Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Neuronové sítě pro ovládání robotických systémů

Bakalářská práce

Autor: Tereza Urbanová
Studijní obor: Informační management, IM3
Vedoucí práce: Ing. Karel Petránek

Hradec Králové duben 2016

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne 29. 4. 2016 Tereza Urbanová

Poděkování

Poděkování směřuje především vedoucímu bakalářské práce Ing. Karlu Petránkovi, za cenné rady při zpracování této práce a pomoc při realizaci její aplikace. Dále děkuji výzkumné organizaci IHMC a zejména M.Eng. Jerry Prattovi, Ph.D za možnost spolupráce.
Ráda bych poděkovala i své rodině, přátelům a všem, kteří mi při zpracování bakalářské práce pomohli.

Anotace

Cílem bakalářské práce je vytvořit systém pro ovládání jednoduchého robotického zařízení pomocí neuronové sítě. První část bakalářské práce popisuje neuronové sítě a možnosti jejich použití ve spojení s robotickými zařízeními. Dále jsou vysvětleny základní principy a elementy využívané při učení neuronových sítí. Praktická část obsahuje analýzu, postup implementace a testování dvou různých způsobů využití neuronových sítí pro ovládání jednoho zařízení. Robotické zařízení je představeno jednoduchým inverzním kyvadlem, které je základním stavebním prvkem většiny humanoidních robotů. Ve výsledku dokáže neuronová síť takové kyvadlo rozpohybovat a přiblížit rovnovážné labilní poloze. Výzkum a samotná implementace řešení proběhla ve spolupráci s vědecko-výzkumnou organizací Florida Institute for Human and Machine Cognition (IHMC).

Klíčová slova

Neuronové sítě, robotická zařízení, inverzní kyvadlo, učení posilováním

Annotation

Title: Neural Networks for Robot Control

The goal of this thesis is to create a neural network based system for controlling a simple robotic device. The first part of this thesis describes the theoretical basis of neural networks and discusses their usage in conjunction with robotic devices. Various approaches to training neural networks are discussed, including base principles common to all training methods. The second part presents the analysis, implementation and testing of two different approaches for controlling a simple robotic device. The device being controlled is a simple inverse pendulum which is a basic building block of most humanoid robots. The trained networks are able to control the pendulum and bring it very close to its unstable equilibrium point.

Keywords

Neural networks, robotics, inverse pendulum, reinforcement learning

Obsah

1. Úvod

Umělé neuronové sítě a robotická zařízení jsou v současné době často skloňované pojmy. Neuronové sítě nedávno překonaly člověka ve dvou důležitých milnících – rozpoznání objektů v obraze [1] a hraní hry Go [2]. Jedná se o úlohy, u kterých se experti donedávna domnívali, že potrvají desetiletí, než v nich počítače budou schopny člověka překonat. Humanoidní roboti zažívají zvýšený zájem díky soutěži Darpa Robotics Challenge (DRC) [3], ve které humanoidní roboti-záchranáři soutěží v provádění úloh jako je řízení automobilu, odstranění překážek či chůze po nerovném terénu.

Ačkoliv neuronové sítě i robotika se vzájemně doplňují, většina humanoidních robotů v současné době neuronové sítě nevyužívá. Proto si tato práce klade za úkol vytvořit spojovací článek mezi těmito dvěma oblastmi. Cílem je prozkoumat možnosti ovládání robotických zařízení pomocí regulátorů založených na neuronových sítích. V práci jsou využity nejnovější poznatky z oblasti neuronových sítí. Pro ovládání jednoduchého robotického zařízení je použito hluboké učení posilováním (Deep Reinforcement Learning), což je metoda, která byla nedávno použita pro hru Go, v níž počítač poprvé porazil bývalého světového velmistra.

Práce vznikala ve spolupráci s institutem IHMC v rámci čtyřměsíční vědecko-výzkumné stáže. IHMC je americká nezisková organizace zabývající se umělou inteligencí, vývojem humanoidních robotů, robotických exoskeletů a systémů propojujících technologii a člověka. V loňském roce robotický tým IHMC obsadil druhou příčku v soutěži DRC s humanoidním robotem Atlas.

V první části se práce věnuje teoretickým aspektům neuronových sítí a robotických zařízení. Je krátce shrnuta historie vývoje neuronových sítí a robotiky, diskutován neurofyziologický základ učení a podoby neuronových sítí a rozebrány teoretické aspekty učení neuronových sítí a ovládání robotických zařízení.

Praktická část práce je věnována učení neuronových sítí pro ovládání robotického zařízení, jednoduchého inverzního kyvadla. Cílem trénování je vyvážit kyvadlo v rovnovážné labilní poloze pomocí neuronových sítí. Jsou představeny a zhodnoceny metody trénování sítí napodobením existujícího regulátoru a na základě učení posilováním.

2. Historie robotů a neuronových sítí

Významným mezníkem v robotickém odvětví byl rok 1921, kdy měla v Praze premiéru divadelní hra Karla Čapka R.U.R., Rossum´s Universal Robots, která českého autora světově proslavila a robotům dala jejich název. Hra pojednává a pokolení robotů, vyráběných na zakázku, které postupně vytlačí lidstvo. Slovo robot vymyslel jeho bratr Josef Čapek a je považováno za běžné mezinárodní slovo, jako jedno z mála českého původu. [4]

2.1. 40. léta

Za zrodem oboru neuronových sítí stojí práce Warrena McCullocha a Waltera Pittse z roku 1943 [5], kteří vytvořili základní matematický model neuronu. Číselné hodnoty parametrů v modelu zvolili bipolární (tj. z množiny {-1,0,1}). Výstupem neuronu v tomto modelu je 0 nebo 1 podle toho, zda vážená suma vstupních signálů je větší, než hodnota prahu. Ve svém článku vyzdvihli přínos neuronových sítí a fakt, že i jednoduché sítě dokáží v principu aproximovat kteroukoliv aritmetickou nebo logickou funkci.

Ani jeden z autorů nepočítal s možností praktického využití vytvořeného modelu, avšak jejich článek měl velký vliv na další výzkum. Inspiroval se jím například zakladatel kybernetiky Norbert Wiener při studiu analogií mezi výpočetní technikou a fungováním nervové soustavy. Své myšlenky shrnul do knihy Kybernetika aneb Řízení a sdělování u organismů a strojů, která vyšla v roce 1948 a brzy se stala světovým vědeckým bestsellerem. [6] Ve své knize se věnoval i samoregulačním mechanismům. [7]

Dále John von Neumann při práci na americkém projektu elektronických počítačů nastínil studie počítačů inspirovaných činností mozku. [8] [9] Modely napodobovaly biologické principy, např. genetické algoritmy, neuronové sítě, metody učení založené na modelech či metody využívání znalostí. John von Neumann vypracoval automat (počítač), jakožto model biologického systému, jež je schopný sebe-produkce. [10]–[12]

2.2. 50. léta

V polovině 20. století se v robotickém odvětví začínají objevovat složitější praktické aplikace. V roce 1954 George Devol, prezident Devol Research a majitel společnosti Unimation,

zaregistroval svůj první patent týkající se robotiky, kterých později bylo nad čtyřicet. Jednalo se o patent na první ovládané digitálně programovatelné robotické paže, jež představoval základ moderní robotiky. V roce 1961 se spolu s J. Engelbergerem a universitou Columbia University U.S.A. zapříčinili o vznik prvního průmyslového robota UNIMATE u firmy General Motors, jež představuje předchůdce automatizačních strojů pro montážní linky po celém světě. [13], [14]

Pro neuronové sítě jsou padesátá léta považována za první zlatý věk. V roce 1957 zobecnil Frank Rosenblatt McCullochuv a Pittsonův model neuronu pro reálný číselný obor parametrů a navrhl nejjednodušší model dopředné neuronové sítě, perceptron, a odpovídající učící algoritmus. [15] Tento učící algoritmus pro daná trénovací data nalezne po konečném počtu kroků vektor vah, pokud existuje, nezávisle na výchozím vektoru stanoveném na počátku trénování. [11] [16]

Frank Rosenblatt využíval perceptrony k rozpoznávání vzorů a dokázal, že pokud existují váhy, které řeší zadaný problém, učící pravidlo k nim konverguje. Své výsledky v oblasti neurovýpočtů Rosenblatt shrnul v knize Principles of Neurodynamics [17], ve které rozebírá použití perceptronů na rozpoznávání vzorů v obraze a navrhuje rozšíření v podobě zpožděných neuronů a zpětných vazeb, jež umožňují perceptronu reprezentovat časové závislosti. Obě použití jsou stěžejní pro robotiku – rozpoznání obrazu pro analýzu vstupů ze senzorů, časové závislosti pro koordinaci pohybů a rovnováhu. V letech 1957–8 se Rosenblatt účastnil sestavení neuropočítače Mark I. Perceptron, který byl realizovaný na von Neumannovské architektuře počítače. Mark I. Perceptron byl navržen pro rozpoznávání znaků, které byly promítány na světelnou tabuli. Vstupem do perceptronu byla matice 400 obrazových bodů reprezentujících intenzitu snímaného světla a úkolem perceptronu bylo klasifikovat, o jaký znak se jedná.

Neuropočítač byl úspěšný, a proto i neurovýpočty, alternativa ke klasickým výpočtům, se staly novým předmětem výzkumu. Frank Rosenblatt je tak mnohými odborníky považován za zakladatele tohoto nového oboru. [11]

Nedlouho po objevení Roseblattova perceptronu v roce 1960 vyvinuli Bernard Windrow a Ted Hoff na Stanford University další typ neuronového výpočetního systému, který nazvali ADALINE (**ADA**ptive **LIN**ear **E**lement) a vybavili jej novým učícím pravidlem, které se používá dodnes [18]. Neuron ADALINE je základním prvkem modelu MADALINE (**M**ultiple

Adaline), který je strukturálně totožný s vícevrstvým perceptronem, avšak místo algoritmu pro učení perceptronu používá ADALINE [11]. Zatímco u perceptronu jsou váhy modifikovány přičítáním či odčítáním vstupního vektoru od vektoru vah, u ADALINE je výstup sítě předán do aktivační funkce a pro nastavení vah je použit rozdíl mezi výstupem aktivační funkce a očekávaným výstupem.

Na rozhraní 50. – 60. let pokračuje rozvoj neurovýpočtů, vznikají nové modely neuronových sítí a jejich implementace. Poté, co bylo zjištěno, že Rosenblattův perceptron dokáže rozlišit pouze lineárně oddělitelné vstupy, je výzkum v oblasti neuronových sítí až do 80. let utlumen a vnímán jako neperspektivní [19], především díky knize *Perceptrons* (Marvin Minsky a Seymour Papert)[20]. Kniha přináší důkaz, že perceptron není schopen aproximovat funkci vylučovací disjunkce (XOR). Tento důkaz spolu s faktem, že učící algoritmus pro vícevrstvé sítě nebyl v té době znám, vedl k omezení použití neuronových sítí ve všech oblastech, včetně robotiky. Dokonce se v mnoha případech přestaly používat neuronové sítě v oblastech, ve kterých do té doby byly úspěšně nasazovány. [11] [12]

Naopak robotika v 70. letech prosperovala a průmyslové roboty se rozšířily, především v oblasti rizikových prací, jako je svařování, manipulace s nebezpečnými či těžkými předměty, výškové práce a v dalších případech, kde nasazení lidských sil je nemožné nebo ohrožující zdraví. V roce 1968 přišel na trh Stanford Research Institute s první mobilní robotkou Shakey vybavenou viděním. Jedná se o první projekt, kdy byl robot schopný uvažování (vnímat své okolí) a zároveň provádět fyzické akce. Shakey se uměla pohybovat z místa na místo, byla schopná manuální činnosti (zapnout/vypnout světla, otevřít/zavřít dveře, chodit po různých předmětech), orientovat se v prostředí a vyhledat nebo přemístit předmět v okolí. Vývoj Shakey měl velký dopad na polích robotiky a umělé inteligence. [21]

2.3. 80. léta až současnost

80. léta přinesla pokrok v robotice i neuronových sítích. V těchto letech se roboty běžně vybavovaly viděním, čidly hmatu, foto-optickými čidly a ultrazvukovými detektory vzdálenosti.

K obrození zájmu o neuronové sítě přispěl v roce 1982 i světově uznávaný fyzik John Hopfield se svými díly [22] a [23]. Přišel s modelem neuronové sítě a propojil ji s fyzikálními modely

magnetických materiálů.[22] Tzv. Hopfieldova síť se skládá z n neuronů, kde každý je vstupem i výstupem a všechny jsou spolu vzájemně propojené. Řídí se Hebbovým zákonem. Hopfieldovy sítě mají asociativní neuronovou paměť, dokáží si pamatovat hodnoty na základě daného klíče. [24]

John Hopfield svůj výzkum neuronových sítí šířil mezi další vědce prostřednictvím přednášek po celém světě. O neuronové sítě se tak znovu začalo zajímat velké množství vědců, technologů a matematiků. Na základě jeho snahy vznikla tzv. „PDP skupina“ (Parallel Distributed Processing Group) badatelů, jenž publikovali své výsledky do sborníku Davida Rumelharta a Jamese McCleallanda [25], který se zasloužil o dosažení opětovného zájmu o neuronové sítě. Zde publikovali svůj článek i tři autoři D. E. Rumelhart, G. E. Hintona a R.J. Williamse [26], ve kterém charakterizovali učící algoritmus zpětného šíření chyby zvané backpropagation pro vícevrstvou neuronovou síť (viz podkapitola 5.2.1.2). Tímto vyřešili problém, který zastavil rozvoj neuronových sítí od 60. let. Ač se později dokázalo, že uvedený algoritmus byl pouze znovuobjevením již existujícího algoritmu představeného o pár let dříve, stala se Rumelhartova a Hintonova varianta nejpoužívanější metodou pro učení neuronových sítí. [11] [27]

V robotickém odvětví se na konci 20. století rozšířilo využití robotických zařízení ve zdravotnictví, konkrétně při náročných operací. Jako první robotický systém v oblasti chirurgie byl předveden v roce 1983 a pojmenován jako ARTHROBOT. O pět let později byl sestaven robot Puma 560, který se používal při biopsii mozku. V roce 1995 byl představen chirurgický robotický systém Zeus pro tzv. minimální invazivní chirurgii. [13]

V roce 1997 byl agenturou NASA vypraven první robot do vesmíru. Byl jím robot Sojourner (Cestovatel), jenž přistál na Marsu 4. července a prozkoumával složení povrchu planety Mars asi 3 měsíce a po celou dobu komunikoval se zemí. [13], [28], [29]

Z tohoto období jsou položeny základy mezinárodním organizacím Federation of International Robot-soccer Association (FIRA) a RoboCup, které se obě soustředí na soutěže robotů ve fotbalu. Cílem obou organizací je urychlení výzkumu v oblasti robotiky. RoboCup má dokonce za cíl, aby robotický tým porazil lidský tým a to mistra světa v roce 2050 v regulérním fotbalovém zápase. [13]

V roce 2000 přišla firma Honda se svým humanoidním robotem ASIMO (Advanced Step in Innovative Mobility), viz Obrázek 1. ASIMO je humanoidní robot s intuitivním rozhraním schopný fungovat ve skutečném lidském prostředí. Porovnání: model z roku 2000 byl vysoký 120 cm, vážil 52 kg a jeho rychlost chůze byla kolem 1,6 km/h, nový model ASIMA z roku 2011 má odlišné míry - měří 130 cm, váží 48 kg (110 lb), jeho rychlost chůze je kolem 2,7 km/h a rychlost běhu až 9 km/h.

