

Mendelova univerzita v Brně
Provozně ekonomická fakulta

Řízení autonomního agenta pomocí neuroevoluce

Diplomová práce

Vedoucí práce:
Ing. Jiří Lýsek, Ph.D.

Bc. Martin Hnátek

Brno, 2018

Rád bych zde poděkoval svému vedoucímu Ing. Jiří Lýskovi, Ph.D. za jeho rady a čas, který mi věnoval při řešení této práce.

Čestné prohlášení

Prohlašuji, že jsem práci: **Řízení autonomního agenta pomocí neuroevoluce** vypracoval samostatně a veškeré použité prameny a informace uvádím v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací*.

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 24. listopadu 2018

.....

Abstract

Autonomous agent control using neuroevolution

Abstrakt

Řízení autonomního agenta pomocí neuroevoluce

Tato práce se zabývá trénováním autonomního agenta - auta s pomocí algoritmu neuroevoluce. Toto zahrnuje tvorbu simulačního prostředí pro agenta, vhodným návrhem agenta (senzorů a řízení) a také návrhem fitness funkce

Obsah

1	Úvod a cíl práce	7
1.1	Úvod do problematiky	7
2	Neuronové sítě	8
2.1	Druhy úloh neuronových sítí	8
2.2	Neuron	8
	Aktivační funkce	8
	Lineární funkce	8
	Sigmoid	9
	Tanh	9
	RELU	10
2.3	Genetické algoritmy	10
	Princip	11
	Kódování	11
	Křížení	11
	Mutace	11
	Selekce	12
2.4	Neuroevoluce	12
3	Použité technologie	13
3.1	Docker	13
3.2	Vue	13
3.3	Node.js	13
3.4	Traefik	13
4	Vlastní práce	14
4.1	Simulace	14
	Fitness funkce	14
4.2	Serverová část	14
	Výpočetní cluster	14
	Průběh vyhodnocování	14
4.3	Klientská část	15
4.4	Simulace	15
	Agent	15
4.5	Experimenty	15
5	Závěr a zhodnocení	16
6	Reference	17
	Přílohy	18

A	CD se zdrojovým kódem
---	-----------------------

19

1 Úvod a cíl práce

1.1 Úvod do problematiky

S růstem výpočetního výkonu a rozvojem **gpugpu** (paralelizace výpočtů na grafické kartě) se neuronové sítě ukázaly jako mocný nástroj pro řešení složitých problémů na které standardní metody umělé inteligence nestačily. K dalším možnostem

2 Neuronové sítě

Neuronové sítě jsou model strojového učení, který je volně založený na principu zvířecího mozku. (PATTERSON, Josh. 2017, s. 41)

2.1 Druhy úloh neuronových sítí

Neuronové sítě se používají především pro řešení regresních a klasifikačních problémů.

2.2 Neuron

Neuron je základní jednotka neuronových sítí, která je definovaná jako suma všech jejích vstupů a aplikace aktivační funkce.

$$\sigma(\sum_{i=0}^N \theta \cdot x_i + b)$$

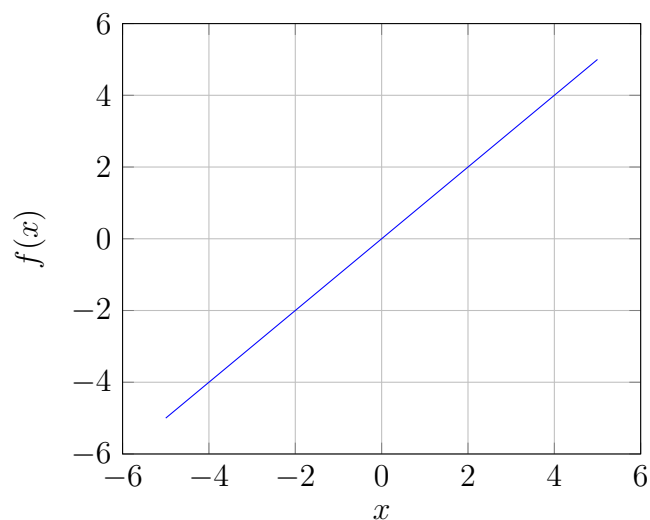
Aktivační funkce

Aktivační funkce se používá pro definování výstupu a zavedení nelinearity. Bez nich by byla neuronová síť schopna aproximovat pouze n-dimenzionální rovinu. (PATTERSON, Josh. 2017, s. 65)

