě vlastní karty a postupně je na stůl vyloženo pět karet společných. V této variantě existují dvě možnosti jak ukončit hru a tím vyhrát žetony:

- Pokud zůstane ve hře pouze jeden hráč, automaticky získává celý bank.
- Po odehrání posledního sázkového kola zbylí hráči porovnají své karty a ten s nejlepší kombinací vyhrává.

Úkolem hráče je tedy buď získat co nejlepší možnou kombinaci karet, nebo přesvědčit soupeře, že má lepší karty než oni a tím je donutit ty své zahodit.

Kombinací rozlišujeme celkem 10 a jsou seřazeny vzestupně od nejlepší po nejhorší [12]:

1. **Královská postupka:** Nejlepší kombinace. Jedná se o postupku v barvě, která má hodnoty karet 10-J-Q-K-A
2. **Postupka v barvě:** Pět po sobě jdoucích karet stejné barvy
3. **Čtveřice:** Čtyři karty stejné hodnoty
4. **Full house:** Trojice a dvojice (viz bod 7 a 9)
5. **Barva:** Pět karet stejné barvy
6. **Postupka:** Pět po sobě jdoucích karet
7. **Trojice:** Tři karty stejné hodnoty
8. **Dvě dvojice:** Dvě dvojice
9. **Dvojice:** Dvě karty stejné hodnoty
10. **Vysoká karta:** Žádná z výše uvedených. Hráč použije kartu s nejvyšší hodnotou

Každá handa může být rozdělena na několik fází.

- Před začátkem handy
- Před flopem
- Flop
- Turn
- River

2.1.1.1 Před začátkem hry

Na začátku celé partie je v pokeru náhodně vylosován tzv. dealer. Hráč na této pozici se vyjadřuje v sázkách jako poslední. Tato pozice se po odehrání každé handy posune o jedno místo po směru hodinových ručiček. [13]

"Hráč nalevo od dealera vloží do hry tzv. small blind, malý blind neboli malou povinnou sázku naslepo. Druhý hráč od dealera pak vloží dvojnásobek této sázky, big blind, velký blind neboli velkou povinnou sázku naslepo. Big blind také hodnotou odpovídá minimální sázce, kterou lze ve hře vsadit. Povinné sázky (blindy) jsou prvními žetony vloženými do banku a jejich hlavním cílem je podpořit akci v prvním kole sázek tak, aby se "o něco hrálo". [13]

2.1.1.2 Před flopem

Poté, co hráči na blindech vloží do hry povinné sázky, je na řadě hráč sedící nalevo od pozice velkého blindu. Hráči můžou buď dorovnat sázku ve výši velkého blindu, navýšit, nebo mohou své karty zahodit a do hry se vůbec nezapojit. Přítomnost povinných sázek zaručuje, že se hráči nemohou účastnit hry bez investice žetonů (kromě hráče na velkém blindu, který, pokud nikdo nevsadil, má svoji účast ve hře "předplacenou"velkou povinnou sázkou a když na něj přijde řada, může počkat a nedělat nic). [13]

2.1.1.3 Flop

Po uzavření prvního kola sázení se na stůl vyloží 3 společné karty. Hráči mají možnost porovnat své karty s těmi společnými a zhodnotit své šance na výhru. První na řadě v novém kole sázek je první hráč sedící nalevo od dealera, který zůstal ve hře. Sázení se řídí stejnými pravidly jako v předchozím kole. Dokud nikdo nevsadí, je možné zůstat ve hře bez vsazení. Po sázce je možno pouze dorovnat, navýšit sázku, nebo složit karty. [13]

2.1.1.4 Turn

Po ukončení sázek na flopu nastává další sázkové kolo - Turn. Vyložena je jedna další společná karta. Sázky se řídí stejnými pravidly jako v kolech předchozích. [13]

2.1.1.5 River

Na stůl se vyloží poslední, pátá společná karta. Následuje znovu kolo sázek a po jeho ukončení si zbylí hráči ve hře ukážou a porovnají své karty. Všechny žetony v společném banku vyhrává hráč s nejlepší karetní kombinací. [13]

2.1.2 Překážky bránící vyřešení pokeru

No limit holdem pro 6 hráčů je hra, která zatím nebyla počítačem vyřešena. Je to způsobeno jejími vlastnostmi, které ji ale také činí tak zábavnou a oblíbenou. Největším problémem je pravděpodobně počet variant, které mohou nastat.

Annual Computer Poker Competition [14] je soutěž pořádaná každý rok. Jejím cílem je vytvořit počítačový program, který hraje poker No Limit Hold'em co možná nejlépe. Podle pravidel má každý hráč k dispozici 200 big blindů. Za takových podmínek je velikost prostoru všech možností průběhu hry $6,3114 \times 10^{164}$. Vyřešení této hry pomocí standardních metod numerické minimalizace by vyžadovalo 1.094×10^{138} yottabytů paměti [15]. Tak obrovské množství paměti určitě nebudeme mít k dispozici v blízké budoucnosti a nejspíše ani v té vzdálené.

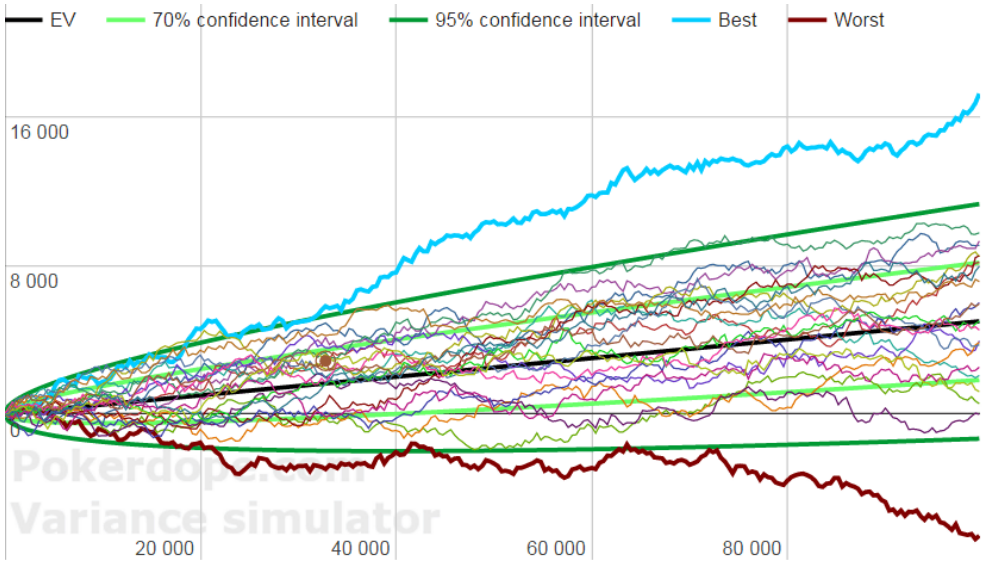
Dalším problémem je neúplná informace o stavu hry. Jelikož má každý hráč dvě karty, které zná pouze on sám, nemůžeme nikdy s jistotou vědět, jestli jsme na tom lépe či hůře než naši soupeři. Neustále musíme odhadovat soupeřovy karty a jedinou infromací nám je právě náš odhad. Co když je náš odhad špatný? Každý hráč také může hrát s odlišnou taktikou. Strategie vyhrávající nad hráčem A už nemusí fungovat na hráče B.

V neposlední řadě nám vše komplikuje velká variace v pokeru. Je úplně běžné, že vyhrávající hráč má i několik desítek tisíců hand dlouhé série, kdy má smůlu a prohrává. Tato vlastnost zvětšuje počet her, které musíme odehrát, abychom získali věrohodná data, a výrazně prodlužuje čas potřebný k simulaci. Graf znázorňující tuto zkutečnost najdeme níže 2.1.

2.2 Umělá neuronová síť

Umělá neuronová síť je jeden z výpočetních modelů používaných v umělé inteligenci. Jejím vzorem je chování odpovídající přírodním strukturám. Skládá se z umělých neuronů, jejichž předobrazem je biologický neuron. Neurony jsou propojeny, navzájem si předávají signály a transformují je pomocí určitých přenosových funkcí. Každý neuron má libovolný počet vstupů, ale pouze jeden výstup. Neuronové sítě se používají mimo jiné pro rozpoznávání a kompresi obrazů nebo zvuků, předvídání vývoje časových řad (např. burzovních indexů), někdy dokonce i k filtrování spamu. V lékařství také slouží k prohlubování znalostí o fungování nervových soustav živých organismů. [17]

Matematický model neuronových sítí reprezentuje funkci, která transformuje libovolně velkou množinu vstupů na množinu výstupů. Pomocí neuronové sítě s jednou skrytou vrstvou s konečným počtem neuronů můžeme reprezentovat jakoukoli funkci, která spojitě mapuje množinu vstupů na výstupy. Problém nalezení této neuronové sítě už je ale velmi složitý a zatím není vyřešen. [18].