ASIMO robot je schopen reagovat na lidské jednání a řeč. V současnosti jej lze dokonce ovládat pouhou myšlenkou, pomocí helmy, která vysílá radiové pokyny. [30] Robot má zabudované dvě kamery, které zachycují vizuální informace a slouží jako oči. Pomocí nich umí rozpoznávat pohyby několika objektů, umí určit vzdálenost a směr, následovat osobu, či čelit někomu, kdo se přiblíží. [31]. Asimo umí reagovat na své jméno a rozeznávat zvuky, hlasové příkazy, lidská gesta. Reaguje přikývnutím, či verbální odpovědí v různých jazycích. [32], [33] Rozpoznání obrazu a mluvené řeči by mohlo být realizováno pomocí neuronových sítí, které v daných oblastech momentálně dosahují nejlepších výsledků, avšak Asimo takových možností zatím nevyužívá. [34]

Obrázek 1 - Asimo robot [33]

V roce 2011 NASA vyslala do vesmíru první humanoidního robotího astronauta s názvem Robonaut R2B. [13] Na rozdíl od dosavadních vesmírných robotů (robotické paže, jeřáby, průzkumní vozítka) je navržený tak, aby dokázal pracovat po boku astronautů. [35]

3. Neurofyziologický vzor umělých neuronových sítí

3.1. Poznatky z neurofyziologie

Neurofyziologické poznatky slouží jako zdroj inspirací a na základě nich byly vytvořeny zjednodušené matematické modely, díky nimž fungují neurovýpočty a praktické úlohy z oblasti umělé inteligence. Schopnost vnímat podněty z okolního světa, vědět o svém vlastním stavu a umět na něj vhodně reagovat je jednou ze základních vlastností všech živých organismů. U člověka a dalších obratlovců tuto funkci zajišťuje nervový systém a mozek. [11] [12]

3.2. Nervová soustava

Nervová soustava živých organismů slouží k zachycení a zpracování podnětů, které působí na organismus a zprostředkovává i vztahy mezi jeho částmi. Také zajišťuje vhodnou odezvu na jeho vnitřní stavy a vnější podněty a nervové řízení pro přenos informací.

V rámci nervové soustavy existují jednotlivá čidla (receptory), z nichž se šíří vzruchy a umožňují přijímat veškeré podněty ve směru k nervovým buňkám. Ty signály zpracovávají a odesílají k příslušným organům (efektorům). Dostávají se až do nejvyššího řídícího centra nervového systému, do mozkové kůry. [11], [36]

3.3. Lidský mozek

Lidský mozek se skládá z několika set až tisíc miliard různých druhů stavebních kamenů, z nichž čtyřicet až sto miliard jsou neurony. Neuron může mít až deset tisíc vstupů spojených synapsemi s výstupy jiných neuronů. Celá struktura mozku i vzájemná propojení neuronů se mohou měnit dle potřeby.

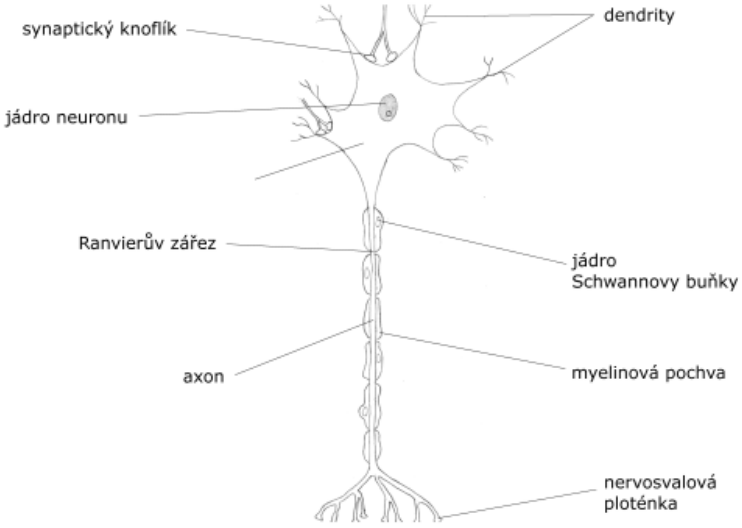
V mozku se nachází centra základních funkcí i složitějších funkcí. Mezi složitější funkce patří vnímání, rozpoznávání, logické rozumové funkce, vědomí, city a emotivní funkce. Některé části mozku nejsou doposud dostatečně prozkoumány. [12], [37]

3.4. Biologický neuron

Základní stavební jednotkou mozku je nervová buňka, neuron. Neurony jsou soustředěny především v centrální nervové soustavě, tedy v mozku a míše a jejím blízkém okolí. Neurony jsou vysoce specializované buňky určené k přenosu, zpracování, uchování a využívání informací, čímž umožňují organismu reagovat na podněty z prostředí. Cílem umělých neuronových sítí je tyto funkce napodobit. [11] [12] [37]

3.4.1. Struktura neuronu

Neuron je vzhledově, funkčně i vnitřní strukturou uzpůsoben přenosu signálů, viz Obrázek 3**.** Nejvýraznější částí neuronu je jeho tělo (soma). Z těla běžně vystupují dva druhy přenosových kanálů, axon a dendrity.



Obrázek 2 - Biologický neuron [36]

Axon znázorňuje funkční výstup z neuronu, na svém konci je výrazně rozvětven do terminálů, které se dotýkají výběžků dendritů jiných neuronů a zde dochází k přenosu informací. Axon je většinou pouze jeden u každého neuronu, na rozdíl od dendritů, kterých je velké množství.

Dendrity představují funkční vstup do neuronu, jsou krátké, úzké a rozvětvené do výběžků – dendritických trnů, které vedou vzruchy k buňce. U každého neuronu jich je velké množství. Synapse vznikají v místech, kde se dotýkají terminálová vlákna axonů s výběžky dendritů jiných neuronů. Dokáží adaptivně přizpůsobovat potřebám a vzniklým situacím. Jejich

průchodnost se mění během procesu učení a zapomínání. Paměť a učení jsou spjaty s tzv. synaptickou plasticitou. Vývoj nervového systému trvá, během něj se neurony učí pomocí eliminací a posilování synapsí. Při narození lze hovořit o chaotičnosti sítě. Stejný princip se používá u umělých sítí, kdy parametry sítě jsou nejprve nastaveny náhodně a v průběhu učení jsou posilovány či oslabovány na základě chybové funkce. [11], [27], [36]

3.4.2. Funkce neuronu

Specifická funkce neuronu představuje činnost axonů s dendrity a buněčné membrány, které vytváří a přenáší nervové vzruchy. Vnitřní systém neuronu si předává tyto vzruchy realizované fyzikálně-chemickými změnami mezi synaptickými částmi. [12]

4. Umělé neuronové sítě

4.1. Analogie umělé a biologické neuronové sítě

Jak biologické, tak umělé neuronové sítě je nutné nejprve naučit, aby mohly vykonávat požadovanou úlohu. Biologické neuronové sítě jsou velmi rozsáhlé a komplexní a při současném stavu technologie není možné v podobném rozsahu imitovat lidský mozek. Lze však alespoň napodobit dílčí funkce mozku vhodným zjednodušením biologických principů. Tyto zjednodušené modely umožňují řešit i komplexní praktické problémy, které se konvenčně nedají jednoduše řešit vůbec nebo s omezením. V některých případech je možné dosáhnout i lepších výsledků než kterých dosahují biologické neuronové sítě, například z hlediska odolnosti vůči vnějším vlivům, spolehlivosti, rychlosti apod. Ve většině případů se umělé neuronové sítě biologickými sítěmi inspirují, ale většinou se nejedná o přímý model a liší se především principy, na kterých jsou založeny. [12]

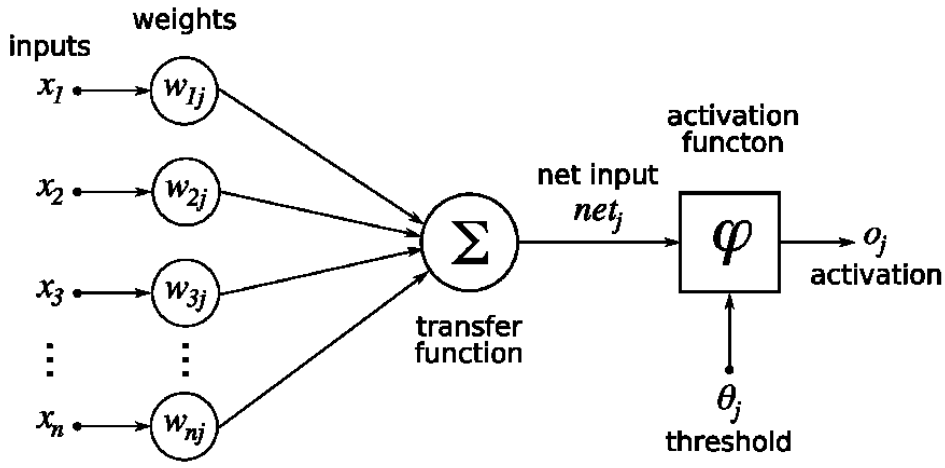
Umělé neuronové sítě jsou prostředky k dosažení výsledků, které lze konvenčními algoritmy dosáhnout jen obtížně. Vstupem do umělé neuronové sítě je signál reprezentovaný sadou reálných čísel. V případě komplikovaných signálů je možné extrahovat významné vlastnosti a ty použít jako vstup. [11], [27].

4.2. Matematický model neuronu

Matematický model neuronu je základní stavební jednotkou umělých neuronových sítí (UNS), podobně jako neurofyziologický neuron je základem biologické nervové soustavy. Model neuronu u UNS je tvořen matematickými funkcemi a různou topologií modelu konkrétního neuronu. Funguje podobně jako princip předávání informace, která probíhá biologickým neuronem. [27]

Modelů neuronu existuje hodně, ať už jednoduché, které používají nespojité funkce, až po velmi složité, které popisují všechny detaily chování neuronu. Nejrozšířenějším a nejpoužívanějším modelem neuronu je tzv. formální neuron (viz Obrázek 3). Jedná se o první reálně použitelný matematický model neuronu. Publikovali jej W. S. McCulloch a W. Pittson [5]. Jejich model našel velké uplatnění, a ač byl mnoha autory několikrát modifikován a vylepšován, je dodnes

základem většiny umělých sítí. Někdy je proto nazýván McCulloch-Pittsonův model neuronu. [12], [27]



Obrázek 3 - Formální model neuronu [38]

Části formálního modelu neuronu:

- **Inputs – vstupní vektor** - x_i (pro $1 \leq i \leq n$) představují elementy vstupního vektoru
- **Weights – vektor synaptických vah** – prvky w_{ij} (pro $1 \leq i \leq n$) představují elementy vektoru synaptických vah j-tého neuronu. Díky vahám je neuron schopen selektivně zvýhodňovat či potlačovat hodnoty na vstupu. Během procesu učení jsou hledány optimální hodnoty w_{ij}, aby pro vstupy z trénovací množiny výstup neuronu odpovídal očekávaným hodnotám.
- **Transfer function –** funkce kombinující váhy a vstupy neuronu, obvykle se používá vážený součet.
- **Prahová hodnota** – hodnota prahu Θj j-tého neuronu určuje aktivitu neuronu. Neuron je aktivní, je-li vážená suma vstupu větší než práh a naopak neuron je neaktivní, pokud je vážená hodnota vstupů menší, než práh.
- **Activation function – přenosová (aktivační) funkce** – obecně nelineární, spojitá nebo diskrétní funkce různého tvaru
- **Vnitřní potenciál neuronu** – stav j-tého emitovaného neuronu [11], [39]

4.3. Výhody a nevýhody umělých neuronových sítí

Mezi výhody umělých neuronových sítí patří schopnost učit se, a to bez nutnosti explicitní znalosti algoritmu řešení, a schopnost generalizovat, tedy správně si zařazovat poznatky, které se nenacházely v trénovací sadě.

Mezi nevýhody patří náročnost na volbu parametrů sítě (počty vrstev, počty neuronů, aktivační funkce, velikost optimalizačního kroku), sklony k přetrénování a vysoká výpočetní náročnost při trénování rozsáhlých sítí.

Neuronové sítě je vhodné používat tam, kde je cílem aproximovat funkční hodnoty, při kontrole a řízení fyzikálních veličin, v případě neznalosti algoritmu řešení, při matematickém popisu, či tam, kde je potřeba simulovat intuici člověka.

Výhodou je možnost implementovat neuronové sítě hardwarově. Existují integrované obvody navržené jako stavební prvky neuronových sítí, jež se používají například v oblastech rozpoznávání vzorů a robotice. [27]

5. Principy učení

5.1. Učení robotických zařízení

Pro učení robotických zařízení jsou používány i jiné způsoby učení, než jen pomocí neuronových sítí. Cílem učení je zajistit, aby byl robot schopný konat různorodé úkoly v prostředí, ať už reálném nebo virtuálním, a zaručit, aby robotická zařízení dělala méně chyb s rostoucími zkušenostmi.

Při učení robotického zařízení je nutné počítat s propojením na prostředí. Pomocí senzorů robot přijímá informace a pomocí efektorů nebo aktuátoru působí na okolní prostředí.

Každý robot, který je schopný se učit, musí obsahovat vnitřní model zadaných úkolů, okolností a prostředí, ve kterém bude fungovat. Také musí být nějak hodnocen pro zpětnou vazbu, zda se činnost naučil dobře. [40]

5.2. Učení neuronových sítí

Vývoj neuronových sítí a jejich vnitřního propojení funguje díky plasticitě synapsí, konkrétně díky jejich eliminaci, či posilování. S tím je spojena paměť a učení, a to jak u biologických, tak umělých neuronových sítí. Proces učení lze definovat jako modifikaci prahů a synaptických vah dle zvoleného algoritmu učení.

V rámci procesu učení je cílem vybrat vhodné atributy ze vstupních dat a nastavit parametry umělých neuronových sítí tak, aby odchylka mezi požadovaným a skutečným výstupem při odezvě na soubor testovacích příkladů byla co nejmenší. Vstupní váhy představují paměť neuronu. Proces probíhá iterativně z předkládaných příkladů.[27]

Adaptace neuronových sítí lze zobecnit i na dopředné vrstevnaté neuronové sítě. Výhodou NS je možnost nalezení transformace i u analyticky obtížně řešitelných úloh či neřešitelných úloh, je-li k dispozici dostatečné množství příkladů.

Učení neuronových sítí se dělí i podle četnosti učení. Existuje jednorázové učení, kdy je cílem dosáhnout v jednom učícím cyklu co nejlepšího výsledku. Druhým způsobem je opakované

učení, kdy jsou neuronové síti opakovaně předkládány vzory a snahou je dosáhnout minimální odchylky od požadovaného výstupu. Vzory se mohou předkládat různými způsoby. Jedním ze způsobů je tzv. hluboké učení posilováním, které je použito v praktické části práce (viz kapitola 7) a u něhož se zasahuje i do průběhu učebního procesu průběžným vyhodnocováním. Základními zákony učení jsou Hebbův zákon učení [24], zákon kompetice [41] a chybové učení [42].

Mezi další způsoby učení patří on-line učení, kdy nové příklady z prostředí jsou přímo předloženy síti a zapomenuty, jakmile je síť zpracuje. Opakem je off-line učení, při němž se sada příkladů síti předkládá opakovaně. [27]

5.2.1. Učení s učitelem

Při učení s učitelem (supervised learning) [39] musí být dopředu známé požadované výsledky, které se v průběhu učení porovnávají s výstupy sítě. Síť během učení nastavuje váhy a prahy tak, aby se její výstupy blížily k požadovaným hodnotám. Při aplikování této metody je nutné mít k dispozici množinu vzorů (učící množinu) ve tvaru dvojic $(\overrightarrow{x}, \overrightarrow{y})$ Během procesu učení jsou na vstup neuronové sítě přiváděny vstupní vektory.

V rámci procesu učení se hledá minimum chybové funkce. Často používanou chybovou funkcí, která je využita i v praktické části, je střední kvadratická chyba:

$$E = \frac{1}{m}\sum_{k=0}^{m}(y'_k - y_k)^2$$

Rovnice 1 – Střední kvadratická chyba

kde y'_k je očekávaný vstup sítě, y_k reálný výstup sítě, a m velikost trénovací sady.

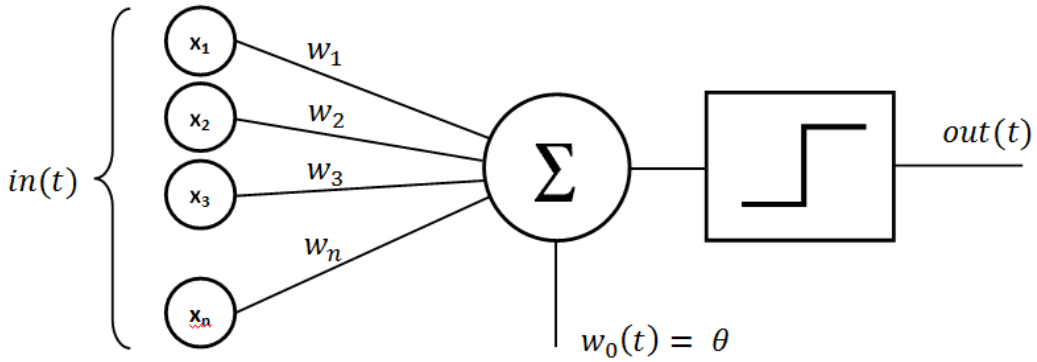
Během procesu učení je potřeba stále sledovat, zda má celková chyba sítě klesající tendenci. Je proto nutné sledovat závislost chyby na počtu iterací. Učení probíhá správně, když průběh chybové funkce je hladký a především v prvních iteracích má velmi strmý pokles. Díky tomu se dá ověřit dobrá volba velikosti trénovacího souboru a dobře zvolené parametry učení. Za nevhodné průběhy učení je považován plochý průběh při volbě příliš velkého trénovacího

vzorku či špatně zvolených parametrů učení a oscilační průběh chybové funkce při příliš malém trénovacím souboru, či špatně zvolené rychlosti učení.

