

Využití umělé inteligence a genetického algoritmu pro autonomní řízení

Marek Bečvář

Červen 2020

1. O projektu

Tento projekt byl zpracován v roce 2020 jako maturitní z předmětu Informatika a výpočetní technika práce v 6.ročníku Gymnázia Boženy Němcové v Hradci Králové.

Hlavní myšlenkou je využití strojového učení umělé inteligence genetickým algoritmem s cílem projetí uživatelem nastavené tratě. Výsledek snažení je postupně promítán do grafu. Schopnost by se měla, i po dosažení cíle (projetí tratí), stále rozvíjet.

2. Mechanismy programu

2.1 Umělá inteligence

Pojem **umělé inteligence** je dnes člověku prezentován v trochu zkreslené formě. Lidem se mohou vybavit některé sci-fi filmové scény s roboty, ale to není úplně přesné. Ve slově *intelligence* se doopravdy skrývá řada výpočtů zpracovávajících vstupní data na požadovaný výstup. Tyto výpočty se provádějí v předem definované struktuře tzv. **neuronové sítě**.

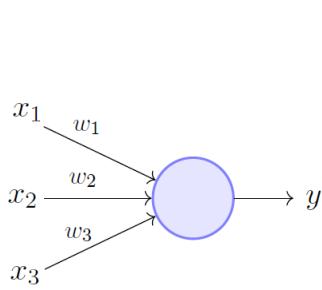


Schéma 1: Perceptron [1]

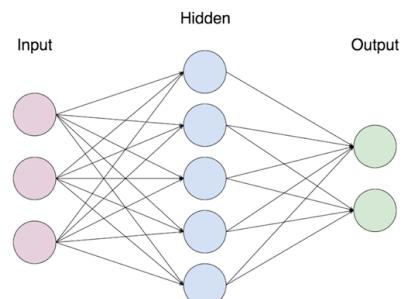


Schéma 2: Neuronová síť [2]

Neuronová síť je modelována s inspirací v mozkových neuronech. Každý bod je považován za samostatný neuron a každý neuron je propojen s řadou dalších (*podobně jako struktura mozku*) Právě tato spojení jsou důležitou funkční složkou neuronových sítí. V programu tedy vždy uchováváme číselnou hodnotu každého z nich. Tato hodnota je obvykle reálné číslo v rozsahu od -1.0 do 1.0. Neurony dělíme do vrstev (VSTUP, SKRYTÁ, VÝSTUP). Data ze sledovaného prostředí přenášíme do vstupní vrstvy a výsledek pozoruje na vrstvě výstupní. Existuje pravidlo, že každý neuron z nižší vrstvy, je spojen se všemi neuronami ve vyšší vrstvě. Nejjednodušším znázorněním neuronu je Schéma 1, skládající se z 3 vstupů a 1 výstupu. Jak se ale tvoří číselné hodnoty?

$$\sigma\left(\sum_{k=1}^n x_k * w_k\right) \quad (1)$$

$$0.5*0.1+1.0*0.5+0.25*0.3=0.625 \rightarrow \sigma(0.625)=0.625 \quad (2)$$

kde σ je *aktivační funkce*. Tyto funkce pouze upravují možný rozsah výstupních hodnot. Takto získáme finální hodnotu na jednom neuronu.

Příkladem takové aktivační funkce může být třeba **ReLU** (Rectified Linear Unit). Ta výstupní hodnoty změní tak, že kladné nechá bez změny a záporné změní na nulu. Tyto změny pomáhají při přenosu signálu mezi neurony v síti.

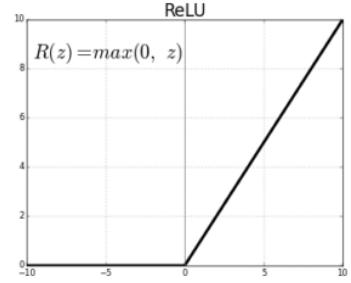


Schéma 3: RELU aktivační funkce [3]

Složitější neuronové sítě jsou už jen opakováním tohoto zavedeného postupu. Dále se pro sítě tohoto typu zavádí termín **hluboká neuronová síť** (*z ang. deep neural network*). To označuje schéma, ve kterém je použita více než jedna skrytá vrstva.