Obrázek 2.1: Graf zobrazuje hráčem vyhrané žetony v závislosti na počtu odehraných hand. Černá barva znázorňuje předpokládaný zisk hráče - 5bb/100 (5 velkých povinných sázek na 100 hand). Z grafu můžeme vyčíst, že po odehrání 100 000 hand bude profit hráče v rozmezí -1.32 až 11.32bb/100 s 95% pravděpodobností. Mezi tímto intervalem a reálnou hodnotou je až dvojnásobný rozdíl a proto je pro přesné zjištění zisku potřeba odsimulovat obrovské množství hand.[16]

2.2.1 Neuron

Neuron (viz obrázek 2.2) je základní stavební jednotkou umělých neuronových sítí. Každý má x vstupů a jeden výstup. Výstup neuronu je vypočten pomocí následujícího vzorce:

$$y = f(\sum_{n=1}(i_n * w_n + b_m))$$

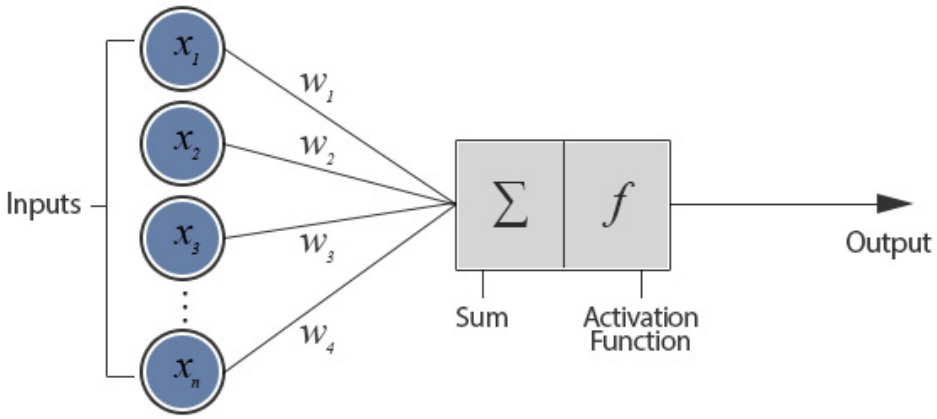
kde:

$\boldsymbol{i_n}$ je í-tý vstup neuronu

$\boldsymbol{w_n}$ je váha pro í-tý vstup

$\boldsymbol{b_n}$ je posunutí í-tého vstupu

$\boldsymbol{f}$ je aktivační funkce neuronu



Obrázek 2.2: Ukázka umělého neuronu [19]

2.2.2 Aktivační funkce

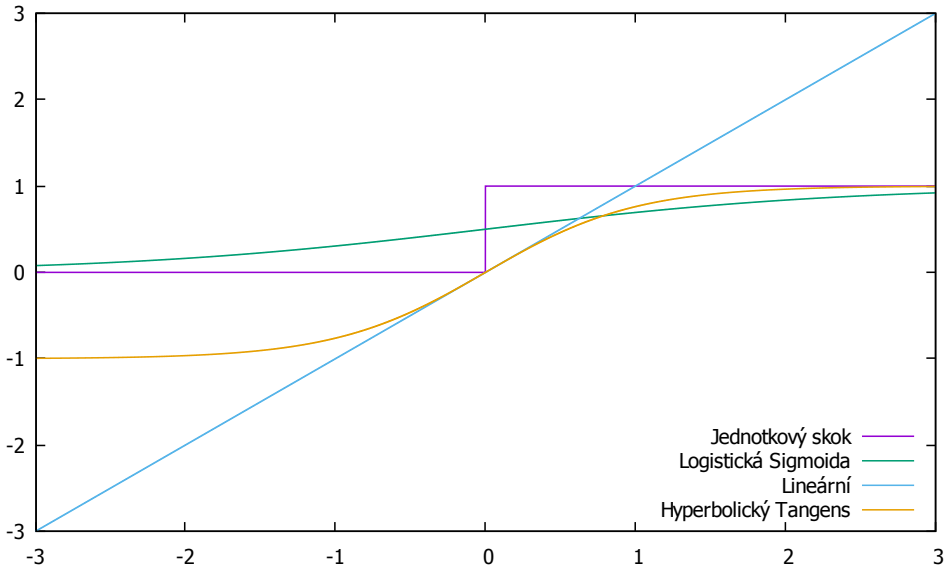
Neuron bez aktivační funkce si lze představit jako lineární separátor, kde váhy jednotlivých vstupů ovlivňují naklonění přímky a posunutí mění bod průniku s osou x - viz obrázek 2.4.

V reálném světě ale většina funkcí lineárně separovatelná není, proto byla k neuronu přidána aktivační funkce. Ta zajišťuje nelineární transformaci a tím umožňuje třídit data, která lineárně klasifikovatelná nejsou. Mezi nejpoužívanější aktivační funkce patří: Jednotkový skok, Logistická Sigmoida, Hyperbolický Tangens, Lineární nebo Softmax [20]. Jejich průběh můžeme vidět výše 2.3

2.2.3 Dopředné neuronové sítě

Dopředná neuronová síť je velké množství neuronů uspořádaných do jednotlivých vrstev, které jsou vzájemně propojeny viz obrázek 2.5. Vstupem každého neuronu jsou všechny výstupy neuronů z vrstvy předchozí. Neurony v jedné vrstvě sdílí stejnou aktivační funkci. Vrstvy se dělí do tří typů: [18]

- **Vstupní vrstva** - První vrstva neuronové sítě. Počet neuronů odpovídá počtu vstupů sítě. Neurony zde většinou nemají žádnou aktivační funkci a slouží pouze k načtení dat.
- **Skryté vrstvy** - Dále zde můžeme najít 1-N skrytých vrstev. V těchto vrstvách dochází k samotnému výpočtu.
- **Výstupní vrstva** - Poslední vrsta neuronové sítě. Její aktivační funkce bývá zpravidla Softmax, která konečný výsledek transformuje na množinu, jejíž součet prvků je roven 1. [22]



Obrázek 2.3: Průběh nejpoužívanějších aktivačních funkcí. Osa x je rovna vnitřnímu potenciálu neuronu a y jeho výstupu.

2.2.4 Zpětná propagace

Aby se mohly neuronové sítě něco naučit, musí existovat způsob, jak vyjádřit velikost chyby neuronové sítě. K tomuto slouží tzv. chybové funkce. Ideální chybová funkce se může lišit problém od problému, ale nejčastěji používaná je kvadratická s předpisem:

$$C = \frac{1}{2n} \sum_{x} \|y(x) - a^L(x)\|^2, [23] \tag{2.1}$$

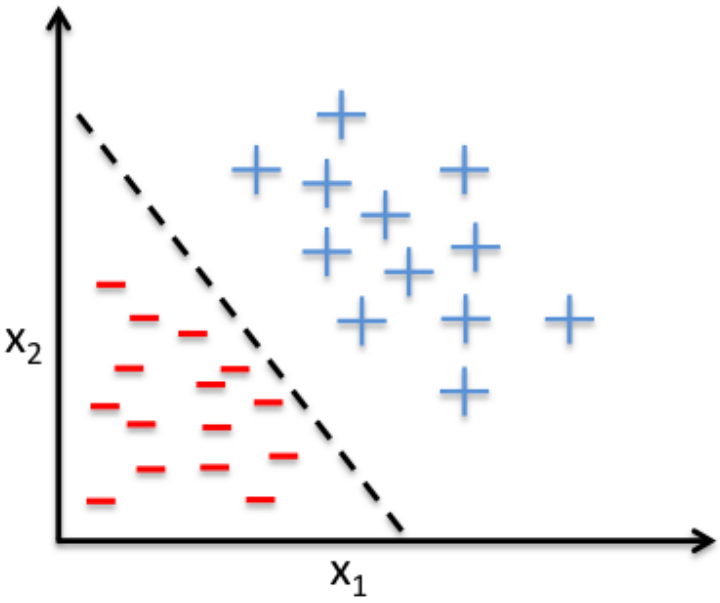
kde:

$\boldsymbol{n}$ je počet trénovacích vzorků

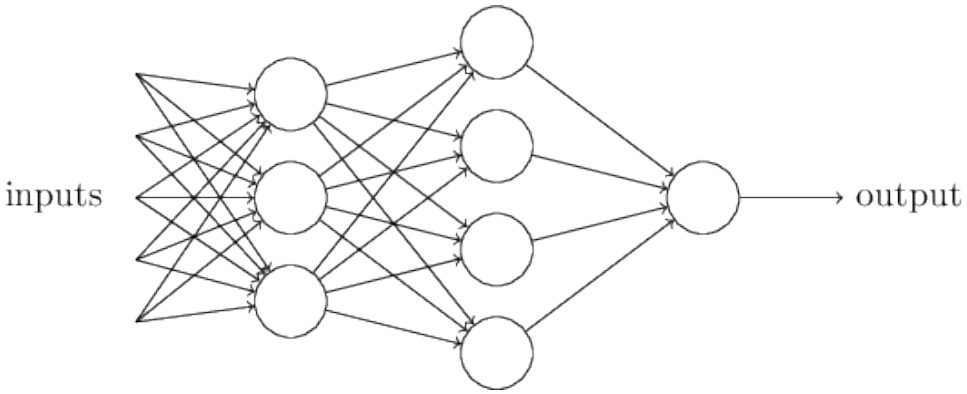
$\boldsymbol{y(x)}$ je požadovaný výstup sítě pro x-tý trénovací vzorek

$\boldsymbol{a^L(x)}$ je výstup L-té vrstvy pro x-tý trénovací vzorek

Samotné učení je poté minimalizace této chyby pomocí gradientního sestupu. Algoritmus zde popsán nebude, jelikož je to téma tak rozsáhlé, že by o něm mohla být napsána další samostatná práce. Informace o algoritmu jsou např. na [23] nebo [24].



Obrázek 2.4: Neuron jako lineární separátor [21]. Graf znázorňuje body, které je možné rozdělit do dvou skupin - modré a červené. V tomto případě je možné body správně klasifikovat pomocí lineární funkce, která je reprezentována čárkovanou přímkou.



Obrázek 2.5: Neuronová síť [23]

2.3 Vytvoření umělé inteligence

Tato práce se pokouší vytvořit umělou inteligenci na hraní hry Poker, který patří mezi hry, u nichž není možné objektivně zhodnotit kvalitu hráčovy hry. Hráči nedostávají žádné body za to, jak daleko se ve hře dostanou. Veškeré hodnocení je subjektivní a platné pouze v momentálním kontextu. Zisk hráče není tranzitivní. Nemůžeme se tedy spolehnout, že pokud hráč A porazí hráče B a hráč B porazí hráče C, tak hráč A vyhraje proti hráči C. Existuje mnoho strategií, jak hrát poker, a každá disponuje klady a zápory.

Při učení se hráč snaží osvojit si strategii, pomocí které porazí svého soupeře. Pokud se má hráč naučit poker na dobré úrovni, musí také trénovat proti hráčům, kteří v této hře vynikají. Stejně jako ve sportu zde platí pravidlo: "Pokud chceš být nejlepší, trénuj s nejlepšími". Narážíme zde na nepříjemnou rekurzi, protože k vytvoření kvalitního hráče potřebujeme kvalitního hráče. Kde sehnat ten prvotní vzor?

