UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

Úvod do Umelej Inteligencie

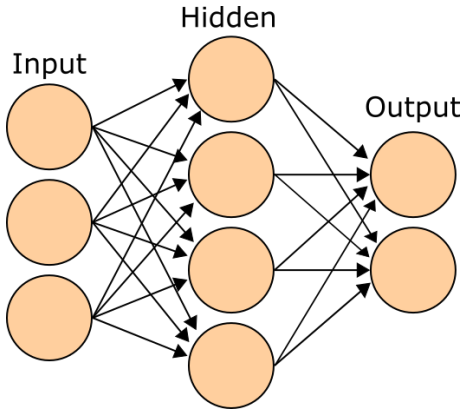
Cvičenie 10 - Geneticky trénované neurónové siete

November 23, 2022

10. VIACVRSTVOVÁ NEURÓNOVÁ SIEŤ TRÉNOVANÁ POMOCOU GA

Budeme programovať viacvrstvovú neurónovú sieť, ktorej váhy budú trénované pomocou genetického algoritmu. Táto sieť bude potom ovládať autíčko jazdiace po jednoduchej dráhe.

Príklad modelu:



Okrem vstupnej a výstupnej vrstvy neurónov má náš model aj ďalšiu vrstvu, tzv. skrytú. Takúto sieť si môžeme predstaviť ako spojenie dvoch jednovrstvových sietí. Najprv sa vypočíta aktivácia skrytej vrstvy (ako výstup prvej jednovrstvovej siete), tá je potom akoby vstupom do "druhej vrstvy", ktorej výstup je výstupom celej siete.

Takáto sieť má teda dve matice váh - W^{in}, ktorá spája vstup so skrytou vrstvou, a W^{out}, ktorá spája skrytú vrstvu s výstupom.

K vstupu aj skrytej vrstve sa opäť pridáva aj bias, veľkosť matíc je teda $(hid \times (n+1))$ a $(m \times (hid+1))$, kde n je dimenzia vstupu, m je dimenzia výstupu a hid je počet neurónov na skrytej vrstve.

Počítanie výstupu:

Výstup z dvojvrstvovej siete vypočítame v dvoch krokoch, najprv vypočítame aktiváciu skrytej vrstvy:

$$\boldsymbol{h} = f_{hid}(\boldsymbol{W^{in}x'}),$$

kde funkcia f_{hid} je aktivačná funkcia skrytej vrstvy, napr. opäť logistická sigmoida a x' je vstup s biasom. Následne môžeme vypočítať výstup siete:

$$\boldsymbol{y} = f_{out}(\boldsymbol{W^{out}h'}),$$

funkcia f_{out} je aktivačná funkcia výstupu a h' je aktivácia na skrytej vstve rozšírená o bias. Pri tejto úlohe budeme používať **hyperbolický tangens** ako výstupnú funkciu. Tá zabezpečí, že všetky výstupy siete budú v rozsahu (-1, 1).

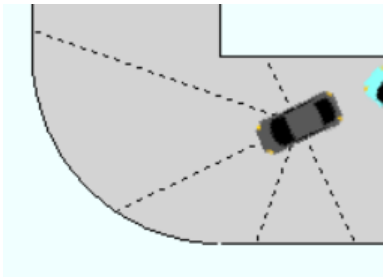
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Trénovanie:

Trénovanie pomocou genetického algoritmu spočíva vo zvolení rodičov (jedinci s najlepšou fitness), z ktorých sa krížením a mutáciou vytvoria potomkovia. Spolu s rodičmi tak tvoria ďalšiu generáciu.
V našom prípade po inicializácii jedincov ich fitness nepoznáme, preto najprv potrebujeme spustiť simuláciu - t.j. autíčka sa budú pohybovať podľa ich sietí až kým im nevyprší čas alebo nenarazia na okraj dráhy. Tým zistíme ako ďaleko jednotlivé autá dokážu zájsť. Čím ďalej auto zájde, tým väčšiu fitness dosiahne.