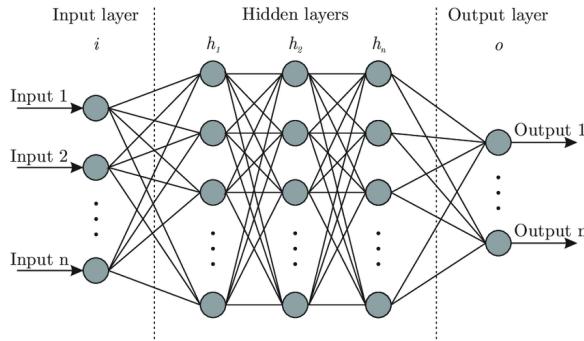


Schéma 4: Hluboká neuronová síť [4]

Při práci s neuronovými sítěmi, abychom zvětšili přesnost výpočtů, upravujeme vstupy (první řadu neuronů) do určitého číselného rozsahu. Poté necháme proběhnout všechny matematické operace uvnitř sítě a sledujeme výsledek. To, jak se program bude při daném výsledku chovat, je opět pouze na volbě autora programu.

Poté nastává **proces učení**, při kterém upravujeme hodnoty všech spojení v síti tak, aby při určitých vstupních hodnotách nastal požadovaný výstup. Pro učení existuje řada algoritmů založených na různých matematických úpravách. Pro moji síť jsem ale zvolil postup učení inspirovaný přírodou, tzv. **genetický algoritmus** ("O původu druhů" Charles Darwin - evoluční teorie).

Samotná architektura (počet neuronů v každé z vrstev) je pro funkčnost velmi důležitá a neexistuje předem daný, jednoznačně správný postup. Tato část je tedy často o spoustě drobných úprav při hledání dobře pracující sítě.

Architektura, která je použita v prezentovaném programu, byla docílena řadou testů různých sítí proti sobě.

Finální síť použitá v programu je ve schématu {6, 8, 4, 2}

5 vstupů měřících vzdálenost + 1 vstup měřící čas

1 výstup pro rychlosť + 1 výstup pro zatáčení

2.2 Genetický algoritmus

Genetický algoritmus má v informatice mnoho využití při řešení logických problémů. Ve svém projektu jsem ho použil na úpravu spojů v neuronové síti. Algoritmus je také známý svými procesy, které přímo kopírují fungování v přírodě. Teorie genetického algoritmu je založena na Darwinově teorii ("O původu druhů")

Rozlišujeme jedince, jejichž genetickou informací jsou všechny hodnoty spojů v neuronové síti. Při učení procházíme stálým cyklem, při kterém se tato genetická informace upravuje tak, aby se výsledky zlepšovaly.

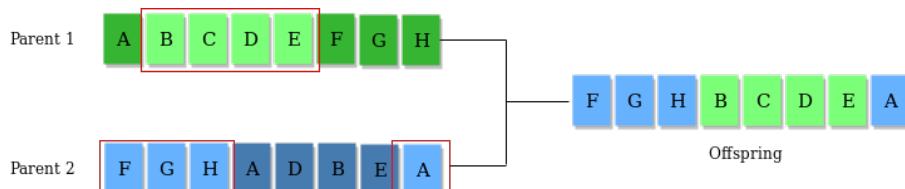


Schéma 5: Genetický crossover [5]

Cyklus genetického algoritmu

1. Určíme, jak velký počet jedinců chceme testovat (**velikost populace**)
 2. Naplníme populaci náhodně generovanými jedinci.
 - každý jedinec je abstrakcí jedné náhodné neuronové sítě
 - všechna spojení sítě tvoří genetickou informaci
 3. Všechny jedince v populaci vyzkoušíme v testovacím prostředí a zapíšeme jejich **zdatnost** v zadaném problému.
 4. Z populace pseudo-náhodně vybereme skupinu rodičů nové generace.
 - ti s **vyšší zdatností mají větší šanci** být zvoleni mezi rodiče
 - každý z jedinců může být zvolen opakováně
 5. Ze skupiny rodičů náhodně vybereme vždy dva jedince a náhodným poskládáním jejich genetických informací vytvoříme dva nové potomky (**genetický crossover Schéma 5**).
 6. Každý gen každého potomka má určitou šanci náhodně zmutovat.
 - hodnotu této náhody musíme určit
 - **pravděpodobnost mutace** se obvykle pohybuje mezi 1% - 5%
Příliš malá hodnota a populace není schopna prozkoumávat nová řešení.
Příliš velká hodnota a mutace může rozbit správná řešení.
 7. Do splnění požadavků opakujeme kroky 3 - 6, při kterých by se měli objevovat stále lepší a lepší jedinci.
-

Základní myšlenkou při křížení genetických informací je předpoklad, že ze dvou úspěšných jedinců vznike vždy potomek buď **zdatnější, nebo stejně dobrý**. V praxi toto není vždy pravda, už kvůli přítomnosti náhody ve většině krocích cyklu. Teorie ale funguje, a tak jsme tímto způsobem schopni dojít správných výsledků.

