Simulated Racing Controller

[The artificial drunken driver]

Masoumeh Bakhtiariziabari

Nhung Nguyen

A basic controller - NeuralNet

- Dataset: the one on the blackboard
- Features (22):
  - $\sqrt[2]{speed_x^2 + speed_y^2 + speed_z^2}$
  - Distance from center
  - Angle
  - Distances From Edge
- Targets (2):
  - Acceleration: We calculated value of brake based on predicted value of Acceleration
  - Steering: we categorized steering to 7 classes.
- Multi-layer Perceptron classifier:
  - optimizes the log-loss function using LBFGS and L2 regularization
  - 4 hidden layers with respectively 100, 100, 20, 2 neurons

Controller that races other cars – NEAT (1/3)

- NEAT-python library: the well structured framework to apply NEAT algorithm with possible modification and extension.
- Features (26): the NN controller features + 4 dist_from_opponent
  - $\sqrt[2]{speed_x^2 + speed_y^2 + speed_z^2}$
  - Distance from center
  - Angle
  - Distances From Edge
  - Four opponent sensors on the 4 corner areas of the car (-140, -50, 40 and 130 degrees).
- Targets (2):
  - Acceleration
  - Steering

Controller that races other cars – NEAT (2/3)

- Trained on Alborg track
- hidden = 0, inputs = 26, outputs = 2
- 60 generations, 50 genomes per generation
- Recurrent neural network (RNN) with clamped non linearity activation function.
- Mutation:
  - The probability of adding or removing a node = 0.2
  - Node Aggregation mutation rate = 0.0
  - Bias mutation rate = 0.7
  - Weight mutation rate = 0.8
  - Enabled mutation rate = 0.01

Controller that races other cars – NEAT (3/3)

$$fitness = max(1, -2.T_{out} + 20.S_{avg} + 5.dist - 0.5.str_{dev} - 0.05.d - 100.r + c)$$

- Where:
  - T_out : number of game tics (20ms) that the car is out of the track ;
  - S_avg : average car speed over all past game time tics till now.
  - dist : distance raced
  - str_dev : standard deviation of steering from the beginning of the race
  - d : total damage
  - r : car´s ranking
  - c :is a constant(1200)
- The max running time for every genome = 60 seconds.
- Termination rules for training:
  - If the car **stuck**: terminate the training of that network & **reduce 1000** points from fitness.
  - If the **damage** is more than 10'000: just terminate the training
  - If the **fitness** is less than 500: since its beginning assignment is 1200, due to c in formula
  - If the car could **survive** in the race for 60 seconds: **reward** it by adding **1000** points to its fitness value, then terminate it to shorten the training time.

Team based racing controller – Swarm (1/2)

- NEAT trainer's best, globally fitted network as the basis of the swarm controller.
- It tries to protect one of our cars with the help of the other one, by attacking the opponents.
- Features and Targets as the Neat-Controller
- For Communication between friends, write in the 3 files:
  - Ports: both cars write their ports in this
  - Car1_state: car1 rank and distance from start
  - Car2_state: car2 rank and distance from start

Team based racing controller – Swarm (2/2)

- Winner car, loser car
- Rules followed by each car:
  - **Act aggressively** if it is the **loser** one and it's distance to winner is between 5 to 15 meter:
    - Hit the closer opponent on the left or right side.
  - **Exceed brake** and make **acceleration zero**, if it is the **loser** car and distances between them is less than 5 meters
  - **Do normal** and run as **NEAT controller**, if one of these conditions are true:
    - the friend is not available yet.
    - the friend car is killed because of crash and etc.
    - the car is **winner** and distance between it and its friend is more than 15 meter.
  - It should go to the **recovery mode** if the car get stuck. (same recovery mode for all of trained network.)

Controllers Testing

- 4 circuits namely Alpine-2 , Brondehach , E-Track and Olethros-Road with 3 races for each circuit.
- 9 other opponent teams of two cars namely: inferno, berniw, berniw two, tita,lliaw, InHist, bt, damned, Olethros.
- The starting positions are shuffled randomly to ensure equal opportunity for every team.
- Results:
  - NeuralNet driver, in combination with recovery mode, is able to finish most of the circuits
  - Both NEAT and Swarm controllers works well on narrow and windy roads.
  - Swarm controller shows ability to slow down nearby opponents.

Table 1. Controller with friends test against opponents and racing on 4 selected circuits. Statistics reported are the average of team's best ranking computed over 3 races

Controller	Alpine-1	Alpine-2	CG-track-3	Olethros	Total
NeuralNet	13.3	16.3	18	18.3	18.1
NEAT	14.3	14.6	12.3	18	14.8
Swarm	3.3	2.3	12	12	7.4

Conclusions

- Neuro evolutionary algorithms achieve better results with simpler structures compared to neural networks trained with stochastic gradient descent.
- Optimal swarm behaviors can further improve over individuals.