Tento způsob učení využívají neuronové sítě, jako jsou např. Perceptron, Adaline, Madaline a vícevrstvé sítě se zpětným šířením chyby (backpropagation). [27] Také se tento způsob učení používá při učení robotů. Při tomto způsobu dává učitel robotu okamžitou zpětnou vazbu o očekávané akci v každém stavu. [40]

5.2.1.1. Perceptron

Perceptron je nejjednodušší model dopředné neuronové sítě a tvoří základní stavební prvek složitějších sítí, viz Obrázek 4. Jeho využití je omezené, častěji se využívá vícevrstvý perceptron, jelikož perceptron s jedním neuronem umožňuje pouze binární klasifikaci. [12], [43]



Obrázek 4 - Model jednovrstvého perceptronu (zdroj: wikipedia)

Základní stavební prvek jednoduchého perceptronu má několik binárních vstupů a jeden binární výstup. Jedná se o nezávislé paralelně pracující neurony. Každý neuron zvlášť transformuje vstup na výstupní hodnotu, nezávisle na ostatních. Výstupem perceptronu je hodnota.

$$\mathbf{y = f\left(\sum_{i=1}^{n} w_i x_i + \theta\right)}$$

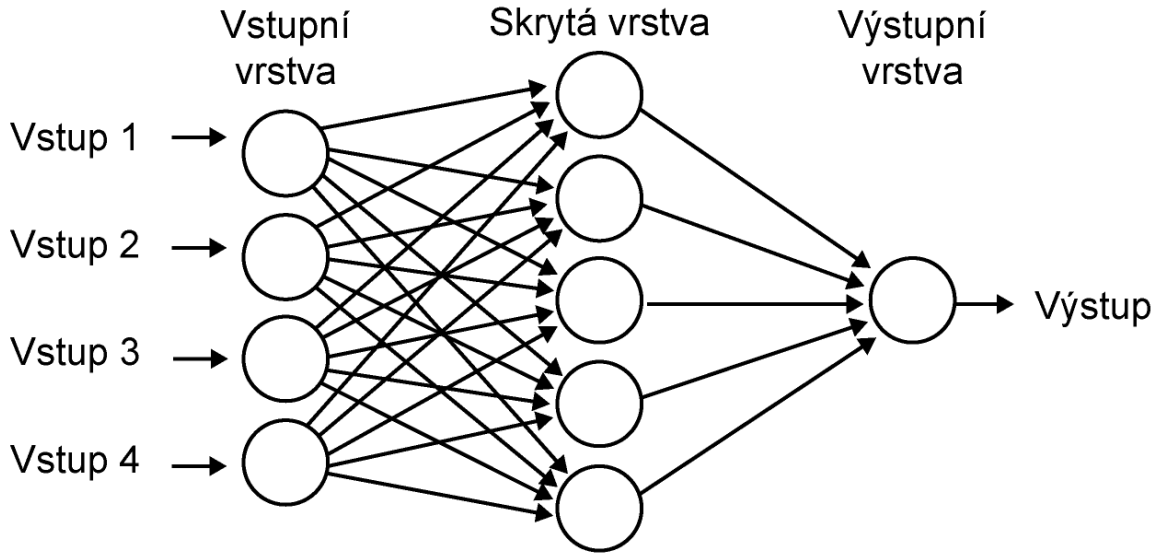
Rovnice 2 - Výstupní vektor jednoduchého perceptronu

Kde f je aktivační funkce (u původního modelu se jedná o prahovou funkci), w_i váha pro vstup x_i a θ hodnota bias.

Při učení neuronových sítí je potřeba namodelovat rozhodovací model a správně zvolit jednotlivé váhy a prahovou hodnotu.

Jak již bylo zmíněno, perceptron lze skládat a vytvořit tak síť, která bude mít několik vrstev či více perceptronů v jedné vrstvě (viz Obrázek 5). Vícevrstvý perceptron umožňuje klasifikaci do více tříd kombinací několika jednoduchých perceptronů do složitějších vícevrstvých modelů s více výstupními neurony. Vícevrstvé perceptrony nevyžadují lineární oddělitelnost vstupních dat, avšak musí být zachována oddělitelnost pomocí nelineární funkce.

Kombinací perceptronů vzniká hierarchický model zpracování informací. V první vrstvě dělá perceptron jednoduchá rozhodnutí vážením vstupních hodnot. V druhé vrstvě dělá perceptron rozhodnutí dle vah výsledků z první vrstvy rozhodování. Díky tomu může perceptron v druhé vrstvě provádět složitější a abstraktnější rozhodnutí, než perceptron v první vrstvě. Ještě složitější rozhodnutí může perceptron řešit v následujících vrstvách. Tímto způsobem se mnohovrstvá síť perceptronů může zapojit do řešení sofistikovanějších rozhodování.



Obrázek 5 - Model vícevrstvého perceptronu [43]

Jednoduchý perceptron má vždy jen jeden výstup. Výstupy z prvních vrstev perceptronů jsou zde považovány jako vstupy do další vrstvy perceptronů. [43]

5.2.1.2. Strategie zpětného šíření chyby (Backpropagation)

Nejpoužívanější model neuronové sítě je vícevrstvá neuronová síť, která se učí podle algoritmu zpětného šíření chyby [26]. Je používána ve zhruba 80% všech aplikací neuronových sítí, protože se zvládá učit i složité multidimenzionální mapování vstupů na výstupy. [44] Cílem algoritmu Back-propagation je efektivním způsobem vypočítat derivaci (gradient) neuronové sítě podle jednotlivých vah. Tato derivace může být následně použita pro optimalizaci numerickými metodami, např. metodou největšího spádu. [45] Výpočet derivace je založen na aplikaci řetězového pravidla pro složené funkce. Neuronovou síť je možné si představit jako postupnou aplikaci dílčích funkcí, které jsou reprezentovány jednotlivými vrstvami, na něž je na závěr aplikována chybová funkce:

$$E(L_n(w_n, L_{n-1}(W_{n-1}, \dots L_1(w_1, x) \dots))$$

kde x je vstupní vektor,w_i matice vah neuronů i-té vrstvy, $L_i$výstup i-té vrstvy a E chybová funkce.

Výpočet derivace poté vede k řetězovému pravidlu:

$$\frac{dE}{dw} * \frac{dL_n}{dw} * \dots * \frac{dL_1}{dw}$$

Díky skládání funkcí je nejprve vypočítána derivace chybové funkce, poté derivace poslední vrstvy. Derivace vah v první vrstvě je vypočítána až jako poslední, algoritmus tedy síť prochází odzadu, z čehož pochází název Backpropagation. Vhodnou eliminací společných podvýrazů z výsledné derivované funkce je dosaženo velmi rychlého výpočtu dílčích hodnot gradientu pro všechny váhy. Představení této metody mělo velký dopad na navazující historii (viz podkapitola 2.3). [11], [43]

5.2.1.3. Přetrénování a podtrénování

Nalezení globálního minima chybové funkce pro trénovací množinu je složitý optimalizační problém. Při nevhodné volbě parametrů může dojít k tzv. efektu přetrénování (overfitting) sítě. Síť sice efektivně minimalizuje chybovou funkci přes trénovací množinu, avšak na úkor obecnosti nalezené transformace pro vzory vyskytující se mimo trénovací množinu. Síť se příliš

adaptuje na nepodstatné detaily v trénovací množině a není schopna zobecňovat. Přeučení se dá omezit přidáním náhodného šumu do trénovací sady, snahou o minimalizaci velikosti vah, náhodným vypínáním částí sítě [46] či omezením počtu iterací během trénování. V praktické části je použita metoda přidání náhodného šumu do trénovací sady, kdy je s určitou malou pravděpodobností provedena náhodná akce (viz podkapitola 7.3.3).[47] Opačným problémem je podtrénování (underfitting), kdy je síť příliš jednoduchá na to, aby byla schopna správně aproximovat cílovou funkci. Řešením je zvýšení počtu neuronů ve skrytých vrstvách, zvýšení počtu vrstev, nebo kombinace obojího.

5.2.2. Učení bez učitele

Učení bez učitele (samoorganizace, unsupervised learning) funguje bez podpory z vnějšku. Pro učení jsou k dispozici pouze holá vstupní data, neuronová síť v těchto datech hledá opakující se vzory.

Princip samoorganizace je považován za zvláštnost umělých neuronových sítí a je používán všude tam, kde není známa učící množina. Kritériem při učení je výpočet vzdálenosti mezi aktuálními hodnotami a vzory. Hledají se extrémy, tj. maximální výstupní hodnota nebo minimální vzdálenost vzoru od obrazu.

Tento princip učení využívají například asociativní mapy a Kohenovy samoorganizující se topologické mapy. [27], [39] V architektuře robota při učení bez dohledu neexistuje žádná zpětná vazba. Robot tak musí umět objevit vzory a kategorie v datech sám. [40]

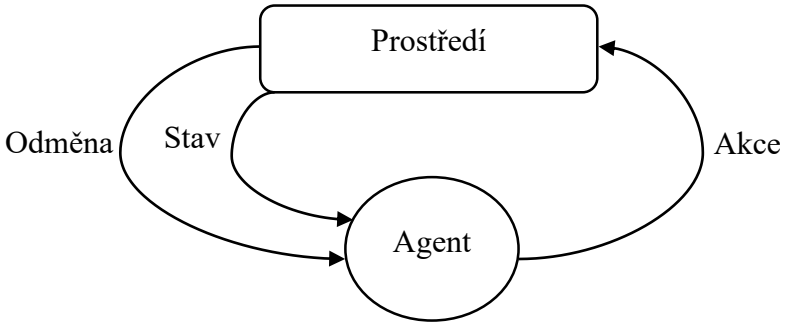
5.2.3. Hluboké učení posilováním (Deep Reinforcement Learning)

Hluboké posilovací učení (Deep Reinforcement Learning) je metoda založená na principu odměn a zpětné vazby. Ve chvíli, kdy se ovládaný systém přibližuje do správného stavu, dochází k odezvě v podobě zvýšené odměny, v opačném případě k jejímu snížení. Tato informace je použita jako chybová funkce pro trénování neuronové sítě. [48]

5.2.3.1. Učení posilováním (Reinforcement Learning)

Učení posilováním se nachází na pomezí mezi učením s učitelem a učení bez učitele. Zatímco u učení s učitelem jsou dopředu známé výsledky pro každý tréninkový příklad a učení bez učitele nemá žádnou podporu a pouze hledá souvislosti ve vstupních datech, metoda učení posilováním má k dispozici odměnu, která však může být časově zpožděná. Celý způsob učení je založen na principu odměn za správné chování, cílem je získání co nejvyšší odměny. Agent se musí naučit chovat v prostředí a samostatně určit, které sekvence akcí vedou ke zvyšování odměny, viz Obrázek 6.

Učení posilováním se často využívá při ovládání her pomocí umělé inteligence, např. právě pomocí neuronových sítí. [49]

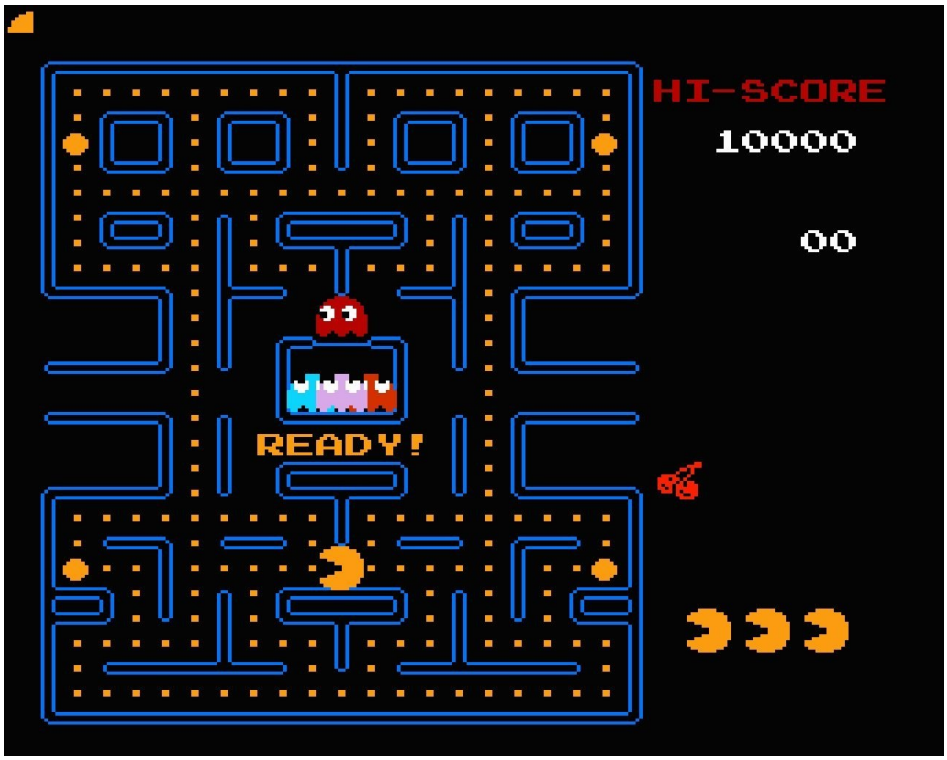


Obrázek 6 - Učení posilováním [48]

Tato metoda však skrývá spoustu úskalí. Agent se ze svých předcházejících akcí musí umět poučit a umět vyhodnotit, která akce (action) v daném stavu (state) přinese největší očekávanou budoucí odměnu (reward). Poté, co přijde na strategii, při které bude pravidelně dosahovat určitého počtu odměn, měl by být schopen prozkoumat své okolí a vyhodnotit, zda může aplikovat výhodnější strategii, aby dosahoval lepších výsledků, neboli umět prozkoumat okolí a těžit z dilema.

Dle tohoto modelu se učí i lidé nebo zvířata, je tedy rovněž inspirovaný neurofyziologií. Učení posilováním v praxi vypadá tak, že si lidé osvojují různé návyky (chování), učí se ze svých chyb a snaží se o lepší výsledky. Příkladem odměn v praxi je chvála od rodičů, lepší plat, postavení apod. Problémy s průzkumem, co je pro koho výhodnější a využívání dilemat v osobním životě, je každodenní lidský proces. To je důvod, proč zkoumat tento problém i pomocí umělé inteligence. V prostředí her lze jednoduše vyzkoušet nové strategie a způsoby

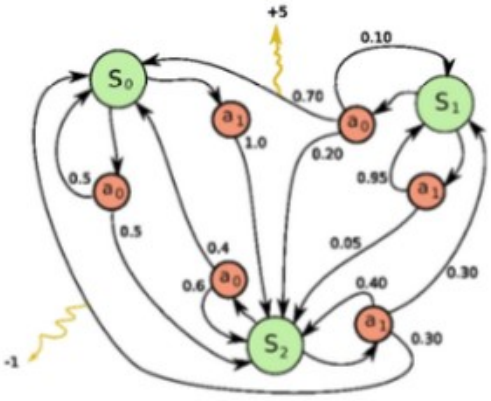
využití neuronových sítí. [48], [50] Přístupy využívající hluboké neuronové sítě byly zkoumány na hrách Atari, konkrétně na 2600 hrách od Arcade Learning [51] (viz Obrázek 7) byla použita stejná architektura neuronových sítí a stejný algoritmus učení. Při souboji s člověkem překonala umělá inteligence člověka odborníka na třech z šesti zkoumaných her. [49]



Obrázek 7 - Učení posilováním na hrách Atari od Arcade Learning [52]

5.2.3.2. Markovovův rozhodovací proces

Učení posilováním lze formulovat pomocí Markovova rozhodovacího procesu. [48] Markovův rozhodovací proces je definován jako množina stavů ve stochastickém prostředí (např. hry), mezi kterými lze přecházet prováděním akcí (např. pohyb vpravo, vlevo), jež jsou vybírány na základě přiřazené pravděpodobnosti. Za různé stavy prostředí získává agent větší, či menší odměnu. Provedením akce dochází ke změně prostředí, která vede k novému stavu, ve kterém agent může provádět nové akce (viz Obrázek 8). Agent volí budoucí akce dle tzv. politiky, jež se snaží maximalizovat pravděpodobnost výběru akce, která povede k maximální budoucí odměně.