V tomto projektu je genetický algoritmus založen s populací o **50 členech**. Každý člen představuje náhodnou neuronovou síť (stejné architektury {6, 8, 4, 2}). Kvůli zdánlivé obtížnosti úkolu a velikosti sítě jsem se rozhodl pro hodnotu **pravděpodobnosti mutace = 2,5%**. Zároveň pro snazší uchování nejlepší genetické informace se nejlepší jedinec poslední generace přenáší do další.

3. Projekt - AI-Závodník

Celý program je napsán s důrazem na objektové programování, kde každá potřebnější funkce je přiřazena své třídě. Hlavní knihovna pro vykreslování je **PyGame**.

Seznam vlastních tříd

- **main** - hlavní, středová část programu, spojující všechny knihovny
- **Car** - třída závodníků se všemi funkcemi
- **Controller** - ovládání, funkce umožňující spojení vozidel a umělé inteligence
- **GeneticAlg** - třída zajišťující práci s mechanismy genetického algoritmu, zpracovávání dat celé populace testovaných jedinců
- **Boundary** - třída objektů, se kterými závodníci mohou na trati interagovat (stěny, záhytné body)
- **Ray** - třída umožňující práci s paprsky, které se používají při měření vzdálenosti (lazerový metr) - senzory vozidel

3.1 Závodníci

Centrálním bodem celého projektu jsou závodnící ovládaní umělou inteligencí. Hlavním kritériem při jejich vývoji bylo předat sítím co největší kontrolu nad vozidlem samotným.

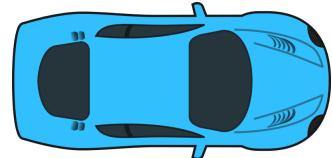


Schéma 6: Model vozidla [6]

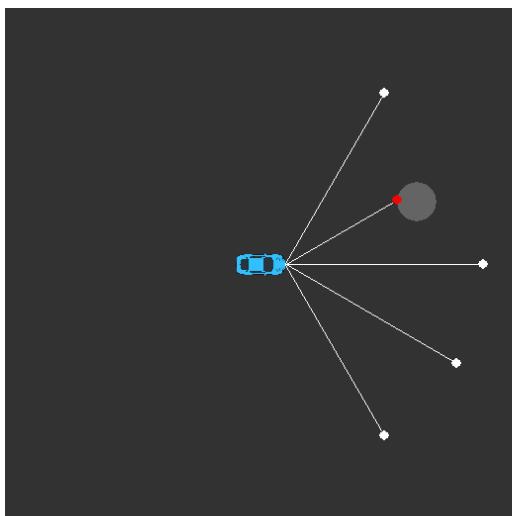
Schéma 7: Povolená ovládání vozidel

Ovládání závodníkům dává plnou kontrolu nad **rychlostí a zatáčením s proměnlivou silou otáčení**. S touto možností se závodníci při dostatečném tréninku zvládají naučit velmi optimalizované průjezdy závodními tratěmi všeho druhu.

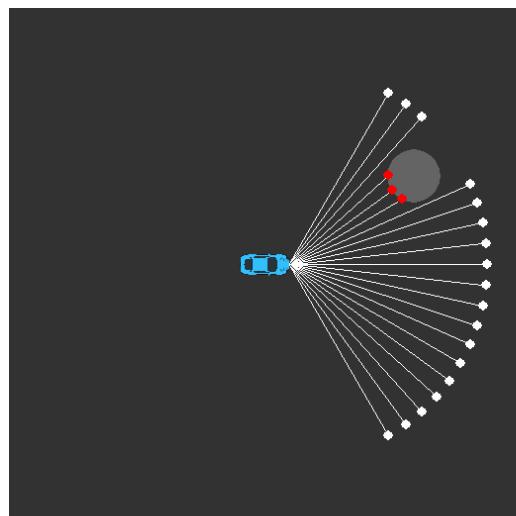
V architektuře neuronové sítě jsou do výstupní vrstvy zabudovány dva neurony. První určuje ovládání rychlosti a druhý zatáčení. Hodnoty, kterých na těchto neuronech může být dosaženo jsou upraveny do **rozsahu -1.0 - 1.0**. Tímto způsobem má ovladač velkou kontrolu při rozhodování o síle zrychlení/zpomalení nebo o úhlu zatočení.