K vyřešení tohoto problému byl navržen algoritmus 1, který žádného učitele nepotřebuje a vše si vygeneruje sám.

Algorithm 1 Algoritmus nalezení dobře hrající umělé inteligence

```
vytvoř množinu náhodně hrajících hráčů
repeat
    vyber náhodného hráče z množiny
    nauč ho hrát co nejlépe proti zbylým soupeřům
    pokud má nově naučený hráč lepší výsledky, nahraď jím toho starého
until konec
```

Tento algoritmus je založen na dvou předpokladech:

- **Existence Nashovy rovnováhy v pokeru**: Nashova rovnováha je koncept pocházející z teorie her. Hráčovu strategii označíme za Nashovu rovnováhu právě tehdy, neexistuje-li žádná jiná strategie, která by ho mohla porazit [25]. Bylo dokázáno [26], že v každé konečné nekooperativní hře existuje strategie odpovídající Nashově rovnováze. Poker tato pravidla splňuje, takže tato strategie určitě existuje.
- **Hráče přiblížíme k Nashově rovnováze tím, že ho naučíme hrát lépe než dosud proti dostatečně velké množině soupeřů**: Pokud budou množinou hráčů rovnoměrně zastoupeny všechny existující strategie v pokeru, tak naučíme-li náhodného hráče hrát proti nim lépe, přiblížíme jeho strategii k Nashově rovnováze.

Z těchto předpokladů je možné usoudit následující. Pokud budeme pracovat s dostatečně velkou množinou hráčů, kteří se budou náhodně střídat v učení, měli by se postupně zlepšovat. Nehrozí tedy uváznutí v lokálním minimu, kde hráč poráží všechny své soupeře v simulaci, ale nedokáže hrát proti

nikomu jinému. Čím více nezávislých hráčů bude v množině, tím stabilnější, ale pomalejší bude hledání nejlepšího hráče.

V naší variantě pokeru proti sobě hraje pouze 6 hráčů současně. Při tréninku i zjišťování kvality hráčů musíme tedy neustále náhodně měnit soupeře a pozice u stolu, aby byl počet situací, do kterých se testovaný hráč dostane, co možná nejvyrovnanější.

Zbytek práce se zabývá 4. řádkem algoritmu 1 - naučením hráče hrát co možná nejlépe proti množině soupeřů.

2.3.1 Potřebná omezení hry

Jak již bylo řečeno, tato práce se zabývá pouze bezlimitní variantou pokeru, kde je velikost sázky omezena pouze zdola a zhora. Záleží zcela na hráči, jak velkou částku se v rámci pravidel rozhodne do hry vložit. Může vsadit žetonů 10, nebo 11, 12 atd. Pro použitelnost neuronových sítí musíme možné tahy hráče nějakým způsobem omezit. Bylo vybráno 5 různých akcí, ze kterých si neuronová síť musí pokaždé vybrat jednu:

- Fold - Zahození karet
- Check / Call - Check pokud není potřebné dorovnat sázku pro pokračování ve hře, jinak Call.
- Bet 50% - Vsazení hodnoty rovné 50% společného banku
- Bet 75% - Vsazení hodnoty rovné 75% společného banku
- Bet 100% - Vsazení hodnoty rovné 100% společného banku

2.3.2 Dva pohledy na učení

K tomu, abychom mohli neuronovou síť něco naučit, potřebujeme tréninková data a ta musíme shromáždit. Existují dva způsoby, jak k tomuto problému přistupovat. Obě varianty byly otestovány a porovnány.

Tou první je rozdělení tréninku neuronové sítě na dvě části - vygenerování tréninkových dat a samotné učení neuronové sítě. Nejdříve zjistíme, jak by se měl agent (hráč) v prostředí (pokeru) chovat a následně se pokusíme vytvořit neuronovou síť, která se toto chování naučí. Zde narážíme na problém, že pro velká prostředí, kterým poker bezesporu je 2.1.2, se nám informace nevejdou do paměti. V tomto případě omezíme počet zapamatovaných stavů stejnou hodnotou jako počet zkušeností 2.4.5. Neuronovou síť tedy naučíme pouze nejvíce se opakující stavy a poté budeme spoléhat na správné vyhodnocení a zobecnění dat. Tento styl učení v práci dále označuje pojem Offline učení.

Druhou možností je Online učení. Agent je umístěn do zcela neznámého prostředí a generuje za běhu tréninková data, ze kterých se ihned učí. Tato

metoda sklízí v poslední době velké úspěchy [4] [27] a je hlavní náplní této práce.

Kromě stylu generování dat a učení se, je rozdíl i ve struktuře sítě a problému, který se síť snaží naučit. Pokud nejdříve vygenerujeme data a ta se potom snažíme síť naučit, jedná se o problém klasifikační a poslední vrstva sítě bude využívat aktivační funkci SoftMax. Po vygenerování dat známe předpokládané zisky jednotlivých akcí. Stačí tedy naučit neuronovou síť odhadnout akci s potenciálně největším ziskem, ale předpokládanou hodnotu zisku této akce nemusíme brát v úvahu. Výsledkem neuronové sítě bude tedy pět hodnot udávajících pravděpodobnost, s jakou je určitá akce strategicky nejvýhodnější.

U Online učení tento princip využít nemůžeme. V pokeru jsou všechny akce porovnávány z dlouhodobého hlediska. Hráčova akce, která vyústí 9x ve ztrátu 10 žetonů a jednou ve výhru 200, má průměrný zisk 11 žetonů a to je jediné, na čem záleží. Pokud by se síť za běhu učila akce klasifikovat, tak by se 9x naučila zahodit a pouze jednou pokračovat, takže by v této situaci pokaždé zahodila. To je ale nesprávné chování. I když jsme vyhráli pouze jednou, tak se nám podařilo získat více žetonů, než jsme předtím ztratili, a tudíž je lepší nezahazovat. Musíme tedy zvolit aktivační funkci, která bere v potaz velikost možného zisku a tou je funkce lineární. Pokud takovou síť 9x naučíme hodnotu -10 a jednou +200, tak by konečný odhad sítě měl být +11.

Samozřejmě by bylo možné i u Offline učení využívat lineární aktivační funkci a snažit se síť naučit přesné hodnoty Q-funkce 2.4.2. Toto řešení ale o mnoho zhoršilo efektivitu Offline učení. Naučit neuronovou síť klasifikovat akce podle tréninkových dat je jednodušší problém, než se snažit ji naučit odhadnout přesnou hodnotu Q-funkce.

2.4 Posilované učení

Posilované učení je druh strojového učení, které připomíná způsob, jakým se učí lidé. Agent je umístěn do neznámého prostředí a pomocí metody pokus-omyl zjišťuje, které akce jsou v určité situaci výhodné a které zase ne. Z takto nabytých zkušeností se ihned učí a zdokonaluje se.

Základem posilovaného učení je Markovův rozhodovací proces (MDP).

2.4.1 Markovův rozhodovací proces

Markovův rozhodovací proces (MDP) je pětice (S, A, T, R, γ), kde

- $\boldsymbol{S}$ - Konečná množina stavů
- $\boldsymbol{A}$ - Konečná množina akcí
- $\boldsymbol{T}$ - Přesunové pravděpodobnosti, neboli pravděpodobnost, že pokud provedu za stavu $\boldsymbol{s}$ akci $\boldsymbol{a}$, dostanu se do stavu $\boldsymbol{s'}$

- $\boldsymbol{R}$ - Okamžitá odměna ve stavu $\boldsymbol{S}$
- $\boldsymbol{\gamma}$ - Faktor důležitosti budoucích odměn

MDP je charakterický tím, že rozhodnutí agenta ve stavu S záleží pouze na tomto stavu a historie předcházejících akcí není důležitá. V pokeru toto pravidlo neplatí. Zajisté hru ovlivňuje akce soupeře v předchozím kole sázek. Aby bylo možné považovat poker za MDP je do stavu S zakomponována informace o historii. Toto je běžná praxe využitá i při hraní her Atari [6].

Pro MDP poté můžeme nalézt ideální rozhodovací politiku pomocí iterativních řešení. Tato politika v každém stavu vybere akci s největší možnou odměnou. Jedním z algoritmů pro její nalezení je algoritmus Q-učení. [28]

2.4.2 Q-učení (Offline)

Q-učení patří mezi techniky posilovaného učení. Může být použito k nalezení ideální politiky pro libovolný konečný Markovův rozhodovací proces. Základem je Q-funkce:

$$Q(s,a) = r + \gamma * max_{a'} Q(s',a')$$

kde

- $\boldsymbol{r}$ - okamžitá odměna po provedení akce a
- $\boldsymbol{\gamma}$ - faktor zohlednění odměny v budoucnu

Q-funkce reprezentuje kvalitu určité akce a ve stavu s. Každé kombinaci přiřadí hodnotu, kterou agent získá, pokud od této chvíle bude hrát bez chyby.