Pri tejto úlohe môžeme čakať, že prevažná väčšina autíčok (hlavne v prvých generáciách) sa bude pohybovať úplne náhodne. Z tohto dôvodu za rodičov nebudeme považovať najlepšiu polovicu jedincov ako sme to robili na štvrtom cvičení, ale oveľa menší podiel z celkového počtu jedincov (default 4 z 16, ale môžeme sa s týmto parametrom pohrať). Potomkov následne generujeme krížením náhodným výberom dvojíc rodičov, až kým naša nová populácia nedosiahne pôvodnú veľkosť. Táto zmena nám výrazne pomáha dosiahnuť cieľ rýchlejšie.

Ako však dokážeme auto počas simulácie ovládať? Na to slúži neurónová sieť, ktorej vstupom sú informácie zo senzorov (spolu s inými dôležitými údajmi) a výstupom sú akcie udávajúce, ako sa má auto v ďalšom kroku pohnúť. V tejto úlohe budeme pracovať s autami, ktoré majú päť senzorov. Všetky majú začiatok v strede auta a smerujú tak, aby zachytávali podstatnú časť dráhy v smere jazdy dopredu. Ich maximálna dĺžka môže byť 200, ale to sa stane iba v prípade, že daným smerom nie je žiadna prekážka. Ukážka senzorov:



Úloha (2b):

Do pripravenej kostry *cars.py* doprogramujte kľúčové časti na to aby sa autá postupne naučili jazdiť po dráhe bez nabúrania.

- *MyCar (__init__)*
  - inicializujte chromozóm *self.chromosome* náhodnými číslami (ideálne blízko nuly).
  - následne nadizajnujte neurónovú sieť (vytvorením aspoň dvoch váhových matíc) a použite skôr vytvorený chromozóm na naplnenie váh tejto siete.
- *compute_output()*
  - vytvorte vstup do siete. Vstupom budú dĺžky senzorov, spolu s aktuálnou rýchlosťou a točením (aktuálnym "uhlom predných kolies") auta. Tieto dĺžky nezabudnite prešk álovať tak, aby do neurónovej siete nevstupovali príliš veľké čísla. Z pôvodného intervalu dĺžok senzorov <0, 200> teda vytvorte menší (napr. <0, 10>).
  - údaje o senzoroch nájdete v premennej self.sensors, ktorá je zoznamom jednotlivých senzorov. Každý prvok tohto zoznamu má tvar $[[x1, y1], [x2, y2]]$ - začiatočný a koncový bod senzoru. Z tohto vieme ľahko zistiť dĺžku senzoru.
  - ďalšími zložkami vstupu sú aktuálna rýchlosť autíčka (self.speed) a aktuálne točenie (self.turn).
  - pri vytváraní vstupu do siete nezabudnite na bias!
  - v tejto funkcii taktiež vypočítajte a vráťte výstup siete. Presné vzťahy na výpočet výstupu sú uvedené v texte vyššie. Počet neurónov na výstupe v našom prípade bude 2. Prvý výstup siete dáva inštrukcie na zmenu rýchosti auta (zrýchliť alebo spomaliť), pričom druhý výstup mení otáčanie auta (doprava/doľava).
- *update_parameters()* - použite premennú *instructions*, ktorá obsahuje výstup siete z predošlého kroku (compute_output) na aktualizáciu smeru kolies a rýchlosti auta.

Bonus (1b):

Implementujte trojvstvovú sieť (teda použite tri váhové matice) na riadenie autíčok. Aktivačné funkcie na skrytých vrstvách si zvoľte podľa seba, no na výstupnej vrstve zachovajte hyperbolický tangens.
Následne porovnajte úspešnosť autíčok pri používaní dvojvrstvovej vs. trojvrstvovej siete. Pohrajte sa aj s hyperparametrami (hlavne počet neurónov, aktivačné funkcie), no úspešnosť veľmi ovplyvňuje aj počet jedincov v generácii, výber k rodičov, pravdepodobnosť mutácie, atď. Preto sa môžete pohrať aj s týmito parametrami.
Výsledky tohoto porovnania stručne zhrňte v krátkom readme.txt alebo ako komentár v kóde.