3.2 Kontrola prostředí

Aby se ovládání mohlo rozhodovat, je potřeba dodat data z prostředí kolem jeho vozidla. Hlavním zdrojem informací (vstupní vrstva neuronové sítě) je **5 dálkových senzorů**, které měří vzdálenost předního nárazníku od stěn před ním. Paprsky senzorů jsou vysílány z předního středového bodu vozidla pod zorným úhlem 120° s určitou maximální vzdáleností, které mohou dosáhnout. Poslední informací je poté **interní časomíra vozidla**, která udává kolik proběhlo výpočtů od startu testu.



Použité řešení



Demonstrace paprsků

Schéma 8: Práce měřících paprsků

Vedle dálkových měřičů jsou ještě vozidla kontrolována proti kolizím s testovacím prostředím. Případná kolize se stěnou závodní trati by ukončila testovací jízdu daného vozidla z generace a následně by byl vypočítán jeho výsledek (zdatnost). Tuto kontrolu zajišťuje **5 senzorů po obvodu modelu vozidla**. Kontrola kolize na nich probíhá každý výpočetní cyklus.

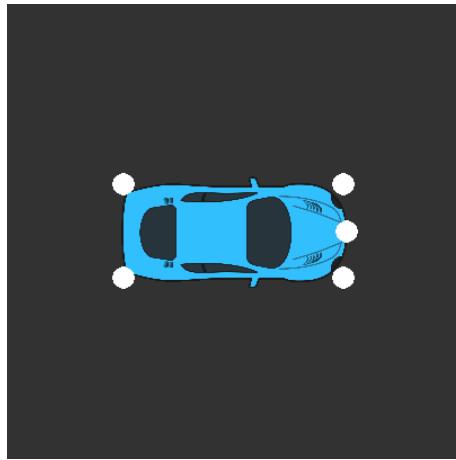


Schéma 9: Senzory kolize

3.3 Trénink

Před trénováním se musí vytvořit testovací prostředí (trať) a vložit do ní pár kontrolních bodů (START, ZÁCHYTNÉ BODY). Je potřeba zmínit, že každý trénink, i když bude prováděn na totožné trati, bude odlišný od těch předchozích. Příčinou je náhodnost celého procesu a styl jakým jsou vytvářeni náhodní jedinci pro první generaci testování.

Po nárazu všech závodníků nastává konec testování pro danou skupinu a je vypočítána zdatnost každého z nich. Zde je potřeba určit **za co bude jedinec odměněn**.

	Délka života	Počet z.b.*	Pořadí na z.b.*	Chování
Použito	0,75 b.	50 b.	100 b.	Rychlé učení celé tratě, po dokončení snaha se na trati zrychlit.
	1 b.	50 b.	25 b.	Schopnost se trať naučit, pomalá jízda. Cena života větší než průjezd záhytnými body.
	0,25 b.	50 b.	150 b.	"Nastavení závodník" - Velká rychlosť na záhytných bodech. Možné nevyřešení celé tratě. Délka života není tak cenná.

* z.b. = záhytný bod

Tabulka 1: Konfigurace odměn za výsledky na trati

Po nalezení správné konfigurace odměn a vyhovující architektury sítě se výsledky při opakovaném testování chovají vždy velmi podobně. Začátek je o pomalém postupu, kde se může vyskytnout jedinec, který je schopen se na trati dostat dál. Tento posun (**správná genetická informace**) se rychle šíří i mezi zbytek populace. Poté nastává zlomový bod, kdy se v jedné generaci objeví takový jedinec, který je schopen projet trať celou (pouze primitivní "vyhýbání se stěnám", než optimalizovaný průjezd). Tato informace se posléze začíná šířit i mezi ostatní jedince a začínají vznikat nové, lepší a rychlejší průjezdy. **V této fázi je problém (trať) vyřešen.** Toto řešení ale nemusí být to nejlepší, a tak i dále jde sledovat postupný nárůst úspěšnosti. Celý postup je na konci každé generace zaznamenáván do grafu.