Nejjednodušší způsob nalezení Q-funkce používá tabulku - viz algoritmus 2. Konstanta α v algoritmu je v rozmezí 0 - 1 a určuje rychlost učení. Pokud je rovna jedné, tak se $Q[s,a]$ společně vyruší a zůstane rovnice identická s rovnicí základní. Hodnota $Max_{a'}Q[s',a']$ používaná ke změně $Q[s,a]$ je ze začátku pouze odhad a může být úplně špatná. Tento odhad se ale iteraci od iterace zpřesňuje a bylo dokázáno [29], že po provedení dostatečného množství iterací začne Q-funkce konvergovat a odpovídat její skutečné hodnotě. [30]

Algorithm 2 Q-učení pomocí tabulky

```
náhodně inicializuj Q[pocetStavu][pocetAkci]
zjisti počáteční stav S
repeat
  vyber a proveď akci A
  zjisti odměnu R a následující stav S'
  Q[s,a] = Q[s,a] + α(r + γMax_a' Q[s',a'] − Q[s,a])
  s = s'
until konec
```

2.4.3 Neuronové sítě jako Q-funkce (Online)

Potíže s používáním tabulky pro reprezentaci Q-funkce nastávají, pokud se pokoušíme vyřešit problém, který má velké množství stavů. Poker mezi tyto problémy patří 2.1.2. Tak velkou tabulku není možné do paměti uložit. Ukládáním pouze nejfrekventovanějších stavů se sice vyřeší problém s pamětí, ale počet iterací potřebných ke konvergenci se mnohonásobně zvýší a reálně už není možné spočítat. Také bychom chtěli znát alespoň přibližnou strategii ve stavech, které ještě nenastaly. Všechny tyto problémy můžeme vyřešit použitím neuronových sítí, které vynikají v generalizování složitých dat. Místo tabulky bude neuronová síť, která bude mít na vstupu informace o stavu hry a jejím výstupem bude aproximace hodnoty Q-funkce. Upravování hodnot v tabulce se změní na učení neuronové sítě pomocí zpětné propagace. Algoritmus 3 využívá neuronovou síť jako Q-funkci. [23]

Algorithm 3 Q-učení s využitím neuronové sítě

Vytvoř neuronovou síť s náhodně inicializovanými hodnotami vah
zjisti počáteční stav S
repeat
  vypočítej hodnoty všech akcí pro stav S
  vyber a proveď akci A
  zjisti odměnu R a vypočítej výsledek sítě pro stav S'
  pro zvolenou akci nastav cílovou hodnotu na $r * \gamma max_{a'} Q(s', a')$
  ostatní hodnoty akcí ponechej
  nauč síť pomocí nově vypočtených hodnot
until konec

2.4.4 Explorace vs Exploitace

Jedním z kroků algoritmu je výběr akce A. Jak vybrat akci, kterou agent provede? Intuitivním řešením je zkoušet vždy nejlepší akci, neboli tu s největší Q-hodnotou. Hned na začátku ale generujeme náhodně inicializovanou neuronovou síť. Může se stát, že hodnoty budou vygenerovány poněkud nešťastně a uvázneme v lokálním minimu. Opakem je zcela náhodný výběr akce. Zde ale narážíme na problém. Pokud neodsimulujeme opravdu velké množství stavů (v praxi nereálné), tak získáme pouze přibližný odhad Q-funkce. Ideální řešení se nachází někde uprostřed. Nejdříve bychom chtěli přibližně zjistit, které akce jsou nejslibnější a tyto akce poté důkladněji prozkoumat [31].

Větší pravděpodobnost explorace zjišťuje obecné vlastnosti prostředí a menší naopak ty specifické.

K dosažení zmíněného efektu byla použita ϵ-hladová explorace, která s pravděpodobností $1 - \epsilon$ vybere akci s největší hodnotou a jinak je akce zvolena náhodně. Simulace začíná s vysokou hodnotou explorace a postupně ji snižuje, jak získává více informací o prostředí a přibližnou hodnotu jednotlivých akcí.

Počáteční hodnota ϵ je 1. Celkový čas věnovaný tréninku je rozdělen na 100 částí a po uplynutí každé z nich je koeficient snížen o předem danou kostantu, jejíž hodnota byla zjištěna experimentálně 4.1. Bylo také nastaveno minimum na 0.1, aby byl alespoň částečný druh explorace zachován vždy.

2.4.5 Opakování zkušeností

Algortimus v podobě zmíněné výše 3 v praxi převážně nefunguje. Neuronové sítě totiž uváznou v lokálním minimu a nemají prostředky na to, aby se z něho dostaly. Je to způsobeno učením se pouze na jednom trénovacím vzorku. Neuronová síť se tedy naučí hrát lépe v této jedné situaci, ale zhorší se v situacích, které se zrovna neučí.

Řešením tohoto problému je zavedení paměti na x posledních zkušeností. Po každém zvolení akce si uložíme čtveřici $< s, a, r, s' >$. Trénování poté probíhá náhodným zvolením několika vzorků - dávky, na kterých bude síť trénována. Tímto zamezíme preferování aktuálních situací společně s učením se pouze na jednom trénovacím vzorku.

V předchozím algoritmu se objevují dvě proměnné - počet uložených zkušeností a velikost dávky. Pro určení těchto čísel neexistují žádné vzorce, které by nám řekly ideální hodnoty.

Platí pouze, že s rostoucím množstvím zapamatovaných zkušeností klesá důležitost aktuálně přidaných zkušeností a že narůstající počet prvků v dávce zvyšuje stabilitu, ale zpomaluje celé učení [32].

Použité hodnoty byly zjištěny experimentálně. Výsledky odpovídají tabulce 4.2:

2.4.6 Cílová síť

Při běhu našeho algoritmu nastavujeme cílovou hodnotu Q-funkce pro učení jako:

$$Q(s,a) -> r + \gamma max_a Q(s',a)$$

Můžeme vidět, že cíl závisí na momentální neuronové síti. Neuronová síť se chová jako celek a každá naučená nová hodnota Q-funkce pro stav S ovlivňuje oblast okolo něj. Body $Q(s,a)$ a $Q(s',a)$ jsou velmi blízko u sebe, protože popisují přechod ze stavu S do S'. Toto přináší problém, že s každým naučením se ze zkušeností, se náš cíl posune. Jako se kočka snaží chytit svůj ocas, neuronová síť nastaví sama sebe jako svůj cíl a následuje ho. Toto chování vede k nestabilitě, oscilaci a divergenci. [33]

Pro překonání problému navrhli výzkumníci [27] vytvoření druhé sítě, která bude počítat cíle pro učení. Tato síť je kopie té originální, pouze zmrazená v čase. Poskytuje stabilní hodnoty pro Q-funkci a dovoluje algoritmu konvergovat k určitému cíli.

$$Q(s,a) -> r + \gamma max_a Q'(s',a)$$

Po několika krocích je cílová síť upravena překopírováním vah ze sítě aktuální. Aby byla tato metoda efektivní, musí být interval mezi aktualizacemi dostatečně velký, aby stihla originální síť konvergovat.

Rychlost učení je tímto poměrně zpomalena. Jakákoliv změna v Q-funkci je propagována až ve chvíli, kdy je síť aktualizována. Intervaly mezi aktualizacemi jsou většinou v rámci tisícovek kroků.

Tato změna umožní algoritmu, aby se naučil správnému chování v komplikovaných prostředích, ovšem na úkor rychlosti učení [27].

V našem prostředí bylo pomocí cílové sítě dosaženo výsledků dostupných v tabulce 4.3.

2.4.7 Dvojité učení

Jedním z problémů algoritmu je fakt, že agent má tendence přeceňovat hodnotu Q-funkce kvůli maximu, které je použito pro získání požadované hodnoty:

$$Q(s,a) -> r + \gamma max_a Q(s',a)$$

Pro znázornění tohoto problému si můžeme představit následující situaci. Existuje stav, pro který mají všechny jeho akce stejnou opravdovou hodnotu Q-funkce. Její odhad je ale přirozeně nestabilní a pro každou z nich se liší. Kvůli přítomnosti funkce max je za cílovou akci zvolena ta s nejvyšší pozitivní chybou a její hodnota je dále propagována zpět do předchozích stavů. Toto vede k přeceňování hodnot a narušuje stabilitu učení.

Navrhované řešení tohoto problému se nazývá dvojité učení [34] [35]. V tomto novém algortimu se nezávisle učí dvě neuronové sítě reprezentující Q-funkci - Q_1 a Q_2. Jedna z nich je poté použita k vybrání akce, kterou agent prozkoumá, a druhá k odhadu její hodnoty. Sítě se v tréninku náhodně střídají a k učení využívají následující předpis:

$$Q_1(s,a) -> r + \gamma Q_2(s', argmax_a Q_1(s',a))$$

a

$$Q_2(s,a) -> r + \gamma Q_1(s', argmax_a Q_2(s',a))$$

Bylo dokázáno [35], že oddělení výběru akce a její hodnoty tímto způsobem vede k eliminování přeceňování hodnot.

Při implementaci této změny byl vzat v potaz fakt, že již dvě neuronové sítě v algoritmu využíváme. Pro odhad hodnoty byla použita cílová síť. I když tyto sítě nejsou nezávislé, zakomponování dvojitého učení znamenalo pouze minimální změnu. [33]

2.4.8 Vstupy neuronové sítě

V pokeru je mnoho proměnných, které můžeme sledovat. Mohou to být karty, které držíme, počet žetonů, nebo předchozí akce našich protihráčů. Takovýchto

útržkovitých informací je více. Důraz byl kromě samotné reprezentace stavu kladen také na zmenšení prostoru všech kombinací bez ztráty důležité informace. Celkem bylo naimplementováno 7 různých informací o hře:

- **Mé vlastní karty** - Každý hráč drží 2 karty. Každá karta je identifikována pomocí její hodnoty a barvy. Jelikož v pokeru mají všechny barvy stejnou hodnotu a jedna není lepší než druhá, můžeme tuto informaci redukovat na 3 proměnné: Hodnota první karty, druhé karty a to, jestli jsou stejné barvy.

- **Společné karty** - Postupně je na stůl vyloženo celkem 5 karet. Nezáleží, stejně jako u našich vlastních karet, jaké barvy jsou. Co nás zajímá je počet karet stejné barvy a s tím související i kolik karet jedné barvy dáme dohromady s ohledem na to, jaké karty držíme v ruce. Celkem zde tedy sledujeme 7 proměnných - 5x hodnotu karty (pokud karta zatím nebyla rozdána, tak na její místo dáme symbol označující tuto skutečnost), počet karet stejné barvy na stole a počet karet stejné barvy po přičtení karet, které držíme. Tímto se ztratí zlomek informace, který by ale neměl být relevantní. Situace, ve kterých je důležité, zda jsou karty v barvě např. AT5 nebo AK6 nastávají tak zřídka, že by to nemělo ovlivnit potenciální zisk sítě.