Obrázek 8 - Markovovův rozhodovací proces [48]

Epizoda procesu (např. konkrétní hra) je tvořená konečnou posloupností stavů, akcí a odměn, končící terminálovým stavem s_n (např. „Game Over“ na obrazovce v případě hry). Markovovův proces vychází z předpokladu, že pravděpodobnost dalšího stavu s_{i+1}, závisí jenom na stavu současném s_i a akci a_i. Nikoliv na předchozím stavu nebo akci.

$$\mathbf{s_0, a_0, r_1, s_1, a_1, r_2, s_2, \dots, s_{n-1}, a_{n-1}, r_n, s_n}$$

Při dlouhodobém posilovacím učení není vhodné brát v potaz pouze okamžité odměny, ale i budoucí odměny, protože cesta k lepšímu stavu může vést přes stavy s horší odměnou. Jelikož je prostředí stochastické, není vždy očekávané budoucí odměny možno dosáhnout. Během trénování se proto používá diskontní očekávaná budoucí odměna, která uměle snižuje váhu budoucí odměny na základě náhodnosti prostředí. V rámci dobré strategie by měl agent vždy volit akci, která maximalizuje budoucí odměnu. [48]

5.2.3.3. Q-learning

Q-learning je definovaný funkcí Q(S, A), která představuje maximální budoucí diskontovanou odměnu při provedení akce a ve stavu s, při optimálním pokračování od aktuálního okamžiku.

$$Q\ (s_t, a_t) = \max R_{t+n}$$

Rovnice 3 - Rovnice pro Q-learning

Rovnice vyjadřuje nejlepší možný dosažitelný výsledek na konci epizody (hry) o n stavech po provedení akce a_t ve stavu s_t. Q-funkce určuje kvalitu každé akce v daném stavu, výběrem akce s nejvyšší Q-hodnotou v každém stavu je dosaženo optimální politiky (strategie).

$$\pi\,(s) = \arg\max_a Q\,(s, a)$$

Rovnice 4 - Pravidlo pro výběr nejlepší akce v každém stavu

Kde π je politika, neboli pravidlo, jak zvolit akci v každém stavu.

Budoucí odměnu lze vypočítat součtem odměn za všechny následující stavy, tento součet však není v praxi použitelný, neboť vyžaduje znalost budoucího vývoje prostředí. Budoucí (diskontovanou) očekávanou hodnotu však lze vyjádřit rekurzivním zápisem, kdy nejvyšší očekávaná budoucí odměna v aktuálním stavu je rovna součtu okamžité odměny v následujícím stavu a budoucí očekávané odměny v následujícím stavu. Tento rekurzivní zápis se nazývá Bellmanova rovnice:

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

Rovnice 5 - Bellmanova rovnice

kde r je okamžitá odměna pro stav $Q(s', a')$, je diskontní konstanta (menší, než 1 a větší, než 0 – určuje váhu očekávaného Q, např.: 0,5; 0,7) a $max_{a'} Q(s', a')$ je nejvyšší očekávaná budoucí odměna z možných akcí v novém stavu s'.

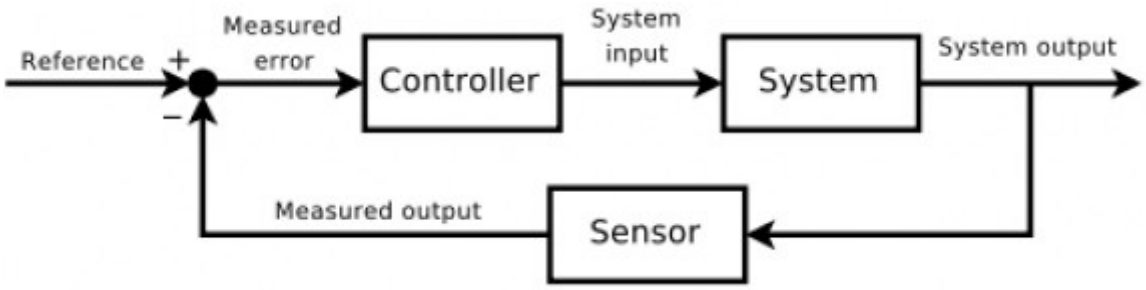
Maximální budoucí odměna za aktuální stav a akci je kombinací okamžité odměny a maximální budoucí odměny dosažitelné z následujícího stavu. Nejdůležitější myšlenkou Q-learningu je Bellmanovu rovnici použít jako iterativní předpis a Q-funkci pomocí tohoto předpisu postupně aproximovat. Na počátku učení je Q-funkce inicializována náhodně. Opakovanou aplikací Bellmanovy rovnice konverguje k ideální politice, učení však musí být prováděno dostatečně dlouhou dobu. [48] U hlubokého učení posilováním je funkce Q aproximována pomocí neuronové sítě a pravá strana Bellmanovy rovnice je použita jako očekávaný výstup při jejím učení.

5.2.3.3.1. Postup trénování

Pokud je učení prováděno on-line a neuronová síť je trénována vždy po každém průběhu zpětnovazební smyčky, může dojít ke konvergenci do nevhodného lokálního minima díky korelacím mezi po sobě jdoucími kroky. Je proto vhodnější neuronovou síť trénovat po dávkách. Nejprve je po nějaký čas pouze zaznamenáván stav simulace, provedená akce a aktuální odměna. Po nashromáždění většího množství záznamů je pro každý záznam vypočítána Bellmanova rovnice a sestavena trénovací sada pro neuronovou síť. Tato sada je následně randomizována, kdy je pořadí řádků v sadě náhodně promícháno. Teprve poté je neuronová síť učena. Po dokončení učení se celý proces opakuje, dokud není dosaženo konvergence. Díky randomizaci je odstraněna korelace mezi po sobě jdoucími kroky, což usnadňuje učení neuronové sítě pomocí metody největšího spádu.

6. Teorie řízení (Control Theory)

Teorie řízení se zařazuje jako interdisciplinární odvětví inženýrství a matematiky a zabývá se chováním dynamických systémů se vstupy a modifikování jejich zpětné vazby. Jedná se o koncept zpětnovazební smyčky, která se používá pro regulaci dynamického chování, viz Obrázek 9.



Obrázek 9 - Teorie řízení, koncept zpětnovazební smyčky (zdroj: wikipedia)

Regulace využívá zpětné vazby z prostředí. Od požadované hodnoty se odečítá senzorem na výstupu změřená regulovaná veličina a celkovým výsledkem je chybový signál. Ten je použit jako zpětná vazba a zpracován regulátorem tak, aby mohl být interpretován jako vstup do regulovaného systému. Regulátor je navržen tak, že monitoruje výkon a porovnává jej s výstupem (output).

V praktické části práce je představeno ovládání jednoduchého robotického zařízení, jehož cílem je maximalizace odměny v daném simulovaném prostředí.

Cílem teorie řízení je ovládání systému (zařízení) tak, aby dosahoval stanovených cílů. Důležitým aspektem je výsledná stabilita systému – zdali výstup konverguje k referenční hodnotě, osciluje kolem ní, či diverguje.

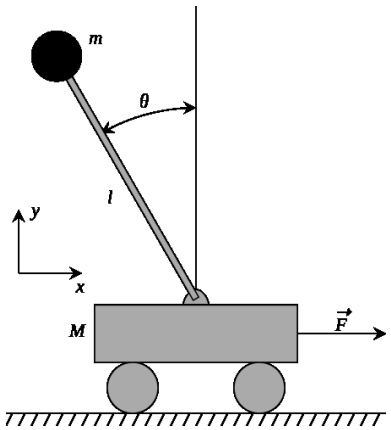
Používá se ve schematickém stylu blokového schéma. Přenosová funkce (funkce systému) matematicky vyjadřuje vztah mezi vstupem a výstupem na základě diferenciálních rovnic, které popisují systém.

Teorii řízení lze aplikovat na všechny systémy, které je možné ovládat a pozorovat pomocí zpětnovazební smyčky, kromě robotiky například na elektronické systémy či biologické a sociální systémy. [53], [54]

6.1. Jednoduché kyvadlo (Single Pendulum)

Klasické kyvadlo je těleso zavěšené na čepu tak, že se může kývat nebo točit kolem celé pevné vodorovné osy. Je-li kyvadlo vychýleno do strany z klidové rovnovážné polohy, vratná síla způsobená gravitací vždy zrychlí kyvadlo zpět k rovnovážné poloze. Díky působení vratné síly bude kyvadlo oscilovat kolem rovnovážné polohy a střídavě měnit potenciální energii kyvadla na kinetickou a naopak. [55]

Podtypem kyvadla je obrácené kyvadlo (Inverted pendulum), které má své těžiště nad jeho vztyčným bodem. Tento typ kyvadla je použit v praktické části, viz kapitola 7. Zatímco klasické kyvadlo má stabilní polohu visí-li dolů, obrácené kyvadlo má stabilní polohu ve vzpřímené poloze neboli labilní rovnovážné poloze (viz Obrázek 10). Rovnovážná poloha je stav, kdy se těžiště kyvadla nachází přesně nad kloubem otáčení a směr tíhové síly se shoduje se směrem vektoru vedoucího od konce kyvadla do kloubu. V případě vychýlení kyvadla z této polohy vytváří tíhová síla v kloubu točivý moment, což vede k oscilaci, která je při ovládání robotických zařízení často nežádoucí. Problém vyvážení převráceného kyvadla je základní úlohou v dynamice a teorii řízení, viz kapitola 6. [56] Pro zjednodušení byl v této práci zanedbán rozdíl mezi tíhovou a gravitační silou a tření v kloubu kyvadla. Je však kladen důraz na to, aby navržená řešení byla schopna tyto rozdíly postihnout, a to s minimálními nároky na změnu.



Obrázek 10 - Obrácené kyvadlo (zdroj: wikipedia)

Výpočet točivého momentu generovaného gravitační silou

$$\tau = m * g * l * \sin\varphi$$

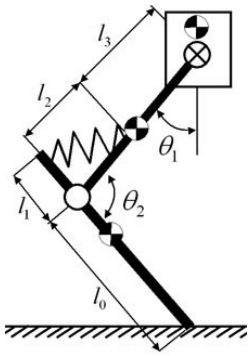
Rovnice 6 - Rovnice pro výpočet točivého momentu

kde m je hmotnost kyvadla,
φ je úhel, o který je kyvadlo vychýleno z rovnovážné polohy
g je gravitační konstanta (g = 9,81) a
l je délka ramene kyvadla

Tento vzorec lze aproximovat pomocí neuronových sítí, které lze následně využít pro ovládání kyvadla, viz kapitola 7. Pokud bude neuronová síť schopna aproximovat rovnici točivého momentu s dostatečnou přesností, může výsledný neuronový regulátor působit silou v opačném směru, a neutralizovat tak efekt gravitační síly[57].

6.2. Dvojité kyvadlo (Double Pendulum)

Kyvadla lze jednoduše spojovat do složitějších zařízení. Nejjednodušším rozšířením je dvojité kyvadlo, což je kyvadlo s dalším kyvadlem připojeným k jeho konci. Dvojité kyvadlo vykazuje dynamické chování a silnou citlivost na počáteční podmínky[58] Existuje několik variant dvojitého kyvadla. Napojená kyvadla na sebe mohou mít stejnou nebo odlišnou délku, či hmotnost. [59] Dvojité kyvadlo je základem pro složitější zařízení, jako jsou například končetiny humanoidního robota. [60]



Obrázek 11 - Dvojité kyvadlo jako základ pro končetiny humanoidního robota [56]

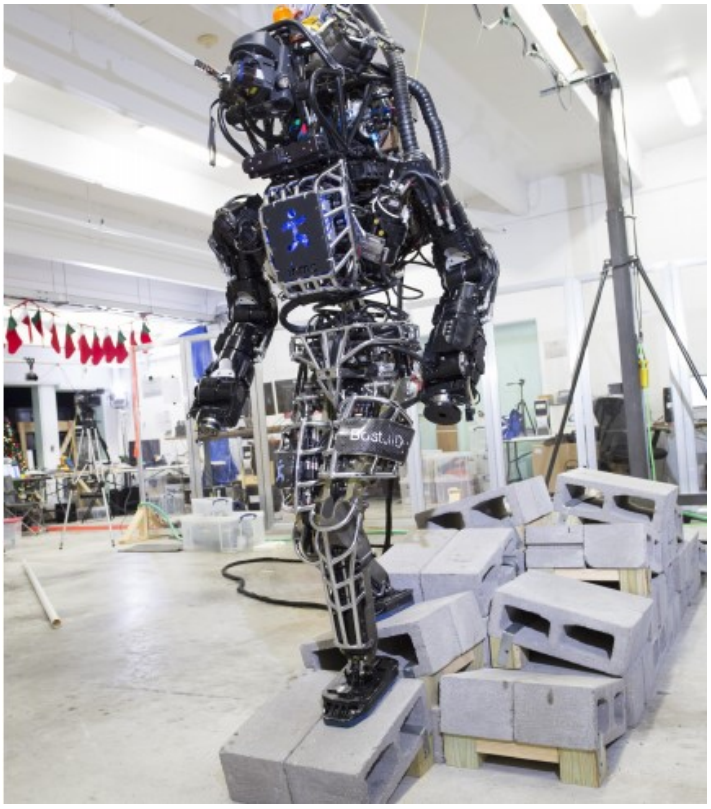
6.3. Jednoduchý chodící robot (Legged Locomotion)

Pohyb pomocí nohou, neboli Legged Locomotion, je jedním z nejdůležitějších problémů teorie řízení, patří i mezi nejsložitější v humanoidní robotice. Významnou roli zde hraje především stabilizace, rovnováha a chůze po dvou končetinách. Při aplikování tohoto způsobu na složitější úkoly, než je chůze, např. chození po schodech nebo posuvném nerovném terénu, je koordinace velkého počtu kloubů velkým problémem. [61]

Způsob dosažení plynulé chůze humanoidního robota by měl splňovat několik podmínek. Mezi ty základní patří: rychlost chůze by měla být konstantní, při dotyku končetiny se zemí by měl být pohyb tak rychlý, jak je jen možné, avšak končetina se musí dotýkat země alespoň polovinu cyklu v rámci mechanismu dvou končetin. Dále se uvažuje výška a délka kroku, také by měl mechanismus končetin umožňovat chůzi dopředu i dozadu. [62]

6.4. Humanoidní robot Atlas

Jednou z nejpokročilejších aplikací teorie řízení je v současnosti robot Atlas, vyrobený firmou Boston Dynamics, s ovládacím systémem od institutu IHMC. Vhodnou aplikací metod teorie řízení je Atlas schopen autonomního pohybu na dvou nohách [60], [63], automatického udržování rovnováhy a plánování komplexních pohybů robotických končetin v reálném prostředí. V publikacích [60], [63] vedených Jerry Prattem, který řídí výzkumnou skupinu mechanických a počítačových inženýrů v robotické laboratoři v IHMC, je představen adaptivní virtuální model řízení kráčejícího robota. Pro kontrolu je použita zpětnovazebná smyčka zahrnující všechny klouby robota s velmi nízkou dobou odezvy v řádu desítek mikrosekund [64]. Díky tomu je Atlas schopen stabilní chůze i po nerovném terénu.