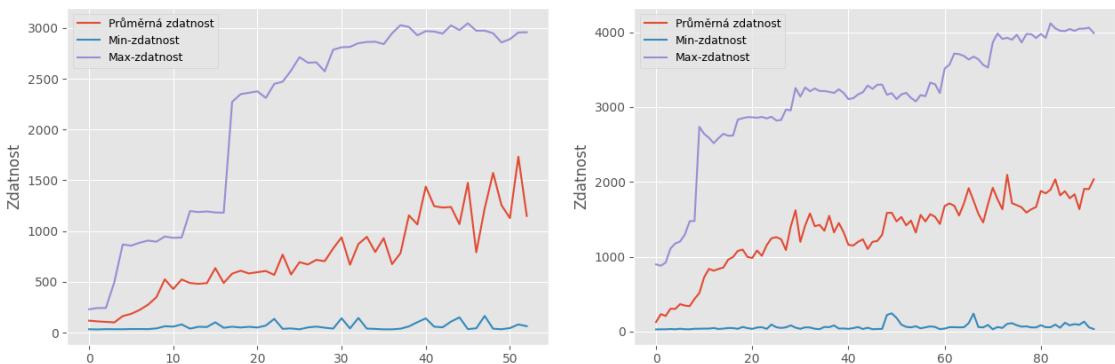


Schéma 10: Výsledky tréninků na dvou různých tratích

3.4 Inteligence

Díky velkému množství různých možností, které se při učení vyzkouší, se ojediněle může stát, že jedinci naleznou a využijí **chyby v systému**(v testovacím prostředí). I u mého programu se jedna taková chyba objevila a byla závodníky velmi rychle zneužita.

Jednalo se o problém s průjezdem záhytným bodem, kdy zprvu v programu nebylo hlídáno, zda danný bod byl již jedincem překonán nebo ne. To vedlo k rychlému zneužití, protože jak již bylo řečeno, za průjezd záhytným bodem se obrdží odměna. Toto byl velice rychlý a efektivní způsob zisku bodů.

Schéma 11: Vychytralost

4. Závěr

I když se může zdát, že genetický algoritmus je plnohodnotnou variantou pro umělé inteligence a strojové učení, jedná se pouze o praktické uplatnění správné teorie, která je ale založena na spoustě náhodných procesů. Příroda tomuto postupu dává potřebný čas k nalezení téměř perfektních řešení každého problému. Ale v počítačové vědě, i když vše lze zvládat ve velké rychlosti, příroda má navrch. Proto existují různé systémy pro strojové učení, založené především na matematických postupech, které již náhodně nebádají po správném řešení, ale jdou cílevědomě přímo k němu.

Tento projekt může být pěknou ukázkou toho, jak jsou nové technologie přístupné pro veřejnost. O problematiku umělých inteligencí se zajímam již přibližně 3-4 roky. Zájem o programování mi pak pomohl posunout tyto teoretické vědomosti do praxe skrz podobné programy.

Obsah

1	O projektu	1
2	Mechanismy programu	1
2.1	Umělá inteligence	1
2.2	Genetický algoritmus	3
3	Projekt - AI-Závodník	5
3.1	Závodníci	5
3.2	Kontrola prostředí	6
3.3	Trénink	7
3.4	Inteligence	8
4	Závěr	9

Seznam obrázků

1	Perceptron [1]	1
2	Neuronová síť [2]	1
3	RELU aktivační funkce [3]	2
4	Hluboká neuronová síť [4]	2

Seznam literatury

- [1] SAPORITO, G. What is a perceptron?. Perceptron model [online]. 2019 [cited 2020-06-10]. Available from https://miro.medium.com/max/1290/0*LJB08Ub7zK_SKMog.
 - [2] NURAMI PUTRI, O. Titanic Prediction with Artificial Neural Network in R.. ANN model [online]. 2018 [cited 2020-06-10]. Available from https://cdn-images-1.medium.com/proxy/1*RGV6Bb3ChmVWsA8Q6Qth6Q.png.
 - [3] SARKAR, K. ReLU : Not a Differentiable Function: Why used in Gradient Based Optimization? and Other Generalizations of ReLU.. [online]. 2018 [cited 2020-10-06]. Available from <https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec>.
 - [4] BRE, F. Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks. Artificial neural network architecture (ANN i-h 1-h 2-h n-o). [online]. 2017 [cited 2020-06-10]. Available from https://www.researchgate.net/profile/Facundo_Bre/publication/321259051/figure/fig1/AS:614329250496529@1523478915726/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o.png.
 - [5] KUMAR, A. Genetic Algorithms. Operators of Genetic Algorithms [online]. 2018 [cited 2020-06-10]. Available from <https://media.geeksforgeeks.org/wp-content/uploads/genetic-algorithm1.png>.
 - [6] Clipart personal collection - Clipart Art [online]. Dostupné z: <https://clipartart.com/images/car-clipart-sprite-sheet-3.jpg>