- **Pozice u stolu** - Pořadí hráče ve hře. Hráč na místě malého blindu je na pozici 0 a dealer 5.

- **Počet soupeřů ve hře** - Počet hráčů, kteří jsou stále ve hře a nezahodili své karty. Nabývá hodnot 1-5.

- **Nevyjádřených hráčů** - Kolik hráčů bude v tomto sázkovém kole hrát po nás.

- **Pot odds** - Hodnota vyjadřující poměr mezi žetony, které musíme vložit do banku a možnou výhrou. Vypočítá se pomocí vztahu $mpz/(mpz+ps)$, kde mpz je rovno minimálnímu počtu žetonů, které musí hráč vložit do banku, aby mohl pokračovat ve hře. Ps odpovídá velikosti banku. Tato proměnná nabývá hodnot 0 (má-li hráč možnost počkat a nedělat nic) až 0,5 (maximální velikost proměnné nemůže být vyšší, protože jsou potřeba alespoň 2 hráči ve hře). Jelikož je číslo reálné (počet možností je nekonečný), byla výsledná hodnota rozdělena do 6 kategorií (0 - 0.1 - 0.2 - 0.3 - 0.4 - 0.5).

- **Akce soupeřů** Poslední informací o stavu hry jsou akce našich soupeřů. Před námi se mohlo vyjádřit maximálně až 5 soupeřů a každý z nich musel zahrát jednu z pěti možných akcí. Pokud před námi nehrálo všech 5 hráčů, doplníme zbytek speciálním symbolem signalizujícím zmíněný fakt.

Tabulka 2.1: Příklad kódování srdcové desítky

	Kód hodnoty	Kód barvy	Úspěšnost
Základní kódování	8	1	0.798
Normalizace	0.666	0.333	0.863
Binární kódování	**0000 0000 1000 0**	**0100**	**1**

- **Historie** Tato informace úplně nepatří mezi výše uvedené, ale spíše je všechny doplňuje. Můžeme si zvolit počet posledních stavů, které budou připojeny k informaci o stavu aktuálním. Uchování historie nám dovoluje zjistit, jak jsme se do tohoto stavu dostali.

2.4.8.1 Kódování

Jednotlivé vstupy je možné kódovat více způsoby. Pravděpodobně nejjednodušší z nich je přímá reprezentace hodnotou. V tomto kódování je potom například karta znázorněna dvěma vstupy: její hodnotou a barvou. Hodnota karty může nabývat čísel v rozmezí 2 až 14, resp. 0 až 12 a její barva 1 až 4, resp. 0 až 3.

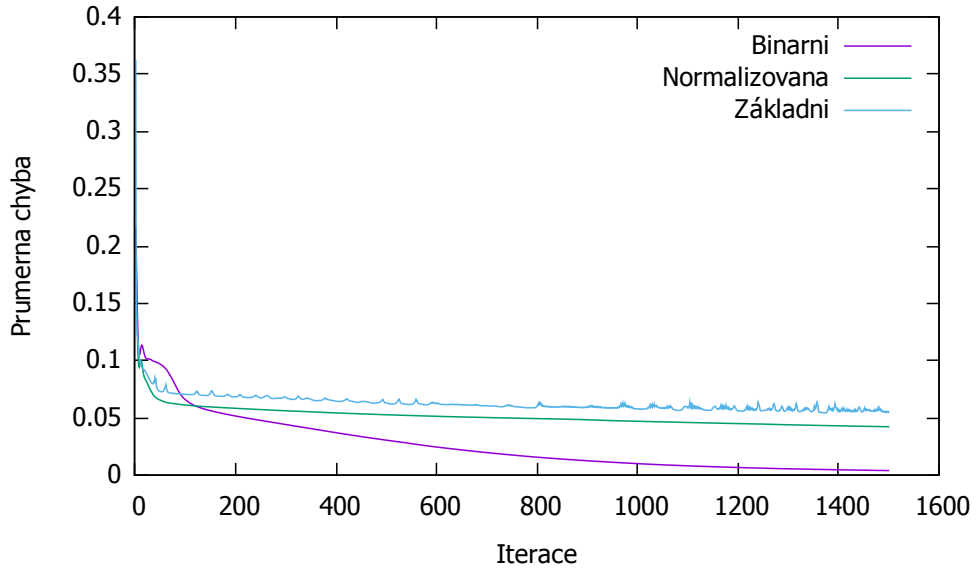
Toto kódování ale způsobuje problémy s učením i pro nejzákladnější informace ze hry - hráčovy vlastní karty. Neuronová síť by nedokázala vyhodnotit složitější situace, když se neosvědčila ani na těch výchozích. Problém normalizace vstupů představuje u neuronových sítí běžnou komplikaci.

Druhá možnost využívá pro kódování normalizaci. V tomto případě jsou hodnoty čísel přeškálovány, aby nejnižší z nich byla nula, nejvyšší jedna a poměry mezi čísly zůstaly zachovány.

Binární kódování zastupuje poslední variantu, v níž jsou jednotlivá čísla převedena na pole, jehož počet prvků je roven maximální hodnotě, kterou může číslo nabývat. Všechny hodnoty v poli jsou rovny nule kromě té, jejíž index odpovídá originální hodnotě čísla. Karta by se tedy zakódovala jako dvě pole. První o velikosti 13 pro hodnotu karty a druhé velikosti 4 pro její barvu.

Pro zjištění nejvhodnějšího kódování pro toto použití neuronových sítí byly všechny tři možnosti otestovány. Pomocí Offline Q-učení 2.4.2 byla vygenerována tréninková data pro jednoduchou reprezentaci stavu - hráčovy vlastní karty. Na těchto datech bylo různé kódování vyzkoušeno.

Výsledky dokumentuje tabulka 2.1. Přiložený je také graf znázorňující průměrnou chybu sítě s ohledem na počet tréninkových iterací 2.6. Nejvyšší úspěšnost prokázala síť, která vstupní data kódovala binárně. Toto kódování bylo následně využito pro všechny reprezentace stavu v celém programu.



Obrázek 2.6: Průběh trénování neuronové sítě pro rozdílná kódování

Kapitola 3

Realizace

Aplikace byla naimplementována v jazyce Java a skládá se ze dvou hlavních částí - pokerové simulace, která podporuje Offline i Online Q-učení a dále z grafického rozhraní aplikace, kde se může uživatel utkat proti naučeným neuronovým sítím.

3.1 Neuronové sítě

Pro neuronové sítě byla ve finální formě využita knihovna Deeplearning4j [37]. Neuronová síť v té nejjednodušší formě byla naprogramována také, ale její výsledky byly o mnoho horší než při využití specializované knihovny. Od této varianty bylo tedy upuštěno. Vlastní implementaci neuronové sítě můžeme najít v balíčku prediction.neuralNetworks.model společně s výběrem různých aktivačních funkcí. Nachází se zde také učení za pomoci zpětné propagace, které podporuje i paralelní výpočet.

3.2 Poker

Vše se točí okolo pokeru, který v našem prostředí realizuje třída Game. Ta podle pravidel řídí hru a volá příslušné metody ostatních tříd, které ji v simulací pomáhají:

- **Card** - Třída reprezentující kartu, která má hodnotu a barvu.

```
public class Card implements Comparable<Card>,
                                    Serializable {

private final int value;
private final Color color;

.
```

```
.
.

}
```

- **Color** - Výčet barev v pokeru.

```
public enum Color implements Serializable{
        SPADE, HEART, CLUB, DIAMOND
}
```

- **Pot** - Bank hry, který udržuje informaci o počtu žetonů. Umí také vybrat hráče s nejlepší karetní kombinací a tudíž určit vítěže. Jedná se o časově náročnou operaci, jelikož musí být postupně vzata v úvahu každá z 10 možných výherních kombinací. Využit byl proto algoritmus vyvinutý na pokerovém fóru TwoPlusTwo [38], který tento problém řeší pomocí vyhledávací tabulky, kde jsou uloženy všechny možnosti, jak sestavit sedm karet. Aby tato metoda fungovala, potřebuje cca 100MB paměti. Základní složkou simulace je selekce hráče s nejlepšími kartami. Cena v podobě potřebné paměti proto není tak vysoká, abychom kvůlí ní tento algoritmus nevyužili. Po jeho zakomponování se celkový výkon zvýšil několikanásobně. Ukázka kódu, který zjišťuje nejlepší možnou kombinaci:

```
public static int lookupHand7(int[] cards) {
    int pCards = 0;
    int p = HR[53 + cards[pCards++]];
    p = HR[p + cards[pCards++]];
    p = HR[p + cards[pCards++]];
    p = HR[p + cards[pCards++]];
    p = HR[p + cards[pCards++]];
    p = HR[p + cards[pCards++]];
    return HR[p + cards[pCards]];
}
```

- **Board** - Společné karty vyložené na stole
- **Move** - Detailní informace o hráčově tahu zahrnující počet žetonů vložených do banku a jeho akci (Fold, Check, Call nebo Raise)

3.3 Hráč

Všechny informace, které specifikují každého hráče, jsou uloženy v třídě Player. Mezi znalosti jednotlivých hráčů patří počet žetonů, jejich karty nebo kolik odehráli celkem hand. Tyto informace si udržuje hráč nezávisle na typu umělé inteligence. V momentě, kdy na hráče přijde řada a je na tahu, tak od

své instance třídy PredictionModel zjistí akci, kterou by měl provést 2.3.1 a podle ní např. vsadí do hry požadovaný počet žetonů.

```
public Move play(Game game, Map<Player,
                List<PlayerActionHistory>> playerActions) {
  actedThisBettingRound = true;
  Prediction prediction = playerAI
                    .predict(game, this, playerActions);
  return playSelectedMove(game, playerActions, prediction);
}
```

3.3.1 PredictionModel

PredictionModel je abstraktní třída reprezentující mozek hráče. K ovládání akcí může sloužit kterákoliv třída, která dědí z PredictionModelu. V této práci byly naimplementovány celkem tři možnosti, jak řídit hráčovy akce. První varianta využívá vlastní implementaci neuronových sítí, druhá knihovnu Deeplearning4j a třetí slouží k ovládání hráče člověkem.