Obrázek 12 - Robot Atlas kráčí po nerovném terénu [64]

6.4.1. IHMC

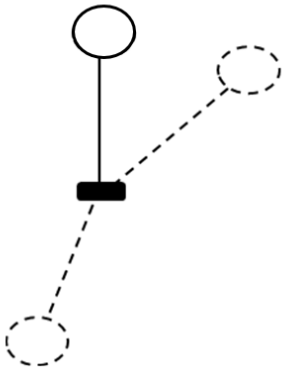
IHMC (Institute for Human and Machine Cognition) byla založena v roce 1990 na University of West Florida dvěma vědeckými pracovníky, Kenneth M. Ford a Alberto J. Caňas. Nyní se jedná o americkou neziskovou organizaci se sídlem ve státě Florida a městech Pensacola a Ocala. Institut spolupracuje s několika americkými univerzitami a dalšími vědeckými institucemi. Po dobu trvání navázala organizace IHMC spolupráci s významnými partnery jako je DARPA, NASA, IBM, Microsoft, Boeing a i s americkou armádou, námořnictvem a letectvem.

Výzkumníci a vědci z IHMC jsou průkopníky v řadě oblastí na využití a rozšíření lidských schopností. Tým obsahuje cca 200 členů a zahrnuje mechanické inženýry, počítačové odborníky, neurology a mnoho dalších profesionálů z různých oborů. Do aktuálního zaměření IHMC patří převážně umělá inteligence, modelování znalostí a jejich sdílení, analýza dat, systém pro tvorbu kognitivních map, odborné studie, zpracování velkých objemů dat, vývoj humanoidních robotů, robotických exoskeletů, rozpoznávání mluvené řeči a mnoho dalších podobných oblastí.[65]

7. Aplikace neuronových sítí pro ovládání robotického zařízení

Cílem této práce je naučit neuronovou síť ovládat jednoduché robotické zařízení. Základem pro ovládání bylo zvoleno jednoduché kyvadlo jako nejjednodušší robotické zařízení. Princip kyvadla je jednoduchý, ale kombinací jednoduchých kyvadel lze skládat složitější zařízení, jako jsou robotické končetiny. Kyvadlo lze jednoduše spojovat ve dvojité nebo trojité kyvadlo. V kompletním mechanismu robota se vyskytují především dvojitá kyvadla v podobě končetin, například nohou a rukou. Dvojité kyvadlo je složeno ze dvou jednoduchých kyvadel. Cílem je proto navrhnout dostatečně obecný postup trénování jednoduchého kyvadla, který umožní snadné budoucí rozšíření na složitější robotická zařízení, jako je dvojité kyvadlo či robotické končetiny (dvě dvojitá kyvadla jsou základ pro chůzi robota). [60]

Cílem trénování neuronové sítě byl stanoven úkol zastavení jednoduchého kyvadla v labilní rovnovážné poloze. Rovnovážná poloha je definována jako dosažení nulového úhlu kyvadla a nulové úhlové rychlosti.



Obrázek 13 - Kyvadlo v rovnovážné labilní poloze

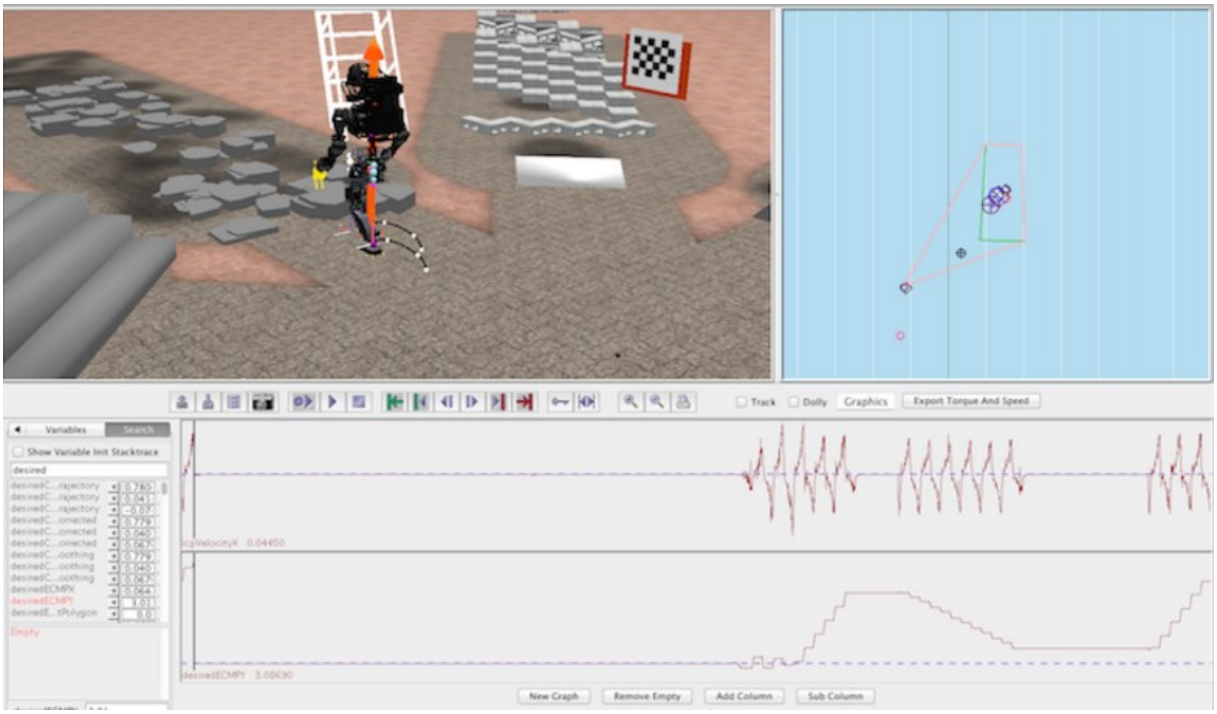
Kód pro již hotové kyvadlo je převzatý od vědeckého centra IHMC (Institute for Human and Machine Cognition. Institut IHMC umožnil v rámci této práce přístup do rozsáhlých projektů DARPA Robotics Challenge [66]. Základní kód pro simulaci kyvadla je jedním z ukázkových projektů v SCS [67] (Simulation Construction Set), avšak v podobě dvojitého kyvadla (Double Pendulum). V práci jsou využity základy kódu IHMC, kód je zjednodušen pro jednoduché kyvadlo (Single Pendulum) a upraven tak, aby jej mohla ovládat neuronová síť.

Existuje několik možností jak neuronovou síť naučit výše uvedenému chování. V rámci této práce jsou použity dva způsoby. První spočívá v natrénování neuronové sítě pomocí rovnice

pro točivý moment, kdy se neuronová síť učí aproximovat již existující funkční regulátor pro jednoduché kyvadlo. Druhý způsob spočívá ve využití metody hlubokého posilovacího učení neuronových sítí. Oba přístupy jsou následně porovnány podle složitosti modelování, problematičnosti trénování sítě a možností rozšíření na složitější robotická zařízení.

7.1. Simulační nástroj (Simulation Construction Set – SCS)

SCS je simulační multiplatformní nástroj s analýzou prostředí vyvinutý v Javě pro vytváření simulací mechanických zařízení, konkrétně pro simulace chůze robotických zařízení a analýzu algoritmů řízení humanoidních robotů. Simulační nástroj začal být vyvíjen roku 1998 a nyní dochází k jeho inovaci. Od loňského roku je současná (původní) verze SCS k dispozici jako open source. [68]



Obrázek 14 - Ukázka simulačního nástroje SCS [68]

Díky SCS může uživatel-programátor využít existující fyzikální engine pro simulaci robotů a celou simulaci vizualizovat v reálném čase pomocí 3D náhledu robota a jeho prostředí. SCS umožňuje nahrát a přetočit uživatelovy simulace, usnadňuje tvorbu 3D grafiky robota s mapováním textur a ovládacích prvků kamery, poskytuje možnost nahrání a uložení dat, úpravu parametrů při běhu simulace, možnost rozšíření simulace pomocí dalších rozhraní (API), v neposlední řadě umí vytvářet výstupní JPG obrázky a krátká videa. [69]

7.2. Naučení neuronové sítě pomocí existujícího regulátoru

První zkoumanou metodou je naučení neuronové sítě ovládat kyvadlo napodobením existujícího regulátoru. Díky ní je ověřeno, zda je neuronová síť vůbec schopná naučit se ovládat jednoduché kyvadlo. Existující regulátor pro jednoduché kyvadlo vychází z rovnice pro točivý moment (viz kapitola 6.2). V rámci simulace bylo použito kyvadlo o hmotnosti 1 kg a délce 0,5 m, gravitační konstanta byla stanovena na 9,81.

Pro definici a trénování neuronových sítí je použit Java Neural Network Framework Neuroph (dále jen Neuroph). [70] Neuroph je psaný v jazyce Java a umožňuje učení neuronových sítí a jako jeden z mála software zaměřených na tuto oblast, dokáže konkrétní neuronovou síť i graficky znázornit. Grafické prostředí Neurophu je založené na platformě NetBeans IDE [71], je proto v ovládání a používání velmi podobný tomuto prostředí.

7.2.1. Návrh neuronové sítě

Volba parametrů sítě (počet vrstev, množství neuronů, aktivační a chybová funkce, rychlost trénování) představuje důležitý krok návrhu řešení. Příliš jednoduchá síť nemusí být schopna cílovou funkci aproximovat s dostatečnou přesností (podtrénování), příliš složitá síť může naopak aproximovat i šum v trénovací sadě, díky čemuž není schopna generalizace na příklady mimo trénovací sadu (přetrénování, viz podkapitola 5.2.1.3). Cílem neuronové sítě je aproximovat rovnici točivého momentu, síť bude mít pouze jeden vstupní a jeden výstupní neuron.

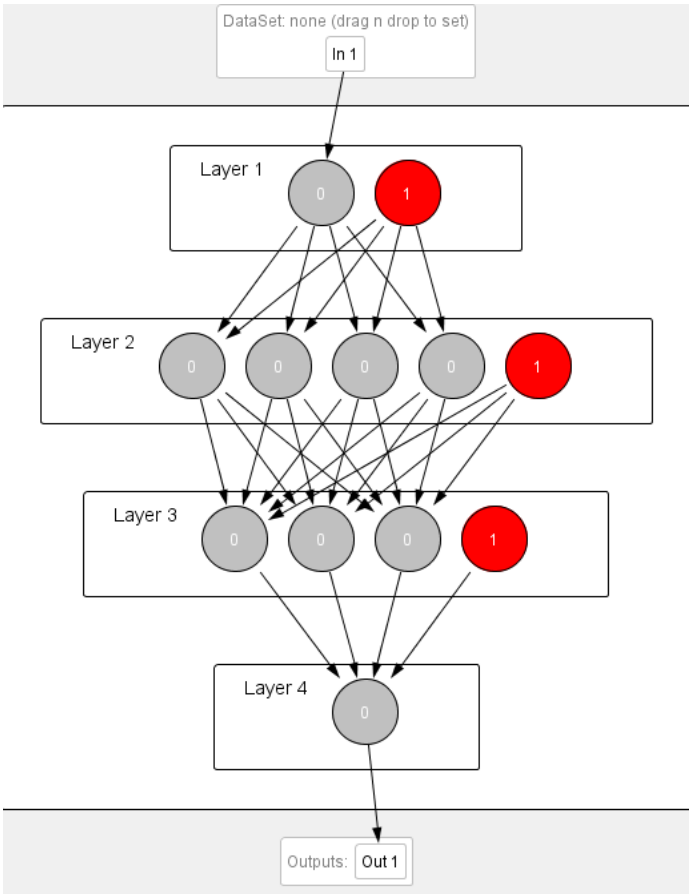
Před vytvořením neuronové sítě v Neurophu je zapotřebí stanovit, kolik má mít vstupů a výstupů. Pro aproximaci rovnice točivého momentu postačí jeden vstup a jeden výstup, jelikož jako vstup bude pouze úhel, ve kterém se kyvadlo nachází

Kyvadlo se umí točit kolem celé své osy, vstupem tedy bude plný úhel, 360 °. Stupně se budou navyšovat po jednotkách, čímž je vytvořeno 360 příkladů. Výstupem neuronové sítě bude výsledek rovnice úhlového zrychlení. Na části datové sady se neuronová síť natrénuje a na menší části otestuje. V Neurophu byla vytvořena neuronová síť s následujícími parametry:

Vstupy:	1
Vnitřní vrstvy:	3, 4
Výstupy:	1
Aktivační funkce	Sigmoid

Tabulka 1 - Parametry pro Neuronovou síť s jedním vstupem, jedním výstupem

Díky Neurophu a jeho grafickému znázornění síť vypadá velmi přehledně.



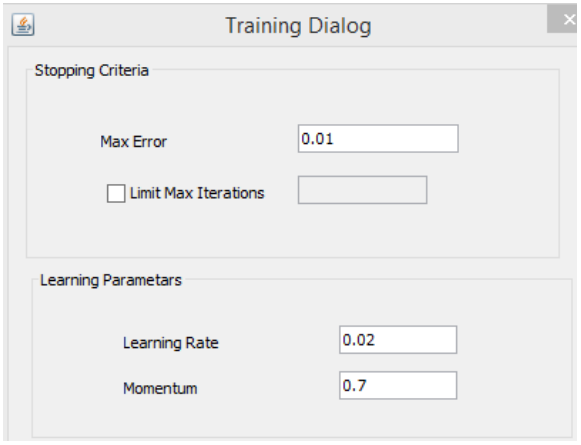
Obrázek 15 - Neuronová síť s jedním vstupem a jedním výstupem

V případě aproximace rovnice pro točivý moment je nutné připravit 2 datové sady. Jedna sada je použita pro trénování neuronové sítě, druhá pro testování. V každé sadě budou vstupem úhly a výstupem výsledek rovnice pro točivý moment. Na počátku byly použity datové sady bez normalizace, aby byla stanovena základní hranice pro následné přístupy učení.

	A	B
1	1	8.25483
2	2	8.920208
3	3	1.384387
4	4	-7.42423
5	5	-9.40705
6	6	-2.74107
7	7	6.445039
8	8	9.705604
9	9	4.042882
10	10	-5.33685

Obrázek 16 – Ukázka nenormalizované datové sady

Než se začne neuronová síť trénovat, je nutné zvolit maximální chybovost (Max Error) a učící frekvenci (Learning Rate). Hodnota Max Error udává velikost chyby, při jejímž dosažení bude trénování ukončeno. Neuronová síť hodnoty však nemusí vůbec nikdy dosáhnout, pokud je špatně navržena či dostává nevhodná data. Postup trénování lze ověřit na grafu a hodnotě celkové chyby sítě (Network Error), která by správně při úspěšném učení měla klesat a postupně se přibližovat hodnotě Max Error. Pokud se neuronová síť učí správně, chybovost klesá až k hodnotě Max Error. Pokud graf například zůstává ve vertikální poloze a hodnoty na Network Error se nijak výrazně nemění, je potřeba neuronovou síť přeučit nebo upravit její architekturu. Hodnota Learning Rate určuje frekvenci učení. Pokud je frekvence příliš velká, neuronová síť se nemusí dokázat nic naučit, pokud však příliš malá, může se naopak přeučit, proto je tuto hodnotu nutné určit experimentálně.

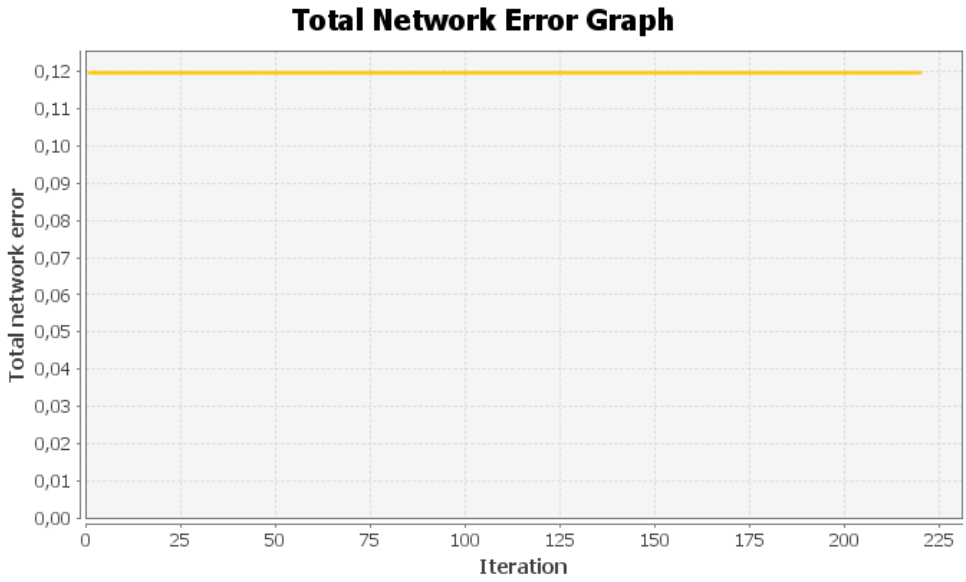


Obrázek 17 - Způsob trénování neuronové sítě

Neuronovou síť je někdy potřeba nechat trénovat několik hodin, složitější i několik dní. Podle grafu a hodnot chyby lze pozorovat, zda se neuronová síť učí správně, nebo se neučí vůbec.

