Machine learning in simple car racing game

My task is to create a simple, 2D, view from above car racing game. The only goal in this game to achieve the best result in time as possible. The player can wheel the car and can speed up or slow down. If the player goes out from the race track, the game is over.

It is simple at this point, but the game also has an AI. The goal of the AI is to learn the "optimal" tracing on the track, based on the the past tracings of the human player.

I'm considering some functions, these can be used as training sets:

- the tracing of the player as the function of the topology of the track
- the speed of the player as the function of position of the track

Of course other functions or information can be useful.

What learning methods can is use to produce a valid track. My goal is not to beat the human player, but to only get to the end of the track. :)

machine-learning | artificial-intelligence

edited Jul 16 '13 at 18:43

asked May 15 '13 at 16:23

WonderCsabo
1,056 1 14 22

1 Why learning, when you can direct compute the shortest path from start to goal? – Thomas Jungblut May 15 '13 at 16:48

Nope, i have to use machine learning in this task. :(– WonderCsabo May 15 '13 at 17:17

Was there something wrong with my answer? Does that not work for you? – Daniel May 15 '13 at 17:20

I commented to your answer. I do not know where my comment is. :S – WonderCsabo May 15 '13 at 17:29

add comment

3 Answers

I am mostly just brainstorming here, but:

as I understand, your situation is something like this:

- you have to go from Start to Finish
- you don't have to repeatedly do this and achieve better and better "lap" times
- you have to incorporate any kind of learning algorithm (is there a specification what learning algorithms/intelligent algorithms are allowed?)
- you know the map, including Start and Finish (?)
- the map is grid based or can be easily represented as a grid (?)

In this case, a very simple model is as follows:

- define some simple features (vector) describing your current (or neighboring) position is (e.g. angle from Finish, distance from the edge of the track)
- define a goodness feature (e.g. distance from Finish)
- in each step, make a decision, which direction will you move (Left, Righ, Forward, Backward)

So you have a set of input features and a decision problem.

- you can define a fuzzy control system that gives you the best direction. (rules like IF I am close to border THEN move away, IF I'm headed to the Finish AND far enough from border, THEN move forward)
- even simpler, you can construct a decision tree
- you can construct an SVM or Neural Network that chooses the next step

(these are not concrete implementation ideas, that depends on what you are choosing)

Choice heavily depends on what tools are you using (Matlab, C++, Python, etc) and which learning algorithm are you familiar with. I suggest choose the one you know the best, and try to fit a model to that.

answered May 16 '13 at 10:55

Oscee
388 2 18

Thanks! Sorry for my huge delay for accepting your answer. I implemented this in Java, and constructed a decision tree. – WonderCsabo May 27 '13 at 21:18

add comment

Maybe you could try a Neural Net?

"In most cases a neural network is an adaptive system changing its structure during a learning phase. Neural networks are used for modeling complex relationships between inputs and outputs or to find patterns in data."

http://en.wikipedia.org/wiki/Artificial_neural_network

answered May 15 '13 at 16:25

Daniel
668 8 22

Yes, i thought of NNs, but i have only basic knowledge of them, and i do not know how can is construct such a net. I was hoping for a more concrete answer. :) – WonderCsabo May 15 '13 at 17:30

1 Ok what about this: break the track into several "sections", where each section is a turn, straightaway, etc. Run a bunch of human tests, storing the results for each section. When you have that data, pick the best from each area and use that. They will likely not mesh perfectly, but that shouldn't be too hard to fix. FYI, this isn't machine learning but you say you need to use machine learning and also to use runs from humans. Those are two independent things. Which do you need to use? – Daniel May 15 '13 at 19:16

@WonderCsabo It wouldn't be the first time that ANN are used in racing games. Colin McRae rally 2 used it. Here some links that you might find useful. togelius.blogspot.ca/2007/08/… ai-junkie.com/misc/hannan/hannan.html generation5.org/content/2001/hannan.asp – Pedrom May 16 '13 at 14:54

Very interesting links indeed. Thanks! I never thought that ANNs are used in simple commercial products like games. – WonderCsabo May 27 '13 at 21:11

@Daniel Thanks! I used this idea in my final implementation, it's very simple, but works. – WonderCsabo May 27 '13 at 21:13

show **1** more comment

I think it's more important to figure out how to represent your environment and "probable" action for that situation, the model is just a thing to link them.

In my opinion, you can try features like "distance from left/right-margin of road to you car", "current car speed" and "angle differences between car's & road's orientation" and more. These will be your model inputs.

Then next is related them to available actions of car, "turn left"/"turn right"/"speed up/down","game continue/over" or something else. These will be your model output.

If you're about to use a NN, I come up with two ways to get your model trained. 1. You can play your game, and make your program log down inputs any time when an action is apply to the car. 2. Make an algorithm that drive the car randomly to sample the training data, and choose effective ones to train your model.

I'm not familiar with reinforcement learning, but i still think it's related, you can also dive into that and have a try.

answered May 16 '13 at 6:45

JeromeZhao
21 4

Reinforcement learning was the first thing I thought of when reading the problem. – Emile May 22 '13 at 9:55

add comment

Not the answer you're looking for? Browse other questions tagged machine-learning artificial-intelligence **or ask your own question.**