Nachází se zde pouze jedna metoda, **predict**, která za využití informací o stavu hry vrátí akci, kterou má hráč vykonat. Zbytek programu nijak neovlivní, zda je akce zvolena za pomoci náhody, neuronových sítí nebo pokynem od uživatele.

```
public abstract Prediction predict(Game game,
        Player player,
Map<Player, List<PlayerActionHistory>> playerActions);
```

3.3.2 DataCreator

Každá instance třídy PredictionModel má také svůj DataCreator, který je zodpovědný za extrahování informací ze hry. Přidáme-li objekt implementující rozhraní **CurrentStateInformation** do seznamu, je možné specifikovat, která data bude mít hráč k dispozici. Dále je možné nastavit velikost historie posledních hráčových akcí. Vstupní data použitelná pro neuronovou síť jsou vygenerována metodou **getCurrentState**.

```
public State getCurrentState(Game game, Player player,
 List<PlayerActionHistory> playerActions) {
  double result[] = new double[getTotalSize()];
        fillCurrentState(game, player, result);
        fillHistory(playerActions, result);
        return new State(result);
}

public interface CurrentStateInformation {
```

```
    int getSize();
    void fillCurrentStateIntoArray(Game game,
            Player player, int startIndex, double array[]);
}
```

3.4 Q-učení

Q-učení zajišťují dvě třídy, QLearningNaiveTable a QLearningNeuralNetworks, které obsahují většinu logiky týkající se učení. Jejich hlavní metody slouží k přidávání nových zkušeností a získávání tréninkových dávek.

Zkušenosti jsou do paměti agenta vloženy po provedení každého tahu. V pokeru ale okamžitý zisk z uskutečněné akce zjistíme až v akci následující. Například naše sázka může donutit všechny soupeře zahodit své karty a tím je zisk z této akce roven velikosti společného banku. Soupeři se ale také mohou rozhodnout dorovnat a potom je okamžitý zisk záporný a odpovídá počtu vsazených žetonů. Novou zkušenost tedy přidáme do paměti v následujících dvou případech: Agent je na tahu a v této handě už nějakou akci provedl (jejím výsledkem je momentální stav), nebo hra skončila a je potřeba přidat poslední agentovu akci.

3.5 Podpora více vláken

Program spouští jak simulaci pro zjištění kvality hráčů, tak samotné učení na více vláknech. Na začátku programu je zjištěn celkový počet jader procesoru a poté je vytvořeno jím odpovídající množství pracovních vláken.

U samotné simulace je výrazné zrychlení, jelikož každé vlákno má své vlastní prostředky a potřeba synchronizace je minimální. Po odehrání zadaného počtu hand jsou pouze sečteny výsledky všech vláken.

Při učení je zrychlení menší, protože se najednou učí pouze 1 hráč. Není nutné hrát handy, kde by tento hráč chyběl, a proto je přítomný ve všech. Po každé odehrané handě jsou nově nabyté zkušenosti přidány do paměti, která už je sdílena všemi vlákny. Společná je také neuronová síť, a proto není možné jí zároveň používat k zjištění dalšího agentova tahu a k jeho učení. Probíhat může pouze jedna z těchto dvou akcí současně.

Zrychlení, které poskytuje více vláken, je pouze v tom, že nemusíme čekat na výpočet tahů soupeřů. Zatímco se neuronová síť učí ve vlákně jedna, druhé vlákno zpracuje tahy všech soupeřů a ve chvíli, kdy jsou společné prostředky uvolněny, s nimi může ihned začít pracovat.

Zrychlení na testovaném počítači znázorňuje tabulka 3.1.

Třídou představující úkol pro vlákno je SimulationThread. Její úkol je jednoduchý. Dokud není odehráno požadované množství hand, vybere vždy v cyklu šest náhodných hráčů (jedná-li se o učení, tak zaručí přítomnost učícího se hráče) a poté vytvoří novou instanci třídy Game a nechá hráče proti sobě

Tabulka 3.1: Ukázka zrychlení programu s využitím více vláken

Počet vláken	Učení nebo simulace	Počet hand za vteřinu
1	Simulace	16003
2	Simulace	27330
4	Simulace	45120
8	Simulace	62560
1	Učení	1024
2	Učení	1245
4	Učení	1268
8	Učení	1266

odehrát šest hand. Tolik hand je potřeba, aby se pozice dealera posunula o celé kolo.

3.6 Grafické prostředí aplikace

Pro aplikace bylo vytvořeno grafické prostředí za pomocí technologie JavaFX. Zde je možné poměřit síly s neuronovými sítěmi vytvořenými v simulaci. Ukázka grafického prostředí je dostupná níže 3.1.

Obrázek 3.1: Grafické prostředí aplikace

Kapitola 4

Experimenty a výsledky

V této práci bylo implementováno několik úprav algoritmu Q-učení. Neuronové sítě jsou také známy svým problémem ladění parametrů jako je počet neuronů nebo rychlost učení. Pokud je jeden z parametrů moc velký nebo malý, mohou být výsledky sítě velmi nespolehlivé. Všechny změny byly nejprve otestovány a až poté zavedeny do programu nastálo.

4.1 Metodika testování

Všechny testované změny byly ověřeny u sítí, které disponovaly finální podobou vstupů ze hry. V případě, že by testy byly provedeny na vybraném podproblému, jejich výsledky by nemusely odpovídat ideálním hodnotám pro celkové řešení. Metodika testování byla stejná pro všechny změny.

Pro zjištění odpovídajících hodnot byla neuronová sít vždy učena celkem 15 minut a každé 3 minuty byla otestována její schopnost hrát proti soupeřům. Veškeré výsledky byly ukládány a po uplynutí zmíněných 15 minut byla jako výsledná síť učení vybrána ta s největším průměrným ziskem na 100 odehraných hand.

4.2 Explorace vs Exploitace

Vyzkoušeny byly různé hodnoty koeficientu snížení ϵ. Čím větší hodnota, tím dříve se agent začne zaměřovat na zkoumání pouze nejziskovějších akcí. Z tabulky 4.1 můžeme vidět, že nejlepších výsledků dosáhla velikost koeficientu rovna 0.02. Agent, který využívá tuto hodnotu, stráví první polovinu času spíše explorací a druhou už zkouší pouze nejslibnější akce.

Tabulka 4.1: Explorace vs Exploitace. Jednotlivé hodnoty v tabulce níže udávají zisk neuronové sítě v počtu velkých povinný sázek na 100 hand za využití určitého koeficientu snížení. Neuronová síť se učila 15 minut a každé tři minuty byl změřen její zisk.

Koeficient snížení	3 min	6 min	9 min	12 min	15 min
0.01	-60.21	-30	-10.17	-17.51	5.36
0.015	-17.57	-14.01	-29.12	-5.62	12.52
0.02	**-7.79**	**-17.22**	**-13.50**	**-6.65**	**29.89**
0.03	-88.51	25.21	9.3	10.35	-98.7

4.3 Opakování zkušeností

Upravováním velikosti dávky a počtu zapamatovaných zkušeností se buď více přibližujeme Online nebo Offline učení. Zde se ukázalo 4.2, že ideální velikost dávky je 64 prvků. Výsledky odpovídají předpokladu, že více zkušeností v dávce zlepšuje stabilitu a kvalitu učení. Test s velikostí dávky 128 ale ukazuje opak. Je to způsobeno tím, že test je omezen časem a ne počtem odehraných hand. Zlepšení na jednu odehranou handu je sice nejlepší pro největší velikost dávky, ale počet odsimulovaných hand je poloviční a tudíž se síť ve výsledku učí pomaleji.

Tabulka 4.2: Opakování zkušeností. Jednotlivé hodnoty v tabulce níže udávají zisk neuronové sítě v počtu velkých povinný sázek na 100 hand za využití různých velikostí dávky a paměti. Neuronová síť se učila 15 minut a každé tři minuty byl změřen její zisk.

Dávka	Paměť	3 min	6 min	9 min	12 min	15 min
1	1	-271.23	-26.34	-8.22	-53.47	-5.23
16	10000	-54.56	-2.10	28.16	9.14	-0.08
32	10000	-5.38	4.49	18.76	32.82	17.93
64	**10000**	**8.36**	**37.26**	**42.11**	**51.77**	**30.91**
128	10000	-2.60	-1.08	25.16	26.76	29.28

4.4 Cílová síť

Z tabulky nacházející se níže lze vydedukovat změnu v učení při využití cílové sítě 4.3. Data potvrzují, že zavedení cílové sítě opravdu stabilizuje učení, avšak není zde patrná korelace mezi zlepšením neuronové sítě a hodnotou itervalu obnovení. Vybrán byl nakonec interval obnovení rovný 100.

Tabulka 4.3: Využití cílové sítě. Jednotlivé hodnoty v tabulce níže udávají zisk neuronové sítě v počtu velkých povinný sázek na 100 hand za využití určitých intervalů obnovení cílové sítě. Neuronová síť se učila 15 minut a každé tři minuty byl změřen její zisk.

Cílová síť	Interval obnovení	3 min	6 min	9 min	12 min	15 min
ne	-	-22.73	8.66	27.60	2.25	22.25
ano	**100**	**-0.30**	**25.40**	**35.82**	**27.57**	**23.51**
ano	250	8.58	14.44	14.04	23.11	23.97
ano	500	-0.05	10.57	24.19	19.08	-4.8
ano	1000	7.65	3.63	-1.17	17.27	18.57

4.5 Dvojité učení

Poslední změnou určenou k zvýšení stabilizace učení bylo zavedení dvojitého učení. Výsledky jsou dostupné v tabulce níže 4.4. Chování popsané v práci, která tuto metodu představila [34], se potvrdilo i zde. Pro využití potenciálu dvojitého učení musí být velikost intervalu obnovení cílové sítě větší, než když je síť aplikována samostatně 4.3, aby na sobě obě neuronové sítě nebyly tolik závislé. Nejlépe v testech obstála hodnota intervalu obnovení rovna 2500.