Dalším využitím je omezení výstupních hodnot. Například aktivační funkce sigmoid se s oblibou používá u výstupní vrstvy neuronových sítí určených ke klasifikačním problémům, protože je to relace $\mathbb{R} \rightarrow \{0..1\}$, která se dá jednoduše jako "jistota" neuronu, že se jedná o výstup, který neuron reprezentuje. Podobně se dá uvažovat i o funkcích jako je například softmax a tanh, které také najdou hojně využití u klasifikačních problémů.

Lineární funkce

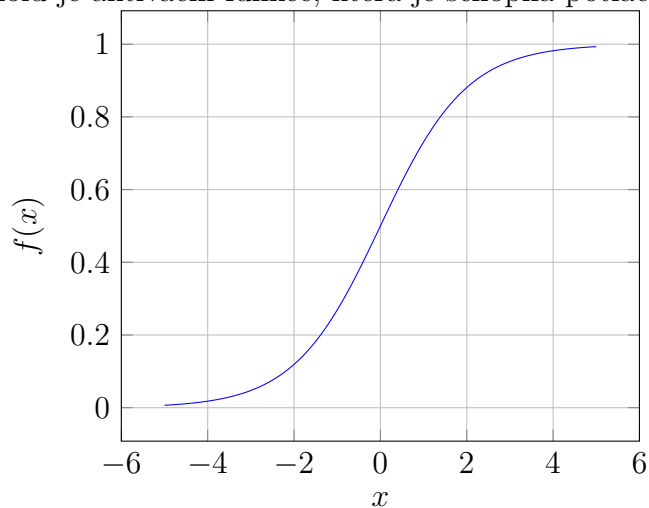
Vrací vstup, tak jak je. Využití najde především u vstupní vrstvy neuronové sítě a u neuronových sítí, které řeší regresní typy úloh.



$$f(x) = x$$

Sigmoid

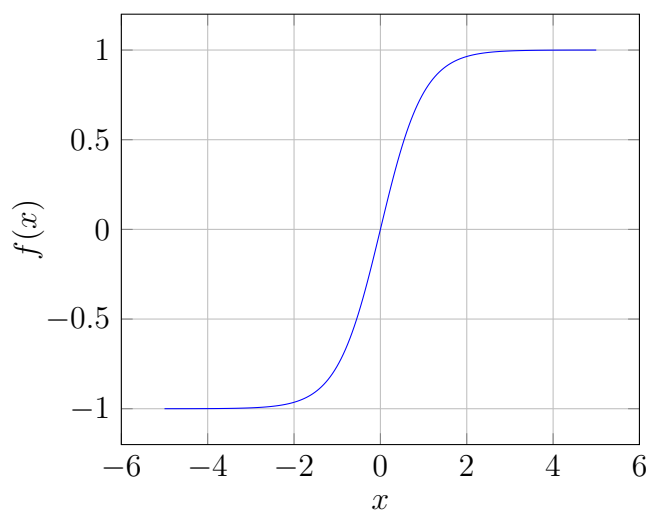
Sigmoid je aktivační funkce, která je schopná potlačit extrémní hodnoty