V takovém případě je nutné neuronovou síť přeučit, změnit část, či celou její strukturu, nebo upravit normalizaci datové sady.

Neuroph během učení neuronové sítě vykresluje graf chyby, ze kterého je možné určit, zda se neuronová síť učí či nikoliv (viz Obrázek 18). Pokud hodnota chyby zůstává na stejné hodnotě, či dokonce roste, není potřeba neuronovou síť nechávat déle učit, jelikož by se s danými parametry cílovou funkci nenaučila aproximovat.



Obrázek 18 - Graf nevhodného učení neuronové sítě

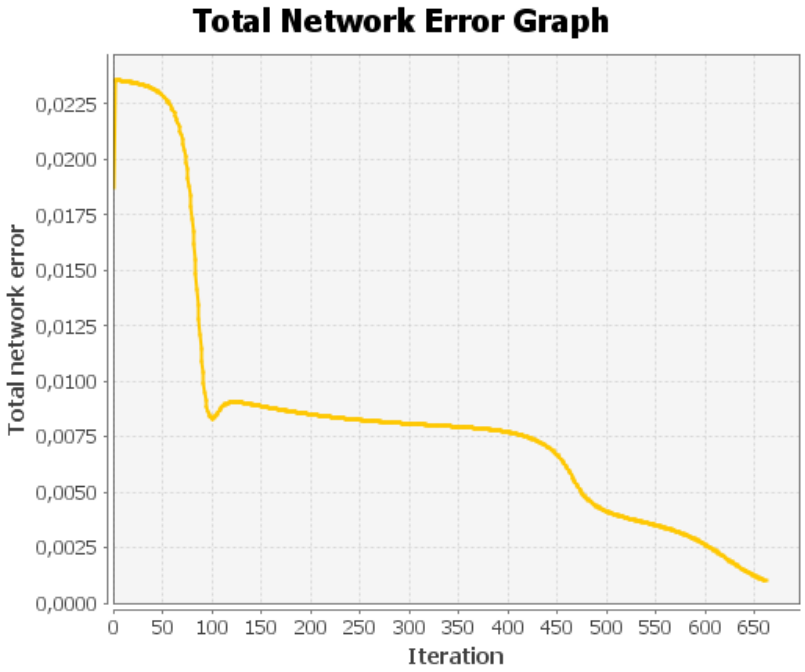
7.2.2. Aproximace funkce sinus

Neuronová síť z předchozí kapitoly se nedokáže natrénovat především proto, že dostává na vstupu nevhodná data z datové sady. Ty jsou nevyhovující, protože se jedná o příliš vysoké hodnoty a trénování pomocí gradientu diverguje. Důležitým krokem je proto normalizace dat. Hodnoty na původním vstupu se vydělí 360, čímž vznikne rozsah vstupů od 0 do 1.

Aby se ověřilo, zda je normalizace efektivní a neuronová síť se dokáže učit, nechá se nejprve neuronová síť učit na nejsložitější části vzorce točivého momentu, a to sinusu vstupu. Architektura neuronové sítě zůstane stejná.

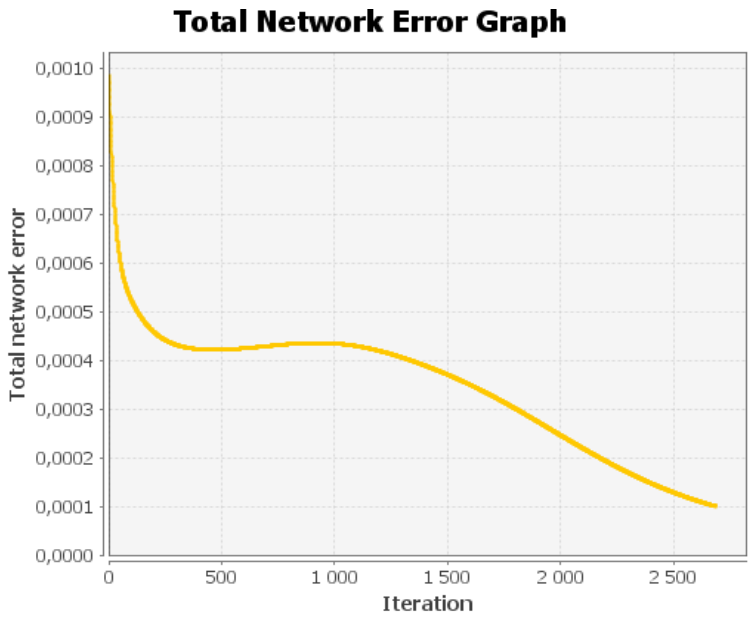
Neuronová síť bude pracovat s novou datovou sadou, ve které budou normalizované úhly jako vstupy a výpočet sinusu úhlů jako výstup. Pořadí řádků trénovací sady je vhodné randomizovat,

aby během trénování pomocí stochastické metody největšího spádu (stochastic gradient descent) nedocházelo k zatížení směru gradientu jedním směrem, a tím konvergenci do špatného lokálního minima. Na
Obrázek 19 a Obrázek 20 je zřetelný efekt promíchání datové sady, kdy graf celkové chyby má výrazně hladší průběh, je-li datová sada promíchána.



Obrázek 19 – Graf učení neuronové sítě na normalizované, nepromíchané datové sadě

V druhém případě pro lepší porovnání bude neuronové síti upravena celá datová sada. Ta se provádí pro vstup i výstup zvlášť. Nejprve je nutné najít ze souboru vstupů minimální a maximální hodnoty pro každou složku vstupního vektoru. Nejprve je od jednotlivých vstupů odečteno minimum a výsledek je vydělen rozdílem maximální a minimální hodnoty, aby byla výsledná hodnota v intervalu od 0 do 1. Výsledkem je normalizovaná sada vstupů. Stejný postup je aplikován i na data výstupu, čímž vznikne normalizovaná sada výstupů. Na konci výpočtu je vhodné data promíchat, aby nebyla pravidelně za sebou, a až poté data set rozdělit sadu na trénovací část a testovací část. Tím je zajištěno, že rozdělení vzorků v trénovací a testovací sadě bude podobné.

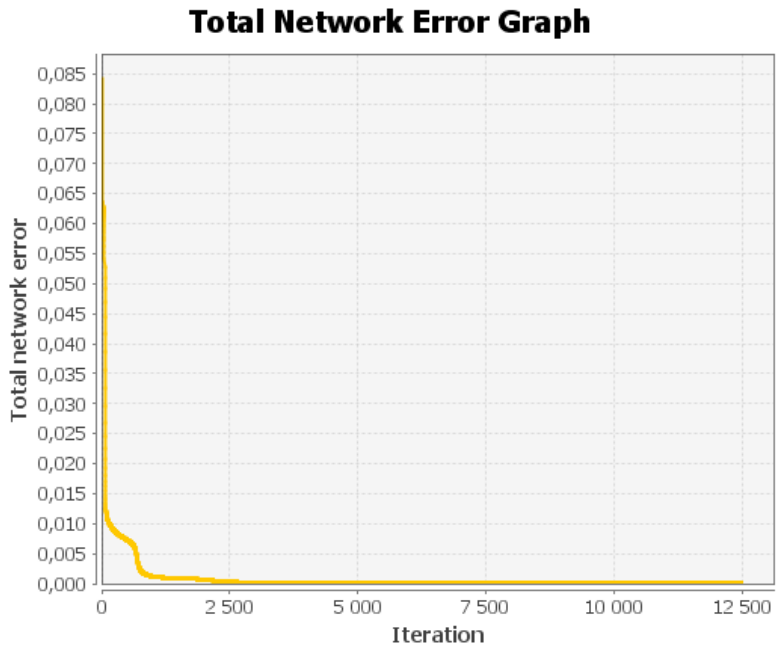


Obrázek 20 - Neuronová síť s normalizovanou a promíchanou datovou sadou

Z obou grafů lze usoudit, že neuronová síť se dokázala naučit výpočty funkce sinus a dosáhla podmínek pro ukončení učícího procesu. Lze porovnat první i druhou možnost, které si jsou velmi podobné, ale už od pohledu je druhý graf učení možnost plynulejší, což značí lépe vyvážený směr gradientu v důsledku promíchání. Proto je potřeba data set upravovat ručně a dávat neuronové síti jen normalizovaná a promíchaná data. Z obou grafů vyplývá, že se neuronová síť zvládla úspěšně naučit počítat sinus a proto by jí nemělo dělat problém naučit se při stejných podmínkách celý vzorec.

7.2.3. Aproximace celého regulátoru

V tuto chvíli by měla být neuronová síť dle předpokladů schopná naučit se celý vzorec úhlového zrychlení. Pro datovou sadu se vstupy osvědčily normalizované a promíchané, jen je potřeba dopočítat výstupy pomocí rovnice točivého momentu. Ty se počítají s původními (neznormalizovanými a nepromíchanými) vstupy (radiány) a až výsledky se normalizují a promíchávají společně. Na konci je opět vhodné data set rozdělit na trénovací a testovací. Na prvním se neuronová síť trénuje, na druhém je testována, zda se naučila správně. Pro správnou funkčnost je během normalizace datové sady uložit i normalizační konstanty (minimum a maximum), aby bylo možné podle nich během používání sítě normalizovat nové vstupy a zpětně de-normalizovat výstup sítě.



Obrázek 21 - Neuronová síť – finální graf

Graf neuronové sítě je výsledkem testovací sady. Z grafu je vidět, že je neuronová síť vhodně naučená a bylo by jí možné zahrnout do kódu kyvadla, které by měla sama ovládat.

7.2.4. Závěr pro neuronovou síť natrénovanou podle existujícího regulátoru

Neuronová síť byla úspěšně natrénována na rovnici točivého momentu pro kyvadlo. Díky tomu je možné pomocí této sítě zastavit pohyb kyvadla v libovolné pozici, včetně cílové labilní rovnovážné polohy.

Tato metoda naučení neuronové sítě ovládat kyvadlo pomocí rovnice slouží především k ověření, že je neuronová síť schopna se naučit kyvadlo s dostatečnou přesností ovládat. Není však příliš výhodná pro rozšiřování na složitější systémy, například pro dvojité kyvadlo nebo pro celého humanoidního robota. Nevýhodou tohoto přístupu je, že je navržen pouze pro jedno konkrétní zařízení. Při použití na složitější zařízení je nutné znovu odvodit rovnice regulátoru, které bude síť aproximovat. Není tedy příliš vhodné trénovat neuronovou síť, která aproximuje existující rovnice, v takovém případě je vhodnější použít definované rovnice přímo.

7.3. Naučení neuronové sítě pomocí hlubokého posilovacího učení

Hluboké učení posilováním (Deep Reinforcement Learning) je metoda založená na učení neuronových sítí na základě odměn za provedené akce. Učení posilováním je založeno na principu zpětné vazby, v případě přibližování správnému stavu dochází k odezvě v podobě zvýšení odměny, na základě této odměny je možné neuronovou síť natrénovat. Více informací je uvedeno v teoretické části, viz kapitola 5.2.3.

Metoda hlubokého učení posilováním je oproti učení kyvadla podle existujícího regulátoru vhodnější z hlediska možných rozšíření na složitější robotické systémy, protože závisí pouze na funkci odměny a nevyžaduje existující regulátor.

Pro experimenty bylo použito kyvadlo se stejnými parametry jako v případě učení podle existujícího regulátoru, tedy kyvadlo o hmotnosti 1 kg a délce 0,5m.

Při použití této metody je nutné kód regulátoru v SCS doplnit o řízení kyvadla pomocí neuronové sítě. Před úpravou kódu je nutné vytvořit vhodnou neuronovou síť, která se stejně jako v předcházejícím přístupu načte do kódu pro ovládání kyvadla, aby kyvadlo bylo ovládáno neuronovou sítí.

Neuronová síť pro použití hlubokého posilovacího učení je rozsáhlejší, má více neuronů, protože jejím výstupem jsou odměny pro jednotlivé diskrétní akce, takže výsledná funkce je složitější na aproximaci. Je označená podle názvu metody v anglickém znění ReinforcementNetwork.

7.3.1. Návrh neuronové sítě pro učení posilováním

Záměrem je, aby neuronová síť zastavila kyvadlo v nulové úhlové rychlosti a nulovém úhlu, v takovém stavu získá největší odměnu. Vstupem do neuronové sítě je stav kyvadla, který v tomto případě představuje úhel a úhlovou rychlost kyvadla. Neuronová síť tedy potřebuje dva vstupní neurony.

Architektura skryté vrstvy byla volena empiricky a to ve vrstvách 7, 7, 6. Později byla rozšířena na 20, 20, 10 pro možnost lepší aproximace cílové funkce, která může být velice složitá.

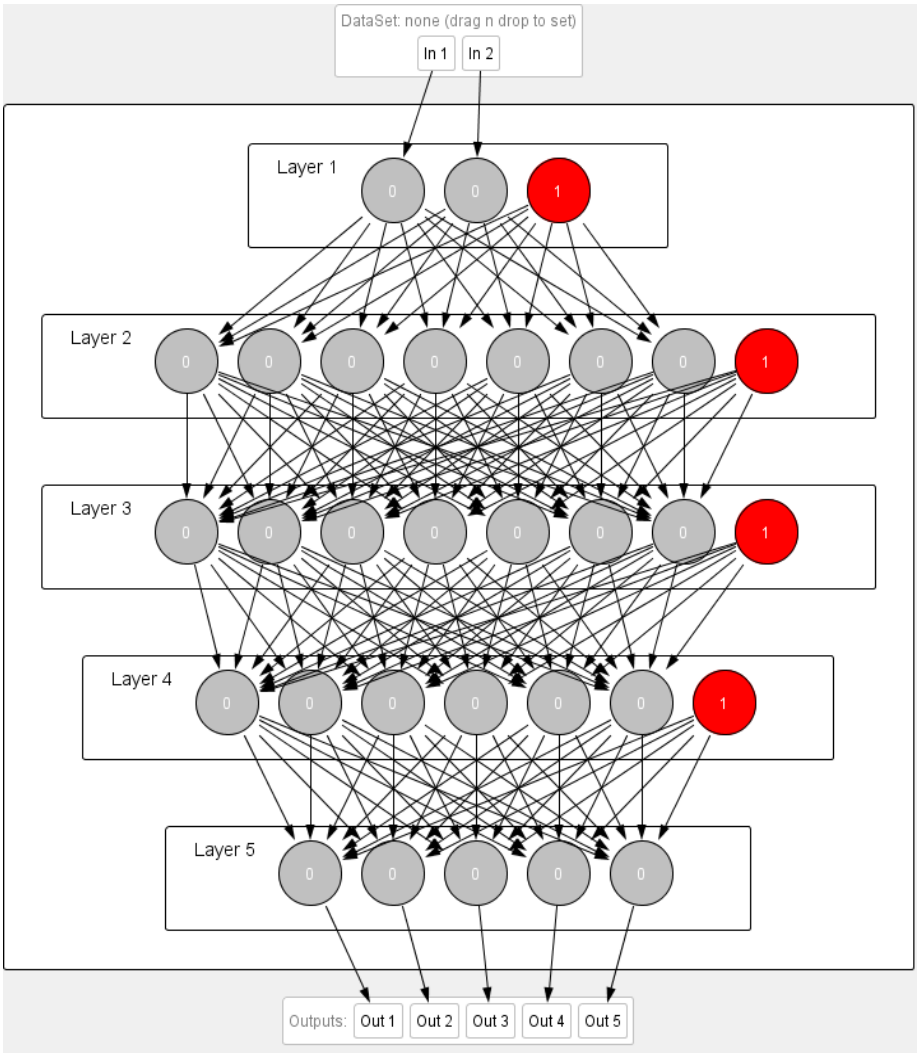
Všechny neurony v celé architektuře používají aktivační funkci sigmoid. Skryté vrstvy by mohly mít jiné aktivační funkce (např. tanh), avšak pro konzistentnost celé architektury byla zvolena jedna výchozí aktivační funkce sigmoid.

Počet výstupů sítě musí odpovídat počtu akcí, které neuronová síť jako agent může v prostředí provést. Pro vyvážení kyvadla bylo zvoleno 5 akcí, 2 akce pro nastavení kladného točivého momentu (pohyb doleva, málo a více), 2 akce pro nastavení záporného točivého momentu (pohyb doprava, málo a více) a 1 akce pro nulový točivý moment. Každá akce je mapována na určitou hodnotu točivého momentu, který je vždy po 10 kroků simulace aplikován v kloubu kyvadla. Hodnoty točivých momentů byly zvoleny tak, aby během 10 kroků simulace měly dostatečný efekt na stav kyvadla, a docházelo tak k měřitelné změně odměny. Zvolené hodnoty točivých momentů pro jednotlivé akce a parametry neuronové sítě jsou uvedeny v Tabulka 2.

Vstupy:	2
Vnitřní vrstvy:	7, 7, 6
Výstupy:	5
Akce (točivý moment)	10, 5, 0, -5, -10

Tabulka 2 - Parametry pro Reinforcement Network

Grafické zobrazení neuronové sítě v Neurophu:



Obrázek 22 - Reinforcement Network

7.3.2. Výpočet odměny

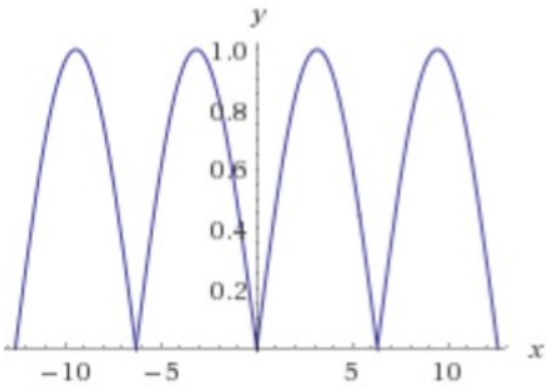
Odměna je největší v případě, kdy se kyvadlo dostane do cílového stavu. V jiných stavech musí být odměna navržena tak, aby při přibližování stavu kyvadla k cílovému stavu rostla. Na základě toho se neuronová síť učí správným akcím.

Cílový stav nastane v případě, kdy jsou úhel i rychlost rovny nule. Pro výpočet odměny na základě stavu je potřeba výchozí stav načtený ze simulace vhodně transformovat. Protože se kyvadlo může před dosažením rovnovážné polohy několikrát protočit, je úhel transformován funkcí sinus, jelikož je periodická a pro nulový úhel nabývá hodnoty 0. Na výsledný sinus je ještě aplikována absolutní hodnota, výsledný rozsah je tak od 0 do 1. Tento rozsah je vhodný pro neuronovou síť využívající sigmoid neurony. Ještě před aplikací funkce sinus je úhel

vydělen 2, aby výsledná funkce nabývala nulové hodnoty pouze v periodách 360° (nikoliv 180°), tedy pouze tehdy, když se kyvadlo nachází v labilní rovnovážné poloze. Celková rovnice pro transformaci úhlu je následující:

$$y = \left|\sin(\frac{x}{2})\right|$$

Rovnice 7 - Rovnice pro transformaci úhlu



Obrázek 22 - Graf ilustrující výpočet úhlu

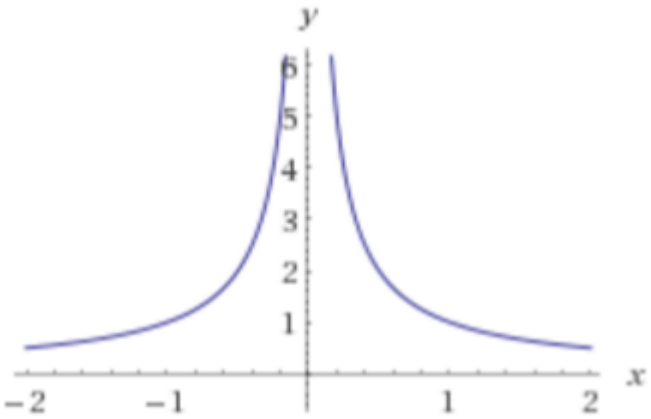
Úhel je normalizován podle výše uvedené rovnice. Úhlová rychlost je normalizována podle maximální rychlosti, které kyvadlo může dosáhnout, když na něj nepůsobí žádná jiná síla. V případě kyvadla zvoleného pro účely této práce se jedná o hodnotu 2π radiánů za sekundu. Tyto normalizované hodnoty jsou rovněž použity jako vstupy do neuronové sítě, aby výpočet odměny a neuronová síť vycházely ze stejných vstupních hodnot.

Po transformaci je možné hodnoty úhlu a rychlosti spojit do jednoho vektoru a vypočítat jeho vzdálenost od ideálního stavu pomocí euklidovské vzdálenosti.

Jak již bylo zmíněno, odměna by měla být tím větší, čím více dokáže neuronová síť dostat kyvadlo do svislé polohy. Ideální je, aby kyvadlo dosahovalo nulové rychlosti a nulového úhlu. V případě, že se budou hodnoty kyvadla přibližovat k nule, měla by neuronová síť dostávat větší odměnu. Jedna z funkcí, která tuto vlastnost splňuje je převrácená hodnota vzdálenosti od ideálního stavu. X zde představuje vzdálenost aktuálního stavu od ideálního, tedy od stavu 0, 0:

$$y = \frac{1}{|x|}$$

Rovnice 8 - Průběžná rovnice znázorňující výpočet odměny



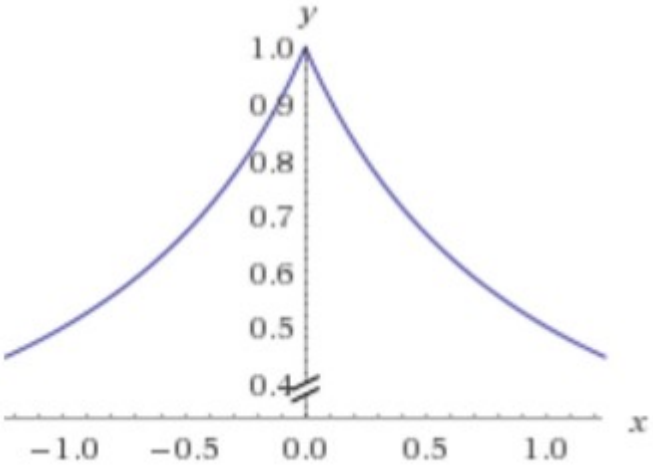
Obrázek 23 - Průběžný graf ilustrující výpočet odměny

V případě neznaménkové vzdálenosti stačí brát v potaz pouze pravou stranu grafu v oblasti kladných hodnot. Graf ilustruje to, že ve chvíli, kdy se parametry kyvadla budou blížit k nule, bude neuronová síť odměňována nejvíce.

Nevýhodou výše uvedené funkce je shora neomezený obor hodnot, směrem k 0 se její hodnoty blíží k nekonečnu, což vede k numerické nestabilitě. Pro trénování neuronové sítě je vhodné výstup omezit na interval od 0 do 1. K tomuto účelu je možné funkci posunout o jednu jednotku směrem vlevo:

$$y = \frac{1}{|x| + 1}$$

Rovnice 9 - Finální rovnice znázorňující výpočet odměny



Obrázek 24 - Finální graf ilustrující výpočet odměny

Tato funkce dokáže pro každý stav vypočítat odměnu. Je potřeba, aby se neuronová síť naučila maximalizovat odměnu.

7.3.3. Trénování pomocí Bellmanovy rovnice

Bellmanova rovnice byla použita pro trénování sítě dle výzkumu hlubokého posilovacího učení. [50] Bellmanova rovnice funguje na základě odměn a zpětného propagování odměn tak, aby byla pro každou kombinaci stavu a akce aproximována očekávaná budoucí odměna. Více informací o Bellmanově rovnici je uvedeno v teoretické části (5.2.3.3 Q-learning).

$$Qs, a) = r + ymax_{a'}Q(s', a')$$

Rovnice 10 - Bellmanova rovnice

Ve vzorci se vyskytuje konstanta y, která umožňuje diskontovat vliv budoucí očekávané odměny ve stavu s'. Její hodnota byla experimentálně testována v intervalu zvolena 0,3–0,9, její hodnota však neměla na učení výrazný vliv. Neuronová síť je trénována vždy na záznamu 1000 kroků simulace, které jsou promíchány kvůli lepší konvergenci učícího algoritmu. Aby se předešlo přetrénování, kdy se neuronová síť naučí provádět pouze jednu akci, jsou v průběhu trénování s pravděpodobností 0,15 provedeny náhodné akce.

Trénovací sada pro neuronovou síť je sestavena ze zaznamenaných stavů, které jsou transformovány pomocí předpisu popsaného v předchozí podkapitole. Očekávané výstupy pro

akci jsou vypočítány pomocí Bellmanovy rovnice, pokud byla akce během simulace provedena a je pro ni známa okamžitá odměna r. Pokud akce nebyla v daném kroku provedena, je její očekávaný výstup nastaven na výstup sítě aplikované na daný vstupní stav (nedojde tedy ke změně výstupu). Pro každý záznam se tak ve vektoru očekávaných hodnot změní pouze jedna hodnota oproti čistému výstupu sítě.

Bellmanova rovnice by měla pro každý stavu a akci konvergovat k očekávané odměně. Správně naučená neuronová síť by měla znát pro každý stav akci s nejlepší budoucí odměnou.

Tento způsob se v práci příliš neosvědčil. Neuronová síť se trénovala velmi dlouho, ale nikdy se nedotrénovala do zdárného konce a ve většině případů se zasekávala v lokálním minimu. Proto Bellmanova rovnice nakonec nemohla být v této práci použita.

Ve článku [50] byla neuronová síť pomocí Bellmanovy rovnice trénována na rozsáhlejších datech (miliony bitmapových obrázků jako stavy) po výrazně delší dobu (na grafických kartách přibližně týden), což může vysvětlovat, proč se výsledky nepodařilo v menším měřítku zopakovat.

7.3.4. Alternativní výpočet odměny a možnost trénování

Alternativní výpočet, který je v práci využit, je rovněž založen na myšlence Bellmanovy rovnice. Neuronová síť se také učí předpovídat výši odměny na konci simulace.

Oproti původní Bellmanově rovnici si neuronová síť si zaznamená všechny kroky jednoho cyklu simulace (např. 1000 kroků), a poté hledá pro každý zaznamenaný stav nejbližší lokální maximální odměnu. Pro každý stav je tak přímo nalezena nejbližší maximální odměna a není odhadována iterativně, jako v případě Bellmanovy rovnice. Stejně jako u Bellmanovy rovnice je možné nalezenou maximální hodnotu diskontovat na základě vzdálenosti aktuálního stavu od nalezeného maxima. Díky přímému výpočtu maximální odměny ze záznamu dochází k výrazně rychlejší konvergenci sítě, kdy je síť schopna kyvadlo přiblížit ideálnímu stavu již po 5 cyklech.

7.3.5. Trénování neuronové sítě

Než se neuronová síť spustí a začne trénovat, je zapotřebí nastavit parametry učení. Parametry použité pro učení sítě v této práci jsou:

Český název	Anglické znění	Nastavená hodnota
Rychlost učení	Learning rate	0,1
Počet iterací	Max iteration	500
Maximální chybovost	Max error	$1E^{-6}$

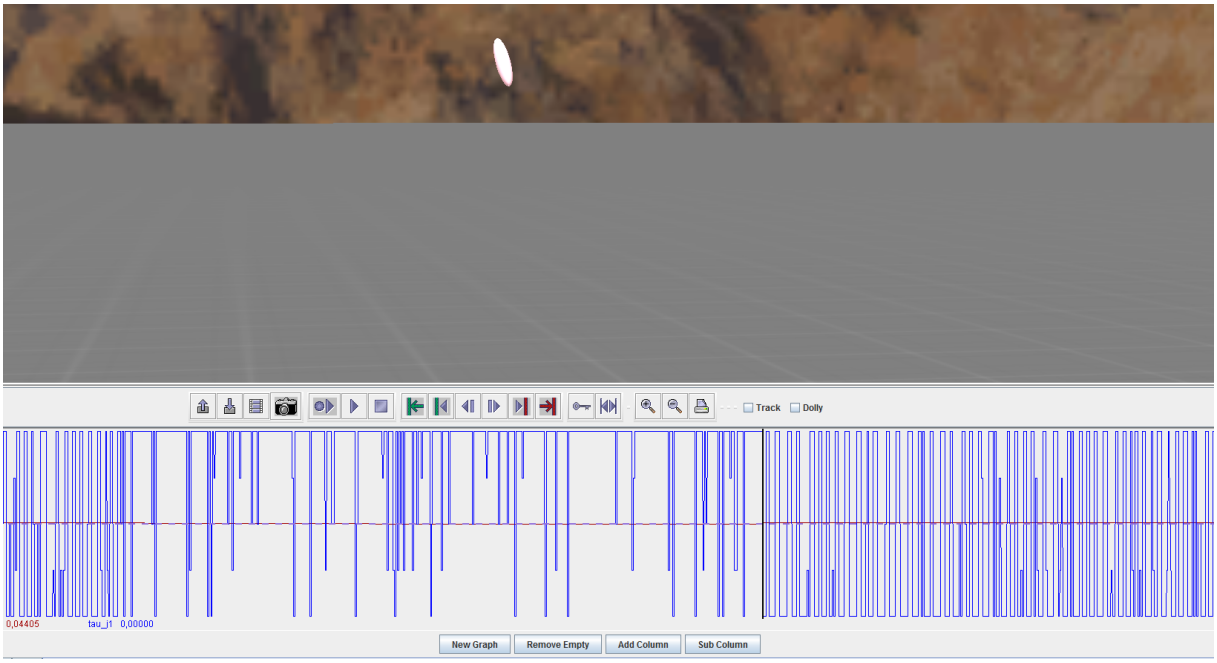
Tabulka 3 - Parametry učení Neuronové sítě

Případně je možné nastavit ještě parametr momentum (momentum backpropagation), který průměruje poslední chody a vytvoří tak plynulejší funkci učení. V případě trénování sítě pro tuto práci nebyl tento parametr nastaven,

7.3.6. Zhodnocení Reinforcement Network

Neuronová síť se dokáže trénovat. Na vstupu má úhly znormalizované pomocí periodické funkce založené na sinu a rychlost normalizovanou podle maximální rychlosti, které kyvadlo může dosáhnout, pokud na něj nepůsobí jiná síla. Neuronové síti je předkládán záznam akcí a jejich odměn v budoucnu zjištěných během simulace. Pomocí maximálních odměn se neuronová síť dokáže naučit chování, které se velmi přibližuje požadovanému.

Pomocí této metody se dokáže neuronová síť naučit kyvadlo velmi těsně přiblížit k labilní rovnovážné poloze. V procesu učení neuronová síť provádí zcela neočekávané chování, na začátku učení např. velmi rychle roztáčí kyvadlo, po několika cyklech a zlepšujících se odměnách začne kyvadlo přibližovat do vhodné polohy. Výsledný graf učení na Obrázek 25 znázorňuje sofistikovanost neuronové sítě a schopnost naučit se i velmi složitou funkci.



Obrázek 25 - Graf akcí prováděných učící se neuronovou sítí v SCS

Pomocí alternativního výpočtu budoucí odměny se neuronová síť dokáže natrénovat velmi rychle. Po zhruba 5 cyklech je neuronová síť naučená tak, že dokáže kyvadlo téměř zastavit v labilní rovnovážné poloze. V porovnání výpočtu odměny s pomocí Bellmanovy rovnice, kdy se neuronová síť učila několik hodin a nebyla schopná se správně natrénovat.

7.3.7. Problémy při použití Reinforcement Network

Během výzkumu bylo nutné čelit mnoha problémům, díky kterým se neuronová síť nedokázala správně učit. Bylo nutné vymyslet jiný způsob namísto Bellmanovy rovnice, pomocí níž se neuronová síť vůbec nebyla schopná natrénovat. Také bylo nutné veškeré hodnoty správně normalizovat a randomizovat. S příliš velkými hodnotami se neuronová síť neumí nic naučit.

Výsledná síť je schopna kyvadlo velmi těsně přiblížit rovnovážnému stavu, není však schopna jej v tomto stavu udržet. Důvodem mohou být zvolené diskrétní akce, které jsou příliš silné na jemné vyvážení poblíž cílového stavu. Přidáním více akcí však dochází k pomalejšímu učení a kyvadlo ani tak není schopné rovnovážné polohy dosáhnout.
Možným řešením by mohlo být použití spojitého učení posilováním, které umožňuje spojitý výstup akce. Jeho implementace je však výrazně složitější, jelikož vyžaduje trénování dvou propojených sítí, a přesahuje rámec této práce.

7.3.8. Možnost dalšího směřování výzkumu

Možností pokračování v tomto výzkumu se nabízí několik. Systém lze rozšířit na složitější zařízení, například dvě spojená dvojitá kyvadla, která by mohla neuronová síť naučit chodit.

Ve výzkumu lze pokračovat i změnou učení neuronové sítě. Nyní se neuronová síť učí na základě diskrétních hodnot, vybírá z pěti nejlepších odměn nejvhodnější akci. Sofistikovanější způsob učení neuronové sítě by byl založen na spojitém učení. Neuronová síť by měla pouze 1 výstup, který by udával, jakou silou má neuronová síť působit na kyvadlo. Nevybírala by z možných akcí, ale vybrala by přímo nejvhodnější hodnotu točivého momentu.

Na takový způsob by bylo nutné použít dvě neuronové sítě. Jedna by uměla vypočítat hodnotu akce. Druhá neuronová síť by sloužila jako kritik, jenž odhaduje, zda daná změna hodnoty akce v daném stavu povede ke zvýšení či snížení odměny. [50]

8. Závěr

V práci byly nejprve představeny neuronové sítě a princip jejich fungování, popsány možnosti jejich využití pro umělou inteligenci a pro ovládání robotických zařízení. Teoretická část práce detailně popisuje a navzájem propojuje dvě témata umělé inteligence, která jsou zásadní pro analýzu problematiky a následnou implementaci – neuronové sítě a robotická zařízení.

V praktické části práce jsou navrženy dva způsoby učení neuronové sítě, které vedou k požadovanému chování. Prvním způsobem se neuronová síť naučila ovládat inverzní kyvadlo a následně jej zastavit v libovolné poloze na základě aproximace rovnice točivého momentu vyvolaného gravitační silou působící na kyvadlo. Druhý způsob zkoumal možnost naučení neuronové sítě požadovanému chování na základě moderní metody hlubokého učení posilováním (Deep Q-learning). Vytvořená neuronová síť, která už v současné chvíli dokáže ovládat jednoduché robotické zařízení, demonstruje široké možnosti uplatnění tohoto přístupu. Trénování a testování neuronových sítí bylo prováděno v simulačním nástroji IHMC Simulation Construction Set.

Výsledky trénování dokazují, že ovládání robotického zařízení pomocí neuronových sítí je možné, a jedná se tedy o perspektivní směr dalšího výzkumu. Ačkoliv v případě učení posilováním nebyla neuronová síť schopna kyvadlo v cílové labilní rovnovážné poloze zastavit, velmi se tomuto stavu přiblížila. Celkové chování této sítě rovněž napovídá, že se naučila inteligentní strategii pro ovládání kyvadla. Další výzkum bude směřovat ke spojitému učení posilováním, které je vhodnější pro plynulé ovládání robotických zařízení, a k rozšíření na složitější robotická zařízení ve spolupráci s institutem IHMC.

Seznam obrázků

Seznam rovnic

Seznam tabulek

Seznam zkratek

Java Neural Network Framework Neuroph ..... Neuroph
Institute for Human and Machine Cognition..... IHMC
Simulation Constraction Set..... SCS
umělých neuronových sítí ..... UNS
Neuronové sítě..... NS

Seznam literatury

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *ArXiv150201852 Cs*, Feb. 2015.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[3] "Home | DRC Finals." [Online]. Available: http://www.theroboticschallenge.org/. [Accessed: 29-Apr-2016].

[4] Ivan Havel, *Robotika: úvod do teorie kognitivních robotů]*. Praha: STNL, 2013.

[5] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[6] N. Wiener, *Cybernetics Or Control and Communication in the Animal and the Machine*. MIT Press, 1961.

[7] J. Kapoun, "Norbert Wiener: otec kybernetiky," 27-Nov-2004. [Online]. Available: http://businessworld.cz/veda-a-historie/norbert-wiener-otec-kybernetiky-3947. [Accessed: 24-Jan-2016].

[8] John Van Neumann, *The general and logical theory of automata*. L.A. jeffress editor, 1951.

[9] John von Neumann, *Probabilistic logisc and the synthesis of reliable organisms from unreliable components*. 1956.

[10] Jan Kapoun, "Scienceworld | Nový druh vědy si dobře rozumí s byznysem." [Online]. Available: http://www.scienceworld.cz/neziva-priroda/novy-druh-vedy-si-dobre-rozumi-s-byznysem-1715/?switch_theme=mobile. [Accessed: 10-Apr-2016].

[11] J. Šíma and R. Neruda, *Teoretické otázky neuronových sítí*. Matfyzpress, 1996.

[12] M. Novák, *Umělé neuronové sítě: teorie a aplikace*. V Praze: C.H. Beck, 1998.

[14] Malone Bob, "George Devol: A Life Devoted to Invention, and Robots," 26-Sep-2011. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/industrial-robots/george-devol-a-life-devoted-to-invention-and-robots. [Accessed: 10-Apr-2016].

[15] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[16] N. Yadav, A. Yadav, and M. Kumar, *An Introduction to Neural Network Methods for Differential Equations*. Dordrecht: Springer Netherlands, 2015.

[17] F. Rosenblatt, *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Spartan Books, 1962.

[18] B. Widrow, "Generalization and information storage in networks of adaline `neurons'," in *Collection*, 1962, pp. 96–104.

[19] R. S. Forsyth, "'The strange story of the Perceptron,'" *Artif. Intell. Rev.*, vol. 4, no. 2, pp. 147–155, Jun. 1990.

[20] M. L. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Mit Press, 1972.

[21] SRI International, "Shakey the Robot | SRI International." [Online]. Available: https://www.sri.com/work/timeline-innovation/timeline.php?timeline=computing-digital#!&innovation=shakey-the-robot. [Accessed: 19-Apr-2016].

[22] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[23] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 81, no. 10, pp. 3088–3092, May 1984.

[24] Sen Song, Kenneth D. Miller, and L. F. Abbott, "Competitive Hebbian Learning Through Spike-Timing-Dependent Synaptic Plasticity," in *Departments of Physiology and Otolaryngology*, San Francisco, 2000.

[25] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," vol. 1986, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, pp. 673–695.

[27] Jana Tučková, *Úvod do teorie a aplikací umělých neuronových sítí*, 1. ed. Praha: ČVUT, 2003.

[28] "Mars Pathfinder / Sojourner Rover." [Online]. Available: http://www.jpl.nasa.gov/missions/mars-pathfinder-sojourner-rover/. [Accessed: 17-Feb-2016].

[29] David Dubov, "NASA Names First Rover to Explore the Surface of Mars," *News from Sorouner*, 08-Jul-1997. .

[30] "Robot Asimo se dá ovládat pouhou myšlenkou," *Novinky.cz*. [Online]. Available: http://www.novinky.cz/internet-a-pc/165277-robot-asimo-se-da-ovladat-pouhou-myslenkou.html. [Accessed: 17-Feb-2016].

[31] Jon Excell, Jason Ford, and Stuart Nathan, "Two legs good," *The Engineer*, 22-Nov-2000. .

[32] "How ASIMO Works," *HowStuffWorks*, 11-Apr-2007. [Online]. Available: http://science.howstuffworks.com/asimo.htm. [Accessed: 17-Feb-2016].

[33] Richard Trenholm, "Honda's Asimo robot shows off new moves," *CNET*. [Online]. Available: http://www.cnet.com/news/hondas-asimo-robot-shows-off-new-moves/. [Accessed: 17-Feb-2016].

[34] Satoshi SHIGEMI, Koji KAWABE, and Takahiro NAKAMURA, "Development of New ASIMO – Realization of Autonomous Machine."

[35] "Robonaut." [Online]. Available: http://robonaut.jsc.nasa.gov/default.asp. [Accessed: 17-Feb-2016].

[36] T. Macháček, "Biomach, výpisky z biologie," *Nervová soustava*, 2005. [Online]. Available: http://www.biomach.cz/. [Accessed: 19-Apr-2016].

[37] SINĚLNIKOV, R. D. a kolektiv, *Atlas anatomie člověka*. Státní zdravotnické nakladatelství, 1965.

[39] Miroslav Šnorek and Marcel Jiřina, *Neuronové sítě a neuropočítače*, vol. 1996. Praha: ČVUT.

[40] "Umělé neuronové sítě pro učení robotů | VŠE." [Online]. Available: https://www.vse.cz/vskp/13834_umele_neuronove_site_pro_uceni_robotu. [Accessed: 27-Mar-2016].

[41] Stanley C. Ahalt, Ashok K. Krishnamurthy, Prakoon Chen, and Douglas E. Melton, "Competitive learning algorithms for vector quantization," in *Department of Electrical Engineering*, Ohio, 1990.

[42] Hiroyuki Miyamoto, Tohru Setoyama, and Ryoji Suzuki, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," Japan, 1988.

[43] M. A. Nielsen, "Neural Networks and Deep Learning," 2015.

[44] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *, International Joint Conference on Neural Networks, 1989. IJCNN*, 1989, vol. 1989, pp. 593–605 vol.1.

[45] Léon Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *19th International Conference on Computational StatisticsParis France*, 2010, pp. 177–186.

[46] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of Neural Networks using DropConnect," presented at the Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1058–1066.

[47] Věra Kůrková, "Kolmogorov's theorem and multilayer neural networks," presented at the Institute of Computer Science, 1992.

[48] T. Matiisen, "Guest Post (Part I): Demystifying Deep Reinforcement Learning," *Nervana*, 21-Dec-2015. [Online]. Available: http://www.nervanasys.com/demystifying-deep-reinforcement-learning/. [Accessed: 17-Apr-2016].

[49] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *ArXiv13125602 Cs*, Dec. 2013.

[50] Jonathan J. Hunt, Timothy P. Lillicrap, Alexander Pritzel, Nicolas Heess, T. Erez, Yuval Tassa, D. Silver, and Daan Wierstra, "CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING," in *Published as a conference paper at ICLR*, 2016.

[51] "The Arcade Learning Environment." .

[52] "Pac-Man for Android," *mobile9*. [Online]. Available: http://gallery.mobile9.com/f/2976587/. [Accessed: 29-Apr-2016].

[53] C. Kilian, *Modern Control Technology*, 3 edition. Clifton Park, N.Y: Delmar Cengage Learning, 2005.

[54] R. Antunes and V. Gonzalez, "A Production Model for Construction: A Theoretical Framework," *Buildings*, vol. 5, no. 1, pp. 209–228, Mar. 2015.

[55] Miriam Webster, "Kyvadlo," *Miriam Webster's Collegiate Encyclopedia*. p. 1241, 2000.

[56] Tom Benson, “Rocket Stability,” *NASA Official*, 2015. [Online]. Available: https://spaceflightsystems.grc.nasa.gov/education/rocket/rktstab.html. [Accessed: 27-Apr-2016].

[57] Rich Chi Ooi, Thomas Bräunl, and Jie Pan, “Balancing a Two-Wheeled - Autonomous Robot.” 2003.

[58] R. B. Levien and S. M. Tan, “Double Pendulum: An experiment in chaos,” *Am. J. Phys.*, vol. 1993.

[59] Alex Small, “One signature of chaos in the double pendulum,” in *Department of Physics and Astronomy*, California, 2013.

[60] Jerry Pratt, J. Hrr, C.-M. Chew, H. Herr, and G. Pratt, “Adaptive virtual model control of a bipedal walking robot,” *IEEE Int. Jt. Symp.*, pp. 245 – 251, 1998.

[61] Nerses Ohanyan and Jan Peters, “Computational Learning and Motor Control Lab | Research / Legged Locomotion browse,” 2008. [Online]. Available: http://www-clmc.usc.edu/Research/LeggedLocomotion. [Accessed: 27-Apr-2016].

[62] Amanda Ghassaei, *The Design and Optimization of a Crank-Based Leg Mechanism*, vol. April 20, 2011. .

[63] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, “Virtual Model Control: An Intuitive Approach for Bipedal Locomotion,” *Int. J. Robot. Res.*, vol. 20, no. 2, pp. 129–143, Feb. 2001.

[64] J. Smith, D. Stephen, A. Lesman, and J. Pratt, “Real-Time control of Humanoid Robots using OpenJDK,” presented at the Proceedings of the 12th International Workshop on Java Technologies for Real-time and Embedded Systems, New York, USA, 2014, pp. 29–36.

[65] “IHMC,” *IHMC*. [Online]. Available: http://robots.ihmc.us/. [Accessed: 28-Apr-2016].

[66] DARPA, “Darpa robotic challenge,” *Finals 2015*, 2015. [Online]. Available: http://www.theroboticschallenge.org/. [Accessed: 25-Apr-2016].

[67] IHMC Robotic lab and Robotics Lab, “IHMC Open Source Software Update,” *Robotics Lab*, 20-Nov-2014. [Online]. Available: http://robots.ihmc.us/blog/2014/11/20/z35nu8u31y8gyxvplrgc77mb5uu03n. [Accessed: 25-Apr-2016].

[68] IHMC Robotics Open Source Software Team, “Blog,” *Robotics Lab*, 2016. [Online]. Available: http://robots.ihmc.us/blog/. [Accessed: 27-Apr-2016].

[69] Jerry Pratt, “Documentation of SCS,” *Control Algorithms for Walking and Manipulation*

[70] *Java Neural Network Framework Neuroph*. [online]. Available : http://neuroph.sourceforge.net/

[71] O. C. NetBeans, *NetBeans*. 2016. [online]. Available : https://netbeans.org

[72] František Šolc and Luděk Žalud, *Robotika*. (Vysokoškolské přednášky) 2002.

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Systémové inženýrství a informatika
Forma: Prezenční
Obor/komb.: Informační management (im3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Urbanová Tereza	Pod Strání 24, Broumov - Nové Město	I1301525

TÉMA ČESKY:

Neuronové sítě pro ovládání robotických systémů

TÉMA ANGLICKY:

Neural Networks for Robot Control

VEDOUCÍ PRÁCE:

Ing. Karel Petránek - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

1. Definice problému, průzkum existujících přístupů 2. Zhodnocení přístupů, návrh řešení 3. Testování 4. Zhodnocení výsledků

SEZNAM DOPORUČENÉ LITERATURY:

[1]C. W. Anderson, Learning to control an inverted pendulum using neural networks, IEEE Control Systems Magazine, vol. 9, no. 3, pp. 31-37, Apr. 1989. [2]M. Hild, C. Thiele, and C. Benckendorff, The Distributed Architecture for Large Neural Networks (DISTAL) of the Humanoid Robot MYON., in IJCCI (NCTA), 2011, pp. 260-266. [3]A Brief History of Neural Nets and Deep Learning, Andrey Kurenkov's Web World. [Online]. Available: http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/. [4]K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, Neural networks for control systems - a survey, Automatica, vol. 28, no. 6, pp. 1083-1112, 1992. [5]K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, arXiv:1512.03385 [cs], Dec. 2015. [6]M. A. Nielsen, Neural Networks and Deep Learning, 2015. [Online]. Available: www.neuralnetworksanddeeplearning.com. [7]?Neural Networks for Machine Learning - University of Toronto, Coursera. [Online]. Available: https://www.coursera.org/course/neuralnets. [8]Deep Learning. [Online]. Available: https://www.udacity.com/course/deep-learning--ud730. [9]J. Hrr, J. Pratt, C.-M. Chew, H. Herr, and G. Pratt, Adaptive virtual model control of a bipedal walking robot, in Intelligence and Systems, 1998. Proceedings., IEEE International Joint Symposia on, 1998, pp. 245-251. [10]J. Smith, D. Stephen, A. Lesman, and J. Pratt, Real-Time Control of Humanoid Robots Using OpenJDK, in Proceedings of the 12th International Workshop on Java Technologies for Real-time and Embedded Systems, New York, NY, USA, 2014, pp. 29:29-29:36. [11]J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, Virtual model control: An intuitive approach for bipedal locomotion, The International Journal of Robotics Research, vol. 20, no. 2, pp. 129-143, 2001. [12]V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602, 2013.

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....