Tabulka 4.4: Dvojité učení. Jednotlivé hodnoty v tabulce níže udávají zisk neuronové sítě v počtu velkých povinný sázek na 100 hand za využití dvojitého učení a různých intervalů obnovení cílové sítě. Neuronová síť se učila 15 minut a každé tři minuty byl změřen její zisk.

Dvojité učení	Interval	3 min	6 min	9 min	12 min	15 min
ne	100	-5.61	10.25	-1.25	5.65	19.58
ano	100	-13.41	12.44	5.04	28.2	28.65
ano	1000	-24.41	3.92	25.80	12.67	25.93
ano	**2500**	**19.12**	**4.18**	**31.31**	**1.63**	**18.01**

4.6 Postupné přidávání vstupů

Ne všechny informace mají stejnou váhu. Některé jsou kriticky důležité a ostatní nám k lepší hře nemusí pomoci vůbec. Místo použití všech informací a doufání, že si s tím neuronová síť "nějak poradí", byl zvolen jiný postup. Jednotlivé informace byly přidávány v závislosti na jejich důležitosti. Ta byla zjištěna experimentálně pomocí algoritmu 4. Důvodem pro tento postup nebyla pouze zmíněná neznalost důležitosti jednotlivých informací. Byla to také snaha o nalezení hranice, kdy Offline metoda přestane fungovat a začne být výhodnější posilované učení.

Algorithm 4 Algoritmus pro zjištění důležitosti jednotlivých informací o stavu hry

vytvoř prázdnou množinu nejlepších stavů - Q
repeat
  vygeneruj množinu sítí se všemi stavy z Q + jedním náhodným
  nauč sítě hrát co možná nejlépe za pomoci pouze těchto informací o hře
  náhodně přidanou informaci nejlépe hrající sítě vlož do množiny Q
until konec

Tabulka 4.5: Výběr nejdůležitější vstupní informace

Druh učení	Informace	Zisk bb/100
Offline	Hráčovy karty	109.7
Online	Hráčovy karty	90.9
Online	Akce soupeřů	-11.2
Offline	Akce soupeřů	-15.8
Offline	Společné karty	-16.4
Online	Společné karty	-16.5
Online	Pozice u stolu	-16.7
Online	Počet nevyjádřených hráčů	-16.7
Offline	Počet nevyjádřených hráčů	-16.7
Offline	Pozice u stolu	-16.7
Online	Počet soupeřů ve hře	-16.7
Offline	Počet soupeřů ve hře	-16.7
Offline	Pot odds	-19
Online	Pot odds	-20.7

4.6.1 Jediná informace o hře

Nejdříve měly neuronové sítě k dispozici pouze jednu z možných informací o hře a velikost historie byla rovna nule. Jako nejdůležitější informace se ukázala být znalost mých vlastních karet 4.5. Hra těchto sítí probíhala stylem, že všichni hráči, kteří neznali své vlastní karty, se naučili nikdy dobrovolně nevkládat do hry žádné žetony. Kdyby hráč seděl u stolu a pouze vkládal do hry povinné sázky a nijak jinak se hry neúčastnil, jeho zisk by byl -25bb/100. V tabulce můžeme vidět, že zisk všech sítí byl lepší než -25bb/100 díky tomu, že většinou nikdo z hráčů nevsázel a podařilo se jim vyhrát nějaké žetony, když byli na pozici velkého blindu a dostali se až do fáze porovnání karet. Hráči, kteří své karty znali, se do hry přidali pouze v případě, pokud je považovali za profitabilní.

Tabulka 4.6: Ukázka špatných výsledků Offline učení

Druh učení	Doplňující informace	Zisk bb/100
Online	Pozice u stolu	67.16
Offline	Pozice u stolu	63.68
Online	Společné karty	53.86
Online	Pot Odds	50.65
Online	-	45.68
Online	Počet nevyjádřených soupeřů	44.14
Online	Akce soupeřů	40.31
Online	Počet soupeřů ve hře	35.27
Offline	Počet soupeřů ve hře	27.01
Offline	Počet nevyjádřených soupeřů	22.56
Offline	Pot Odds	-8.56
Offline	-	-10.40
Offline	Společné karty	-94.61
Offline	Akce soupeřů	-319.96

4.6.2 Dvě informace o hře - konec Offline učení

Z pozorování, že moje vlastní karty jsou nejdůležitější, byla vytvořena druhá generace hráčů. Každý z nich znal své vlastní karty a k tomu byla přidána jedna další informace. Počet předchozích tahů byl nastaven na 1.

Už v této fázi vyvstal problém se špatně fungujícím Offline učením. Celkový počet stavů byl mnohonásobně větší než dostupná paměť a neuronové sítě naučené na této podmnožině podávaly řádově horší výsledky než ty, které používaly Online učení 4.6.

Tato skutečnost kompletně zastavila pokrok neuronových sítí. Poker je hra s nulovým součtem [36]. Každý žeton, který jeden hráč vyhraje, musí druhý prohrát. Jestliže je jeden ze soupeřů velmi špatný, sítě se naučí hrát hlavně proti němu. To je přesný opak záměru této práce. Sítě musí hrát proti co možná nejlepším soupeřům, aby se přibližovaly k Nashově rovnováze.

Řešením je zavedení priority pro trénování sítí. Sítě se nebudou rovnoměrně střídat v tom, která se zrovna učí. Zlepšovat se bude vždy hráč, který je nejhorší. Tím omezíme možnost zneužívat špatnou hru soupeře.

4.6.3 Dvě informace bez Offline učení

Po odstranění Offline učení byly také zafixovány parametry pro neuronové sítě. Struktura sítí byla stejná pro všechny hráče - 1 skrytá vrstva s 50 neurony a rychlost učení rovna 0.0015.

Odebrání poloviny hráčů přineslo problémy s uváznutím v lokálním minimu. Jelikož byla množina hráčů moc malá, nejlepší síť byla vždy ta, která se učila jako poslední. Učení tedy funguje, ale je třeba zvětšit počet hráčů.

Tabulka 4.7: Důležitost druhé informace

Doplňující informace	Zisk bb/100
Společné karty	11.87
Akce soupeřů	2.97
Počet nevyjádřených soupeřů	-0.91
Počet soupeřů ve hře	-3.15
Pozice hráče	-3.81
Pot odds	-6.97

Pro každou kombinaci vstupů bylo vytvořeno celkem 6 neuronových sítí, které se učily a operovaly zvlášť. Shodné byly pouze vstupní informace. Výsledky v tabulce 4.7 odpovídají průměrným ziskům sítí se stejnými vstupy.

Vítězem tohoto kola je informace o společných kartách. Neuronové sítě si vytvořily vztah mezi kartami vlastními a společnými. Pokračovat ve hře se rozhodly pouze v případě, že jim společné karty pomohly vylepšit karetní kombinaci.

4.6.4 Výsledná neuronová síť

Další experimenty s postupným přidáváním informací provedeny nebyly. Rozdíly mezi přínosem jednotlivých informací o hře se stíraly a jejich spolehlivé nalezení by nijak neovlivnilo výsledek práce. Vygenerována byla tedy konečná množina 18 hráčů, kteří se střídavě učili přibližně týden s následujícími parametry:

- **Historie:** 1
- **Neuronů ve skryté vrstvě:** 100
- **Rychlost učení:** 0.01
- **Koeficient snížení explorace:** 0.02
- **Velikost dávky:** 64
- **Velikost paměti zkušeností:** 10000
- **Interval aktualizování cílové sítě:** 2500

4.7 Posouzení kvality hry nejlepší sítě

Nejlepší hráč z konečné množiny byl pomocí grafického prostředí otestován proti reálnému soupeři, pro kterého je poker koníčkem a věnuje se mu příležitostně. Jedná se o běžného hráče pokeru, který nijak nevyniká. Celkem bylo odehráno 2000 hand.

Vítězem této partie se stal reálný hráč s průměrným ziskem 1bb/100. Vzhledem k počtu odehraných hand není možné vyvodit závěr, že je neuronová síť nepatrně horší než rekreační hráč. Průměrný zisk by se po dalších 100 handách pravděpodobně znovu změnil.

S jistotou ale můžeme říci, že se neuronová síť naučila základní prvky pokeru. Umí posoudit sílu svých karet a podle ní vsázet. Naučila se, že startovní kombinace s vyšší hodnotou karet mají lepší šanci na výhru. Zjistila také, že ideální chvíle pro 'blaf' je tehdy, pokud soupeř neprojevuje žádný zájem o společný bank a pouze checkuje.

Na druhou stranu se hra neuronové sítě odvíjí převážně podle momentální situace. V případech, kdy doposud vyložené společné karty s těmi našimi netvoří žádnou výherní kombinaci, ale je velká šance, že se v dalším kole na stole objeví karta, která by mohla situaci vylepšit, má neuronová síť problémy tento stav správně vyhodnotit. Pokud tedy drží kombinaci karet 5678, tak ve hře nepokračuje, jelikož právě teď je její výherní kombinací pouze vysoká karta, i když šance na sestavení postupky v dalším kole je přibližně 30%.

Neuronová síť se tedy bez jediného externího zásahu naučila nejdůležitější prvky pokeru a její kvalita přibližně odpovídá úrovni hry rekreačního hráče.

Závěr

Hlavním cílem této práce bylo vytvoření umělé inteligece pro karetní hru poker s využitím posilovaného učení a neuronových sítí. K jeho dosažení bylo potřeba naimplementovat simulátor pokeru, kde proti sobě neuronové sítě mohou hrát a trénovat. Dále mělo být vytvořeno grafické rozhraní pro aplikaci, aby mohl uživatel porovnat síly s nejlepšími neuronovými sítěmi.

Celá práce byla naprogramována v jazyce Java. Výsledkem je plně funkční pokerový simulátor, který využívá více vláken pro urychlení simulace. Je také dostatečně obecný na to, aby bylo možné jeho znovupoužití přidáním jiného typu umělé inteligence. Grafické prostředí bylo vytvořeno pomocí technologie JavaFX.