$$f(x) = \frac{1}{1 + e^{-x}}$$

Tanh

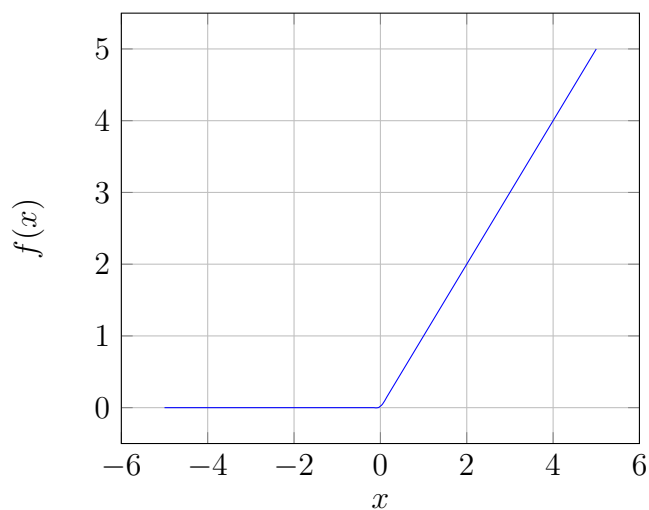
Tanh je funkce obdobná sigmoidu. Hlavní rozdíl mezi ní a sigmoidem je ten, že její obor je v rozmezí -1 a 1 hodí se proto i pro záporná výstupy, které vyžadují záporná čísla. (PATTERSON, Josh. 2017, s. 67)



$$f(x) = \tanh(x)$$

RELU

RELU je aktivační funkce, která je podobná lineární aktivační funkci s tím rozdílem, že pokud vstupní hodnota nepřesáhne určitého prahu výstupem je 0. Její hlavní výhodou je to, že zabraňuje problémům s takzvaným explodujícím gradientem (PATTERSON, Josh. 2017, s. 69)



$$f(x) = \begin{cases} x & x \geq 0, \\ 0 & x < 0, \end{cases}$$

2.3 Genetické algoritmy

Genetické algoritmy slouží k řízenému prohledávání stavového prostoru založené na teorii evoluce.

Princip

Základní myšlenka spočívá ve vygenerování náhodných jedinců (řešení problému) a jejich postupné zlepšování s pomocí operací křížení a mutace. Proces zlepšování genomů probíhá na základě jeho hodnocení (fitness).

Samotný algoritmus pak lze rozdělit na následující kroky (MITCHELL, Melanie., 1996, s. 12)

1. Vygeneruj náhodnou populaci o n chromozomech
2. Pro každý chromozom spočítej jeho fitness
3. Opakuj n krát
 - a) Vyber pár chromozomů na základě jejich ohodnocení (selekce)
 - b) S určitou pravděpodobností p_c rozděl pár chromozomů na náhodném místě a jejich spojení.
 - c) S určitou pravděpodobností mutuj daného jedince
4. Nahraď současnou populaci populací, která vznikla předchozím krokem
5. Jdi na krok 2

Kódování

Způsob zápisu řešení problému. Existuje mnoho různých kódování a každý má své výhody a nevýhody. Většinou se snažíme vybírat kódování, které dobře reprezentuje daný problém. Například při

Zde je seznam několika nejpoužívanějších kódování (HYNEK, Josef., 2008, s. 42-43):

1. Binární - řetězec bitů, který může například reprezentovat jednu nebo více numerických hodnot.
2. Realná čísla - Jedno nebo více reálných čísel
3. Kombinatorické - Může být například seznam čísel označujících

Křížení

Křížení je operátor, který kombinuje dva jedince do jednoho. Jeho implementace je samozřejmě závislá na

Mutace

Mutace náhodně modifikuje chromozom a zavádí tak do populace variaci.

Selekce

Selekce je proces výběru dvou jedinců na něž jsou později aplikovány genetické operátory jako je křížení a mutace.

2.4 Neuroevoluce

3 Použité technologie

3.1 Docker

3.2 Vue

3.3 Node.js

3.4 Traefik

4 Vlastní práce

Vlastní práce se skládá ze dvou částí: klientská část, která slouží k vizualizaci algoritmu a zobrazení výsledků ze serverové části. Serverová část pro maximální zrychlení simulace.

4.1 Simulace

Fitness funkce

Fitness funkce je důležitou součástí simulace, která zásadně ovlivňuje chování výsledných agentů.

4.2 Serverová část

Serverová část vyhodnocuje jednotlivé jedince distribuované s pomocí fronty úkolů. Frontu poskytuje knihovna **bull**, která používá **redis** pro správu údajů o jednotlivých úkolech.

Výpočetní cluster

Ukázalo, že vyhodnocování simulace zabírá neúměrné množství času a to i na nejvýkonnějším dostupném počítači. Například vyhodnocení jedné generace populace o 1024 jedincích zabralo 290 s na nejsilnějším dostupném pc. Z tohoto důvodu bylo rozhodnuto o distribuci výpočetní zátěže mezi více počítačů. Byl vytvořen výpočetní cluster s následující specifikací:

Procesor	RAM	Počet
S5P6818 Octa core	1 GB	2
Broadcom BCM2837B0 quad-core	1 GB	1
Phenom X4 965	8 GB	1
Intel atom		

Pro snadnou distribuci a správu byly všechny počítače zorganizovány do docker swarmu. Docker swarm obsahoval jednoho managera, který zároveň spouštěl klientskou aplikaci a další služby:

1. **Portainer** pro správu clusteru
2. **Arena** pro správu **bull**
3. **redis** používaný knihovnou **bull**

Průběh vyhodnocování

Serverová část pracuje dle diagramu ??, kde je vidět, že klient zadává do fronty úkoly (genom a nastavení simulace) a jednotliví zpracovatelé (počítače v clusteru), kteří si je z ní vyberou, jednotlivé genomy vyhodnotí a hodnotu fitness funkce pošlou zpět

na klienta, který jakmile dostane všechny hodnoty zpět provede genetický algoritmu (mutace, křížení, ...) a novou generaci pošle znovu na vyhodnocení. Tento přístup má několik výhod a to:

1. Robustnost - Pokud jeden nebo více zpracovatelů selže (je například odpojen ze sítě) je možné pokračovat ve vyhodnocování (neúspěšný úkol se automaticky vrátí zpátky do fronty)
2. Dobré rozložení zátěže - Jelikož si zpracovatel vytahuje úkoly z fronty je vždy optimálně zatížen a není třeba řešit rozložení mezi různě výkonnými a zatíženými počítači.
3. Možnost vyhodnocení více úkolů zároveň - Jelikož se vyhodnocují jednotlivé genomy lze spustit i více simulací zároveň bez většího dopadu na výkon výpočtů.

Lze i namítnout, že se zde projevuje určitá rezie při síťové komunikaci se serverem, což může být zdrojem určitého zpomalení. Nicméně se toto zpomalení neprojevilo v průběhu testování clusteru.

4.3 Klientská část

4.4 Simulace

Agent

Definice samotného agenta zásadně ovlivňuje výsledek simulace, protože stanovuje vstupy a výstupy do a z neuronové sítě.

4.5 Experimenty

5 Závěr a zhodnocení

<https://snapshot.raintank.io/dashboard/snapshot/FmoHRo3sfdk8eAQlkRc4rfKK3YF043oE?orgId=2>

6 Reference

- BUDUMA, NIKHIL. *Fundamentals of deep learning: designing next-generation machine intelligence algorithms*. Sebastopol: O'Reilly, 2017. ISBN 978-149-1925-614..
- PATTERSON, JOSH. *Deep learning : a practitioner's approach Deep learning : a practitioner's approach. 1*. Beijing ; Boston ; Farnham ; Sebastopol ; Tokyo: O'Reilly, 2017. ISBN 978-1-491-91425-0..
- MITCHELL, MELANIE. *An introduction to genetic algorithms*. Cambridge: Bradford Book, c1996. ISBN 0-262-13316-4..
- HYNEK, JOSEF. *Genetické algoritmy a genetické programování*. Praha: Grada, 2008. Průvodce (Grada). ISBN 978-80-247-2695-3..
- LÝSEK, JIŘÍ a ŠŤASTNÝ, JIRI. (2014). *Automatic discovery of the regression model by the means of grammatical and differential evolution*. Agricultural Economics (AGRICECON). 60. 546-552. 10.17221/160/2014-AGRICECON. .

Přílohy

A CD se zdrojovým kódem