Učení hráčů probíhá s pomocí algoritmu Q-učení, které využívá neuronové sítě jako Q-funkci. Stabilita učení je podpořena technikami jako je opakování zkušeností, využití cílové sítě nebo dvojité učení. Všechny úpravy algoritmu jsou otestovány s různými hodnotami parametrů, abychom nalezli ty, které podávají nejlepší výsledky.

V práci jsou také porovnány dva přístupy k učení. První, který nejdříve vygeneruje potřebná data a poté podle nich naučí hrát neuronovou síť, a druhý, který si generuje data za běhu a hned se z nich učí. Je nalezena hranice, kdy první zmíněný algoritmus přestane fungovat a není dále spolehlivým řešením.

Vstupy neuronové sítě, tedy informace o stavu hry, jsou rozděleny na dílčí části. Ty jsou testovány odděleně a poté je zhodnocena jejich relativní důležitost. Bylo zjištěno, že nejdůležitější informace ze hry jsou hráčovy a společné karty, zatímco ostatní tak podstatné nejsou.

Pro vygenerování nejlepšího hráče byla vytvořena množina celkem 30 hráčů, každý se svojí neuronovou sítí. Ti se postupně učili a jejich hra se zlepšovala. Po uplynutí jednoho týdne byla vybrána nejlepší neuronová síť z množiny a ta byla otestována partií proti reálnému hráči. Ukázalo se, že je schopná konkurovat rekreačnímu hráči a hraje na přibližně stejné úrovni. Projektům, jako je DeepStack[7] se ale nevyrovná, jelikož veškeré výpočty byly prováděny na standardním počítači bez specializovaného hardwaru, a tudíž byla rychlost

učení limitována. Dalším důvodem je celkový přístup zvolený v této práci, který spoléhá na schopnost sítě si se vším poradit.

Nedostatky zmíněné v kapitole Posouzení kvality hry nejlepší sítě jsou způsobeny snahou naučit jednoduchý model správnému chování ve složitém prostředí. Nabízejí se dvě varianty řešení tohoto problému. Buďto můžeme zvýšit výpočetní kapacitu modelu vložením dalších skrytých vrstev a neuronů, nebo zjednodušit prostředí předzpracováním dat a přidáním pokročilých informací o hře. Určitou část výpočtu tedy můžeme přesunout z neuronové sítě do deterministického algoritmu a tím uvolnit kapacitu modelu pro nalézání složitějších vztahů. Příkladem může být zakomponování pravděpodobnosti, že v příštím kole bude na stůl vyložena pro nás užitečná karta. Využití obou variant by mělo vést ke zlepšení neuronové sítě.

Do budoucna je také možné práci zkvalitnit zavedením priority, podle které by byly zkušenosti vybírány. Výzkumníci ukázali, že chytrý výběr zkušeností zlepšuje výsledky sítě [39]. Dalším možným zdokonalením programu je zrychlení celé simulace za pomoci grafické karty.

Poslední navrhovanou změnou je přidání informací o soupeřích a jejich tendencích. Neuronová síť v této práci nemá žádné informace o tom, kdo je její oponent. I kdyby se podařilo vytvořit umělou inteligenci, která hraje přesně podle Nashovy rovnováhy, neznamená to, že je to stragie nejprofitabilnější. Znamená to pouze, že není možné neuronovou síť porazit. Nejlepší strategie v aktuální situaci ale může být absolutně odlišná.

Literatura

[1] Deep Voice: Real-time Neural Text-to-Speech. *CoRR*, ročník abs/1702.07825, 2017. Dostupné z: http://arxiv.org/abs/1702.07825

[2] Waymo. Dostupné z: https://waymo.com/

[3] Autopilot | Tesla. Dostupné z: https://www.tesla.com/autopilot

[4] David Silver, C. J. M., Aja Huang: Mastering the Game of Go with Deep Neural Networks and Tree Search. 2016.

[5] DeepMind AlphaGo vs Lee Sedol. 2016. Dostupné z: https://gogameguru.com/tag/deepmind-alphago-lee-sedol/

[6] V Mnih, D. S. A. G. I. A. D. W. M. R., K Kavukcuoglu: Playing Atari with Deep Reinforcement Learning. 2013. Dostupné z: https://arxiv.org/pdf/1312.5602v1.pdf

[7] Moravčík, M.; Schmid, M.; Burch, N.; aj.: DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker. *ArXiv e-prints*, Leden 2017, 1701.01724.

[8] David Silver, C. J. M. A. G. L. S. G. v. d. D., Aja Huang: Mastering the game of Go with deep neural networks and tree search. 2016.

[9] Wikipedia: Poker — Wikipedia, The Free Encyclopedia. 2017, [Online; accessed 3-April-2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Poker&oldid=770876335

[10] Top 10 Poker Variations - Best Poker Games To Play. 2017. Dostupné z: https://www.cardschat.com/top-10-poker-games.php

[11] Pravidla sazeni | Poker pravidla. 2009. Dostupné z: http://www.pokerpravidla.cz/pravidla-sazeni/

[12] Vyherni kombinace | Poker pravidla. 2009. Dostupné z: http://www.pokerpravidla.cz/poker-vyherni-kombinace/

[13] Pravidla pokeru texas holdem | Poker-arena.cz. Dostupné z: http://www.pokerarena.cz/rubriky/poker/pravidla-pokeru-texas-holdem_5827.html

[14] Annual Computer Poker Competition. Dostupné z: http://www.computerpokercompetition.org/

[15] Johanson, M.: Measuring the Size of Large No-Limit Poker Games. *CoRR*, ročník abs/1302.7008, 2013. Dostupné z: http://arxiv.org/abs/1302.7008

[16] Poker Variance Calculator - Pokerdope. Dostupné z: http://pokerdope.com/poker-variance-calculator/

[17] Wikipedia: Artificial neural network — Wikipedia, The Free Encyclopedia. 2017, [Online; accessed 14-April-2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=775203700

[18] Heaton, J.: *Introduction to neural networks with Java*. Heaton Research, 2009.

[19] Jacobson, L.: Introduction to Artificial Neural Networks - Part 1. Dostupné z: http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7

[20] Wikibooks: Artificial Neural Networks/Activation Functions — Wikibooks, The Free Textbook Project. 2017, [Online; accessed 29-April-2017]. Dostupné z: https://en.wikibooks.org/w/index.php?title=Artificial_Neural_Networks/Activation_Functions&oldid=3193129

[21] Raschka, S.: Single-Layer Neural Networks and Gradient Descent. 2015. Dostupné z: http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

[22] Wikipedia: Softmax function — Wikipedia, The Free Encyclopedia. 2017, [Online; accessed 15-April-2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=775053054

[23] Nielson, M. A.: *Neural Networks and Deep Learning*. Deremination Press, 2015. Dostupné z: http://neuralnetworksanddeeplearning.com

[24] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[25] Wikipedia: Nash equilibrium — Wikipedia, The Free Encyclopedia. 2017, [Online; accessed 30-April-2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Nash_equilibrium&oldid=773945686

[26] Mobius, M. M.: Lecture VI: Existence of Nash equilibrium. 2008. Dostupné z: http://isites.harvard.edu/fs/docs/icb.topic449892.files/lecture6.pdf

[27] Volodymyr Mnih, D. S., Koray Kavukcuoglu: Human-level control through deep reinforcement learning. 2015.

[28] Wikipedia: Markov decision process — Wikipedia, The Free Encyclopedia. 2017, [Online; accessed 15-April-2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Markov_decision_process&oldid=773788606

[29] Melo, F. S.: Convergence of Q-Learning: A Simple Proof. Dostupné z: http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/ProofQlearning.pdf

[30] Matiisen, T.: Demystifying Deep Reinforcement Learning. 2015. Dostupné z: https://www.nervanasys.com/demystifying-deep-reinforcement-learning/

[31] Restelli, M.: Reinforcement Learning - Exploration vs Exploitation. 2015. Dostupné z: http://home.deib.polimi.it/restelli/MyWebSite/pdf/rl5.pdf

[32] Tim de Bruin, K. T. R. B., Jens Kober: The importance of experience replay database composition in deep reinforcement learning. 2015. Dostupné z: http://rll.berkeley.edu/deeprlworkshop/papers/database_composition.pdf

[33] Janisch, J.: Let's make a DQN. 2016. Dostupné z: https://jaromiru.com/2016/10/21/lets-make-a-dqn-full-dqn/

[34] van Hasselt, H.; Guez, A.; Silver, D.: Deep Reinforcement Learning with Double Q-learning. *CoRR*, ročník abs/1509.06461, 2015. Dostupné z: http://arxiv.org/abs/1509.06461

[35] Hasselt, H. V.: Double Q-learning. In *Advances in Neural Information Processing Systems 23*, editace J. D. Lafferty; C. K. I. Williams; J. Shawe-Taylor; R. S. Zemel; A. Culotta, Curran Associates, Inc., 2010, s.

2613–2621. Dostupné z: http://papers.nips.cc/paper/3964-double-q-learning.pdf

[36] Wikipedia: Zero-sum game — Wikipedia, The Free Encyclopedia. 2017, [Online; accessed 3-May-2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Zero-sum_game&oldid=775172495

[37] Deeplearning4j: Open-source, Distributed Deep Learning for JVM. Dostupné z: https://deeplearning4j.org/

[38] 7 Card Hand Evaluators. Dostupné z: http://archives1.twoplustwo.com/showflat.php?Number=8513906

[39] Schaul, T.; Quan, J.; Antonoglou, I.; aj.: Prioritized Experience Replay. *CoRR*, ročník abs/1511.05952, 2015. Dostupné z: http://arxiv.org/abs/1511.05952

Příloha **A**

Seznam použitých zkratek

hand Jedna celá hra od rozdání karet po zvolení vítěze

dealer Pozice udávající hráče, který se vyjadřuje v handě jako poslední

blind Povinná sázka

bb/100 Průměrný počet vyhraných velkých povinných sázek na 100 odehraných hand

GUI Grafické rozhraní aplikace

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF