Memory Based Reasoning in Artificial Intelligence for Racing Cars

Sahitya Potluri, Vignesh Krishnakumar, Abhiram Ravi

Indian Institute of Technology, Madras

Abstract. Computer games have been a playground for academicians in the experimentation of several Artificial Intelligence techniques. Game Bots (a.k.a Game AI) borrow several techniques from the broader field of AI and Machine Learning in bringing human-level cognitive processes to the non-player characters/bots in the game, providing a strong platform for the simulation of AI techniques that are applicable to the real world. In this course, we experiment with the application of Case-based reasoning in the building of an AI-driven car for the Open Source Car-Racing game *Speed-dreams*(a.k.a TORCS). TORCS is a famous open source racing game project, which has been used extensively by academicians for game AI research.

Introduction

Virtual Gaming Environments are killer apps for the application of Human-level Artificial Intelligence. Although AI techniques have been applied to games such as Chess, Checkers etc., the traditional techniques fail in the domain of more complicated games such as Multiplayer Role-playing games or real time strategy games because the decision space is too vast for making decisions in real-time. Traditional Machine learning techniques, where we predict an action based on the state, also fail to perform, since we would require a dataset that is exponential in the number of elements that describe a state in the game. This introduces the need for an efficient memory model and additional reasoning to adapt previously seen instances to a newly encountered scene. We tackle the situation of a Racing-car scenario, where we would like an AI to learn to drive by observing other expert cars play. In our approach, the AI builds a case base by observing the actions of an expert racer and uses this case base to decide actions given a new scenario. The rest of this paper is organized as follows - We first present a detailed outline of our approach. We then present the details of implementation on the Speed-dreams simulator. We then evaluate the performance of our approach by running our AI car on an *unseen* racing track.

Case Based Reasoning for Racing Cars

The goal of our approach is to build an AI racing car that learns to play by studying the actions of expert racers on various scenarios. By exploiting the paradigm of case-based reasoning, our AI learns to conquer the road with the following three steps

1. **Case Base Construction** - We first let expert racers run on a variety of different racing tracks, and log information regarding the actions taken by these experts for different state parameters. The problem attributes include some crucial parameters involving the internal and external states of the car along with parameters of the

portion of the track local to the position of the car. The lesson attributes include the control parameters of the car i.e the decisions takes by the expert racer during the corresponding problem scenario. This is done by running a single expert car (denote **E**) on a variety of training tracks (denote **TT**). For reasons of simplicity, we work with a single car model

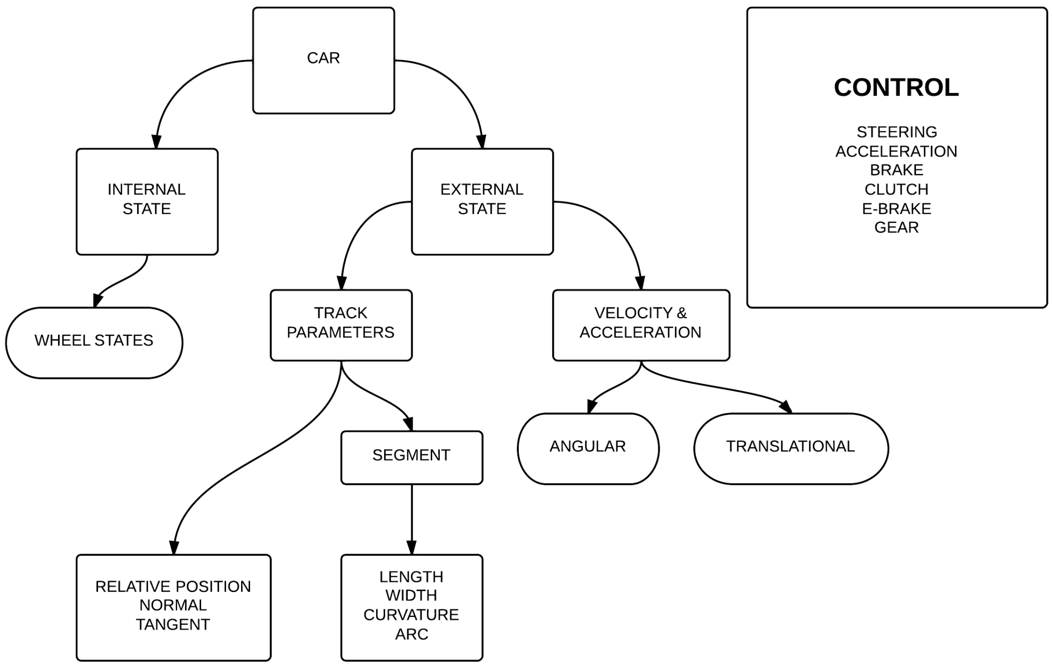

Fig. 1. An overview of the case base, with the comprehensive list of attributes (both problem and lesson)

2. **Similarity Computation** - Since we would like to run our AI car on a previously unseen track, we would like to approximate the newly encountered situation to seen data from the case base, and obtain the *nearest* neighbors, before adapting it to make our decision. Computing the nearest neighbors requires the description of a similarity measure between two problems. We experiment with several local and global similarity measures including the default one provided by the myCBR framework. The global similarity measure is a weighted average of all the local similarity measures between the individual attributes. The weights that we used were inferred from the importance of the physical interpretation of the attributes in making the control decisions of the car.

3. **Solution Adaptation** - It is not practically feasible to directly use the solution of the nearest neighbor of the problem in the case base, as the situation may be semantically different, although the closest neighbor by definition. We can improve the performance of our AI car by adapting the solution of the nearest matches using

one of several adaptation techniques. In this light, we build and train a neural network using the top thirty nearest neighbors of the query problem, and use the solution provided by the neural network to make our final decision.

Implementation & Evaluation

Speed-Dreams

Speed-dreams is an open-source 3D Racing Car Simulator, a fork of the TORCS project with improved physics and car models/tracks. We hack this simulator to log data, and also to build our AI bot that plays the game. The game source provides a separate module to implement custom AI game bots, and documents the necessary interface functions that need to be implemented for the integration of a custom bot into the game.

Fig. 2. A screenshot of the speed-dreams racing car simulator

Our implementation process is multi fold. In order to build our case base, we first modify the source of an existing AI bot to log the corresponding game state and control attributes. We then let the AI bot play over various different racing tracks, and dump all the required information into a *csv*. Once the information has been logged, we use myCBR to import the *csv* and build the case base. We experiment with similarity measures using the same software. For adaptation, we query for the nearest neighbors using the API provided by myCBR, and implement the neural network using the OpenCV Machine Learning Module.

To put it all together, after setting up the above, we build an AI bot that records the game state parameters at every instant, queries the constructed case base using the myCBR API to obtain the nearest neighbors, trains a neural network using these neighbors, and then obtains the final decision to make for the given game state by feeding it into this neural network.

Discussion

Note that we don't learn across cars, but only across tracks. A major assumption we made is that there are no opponents on the track. A possible extension of the case base is

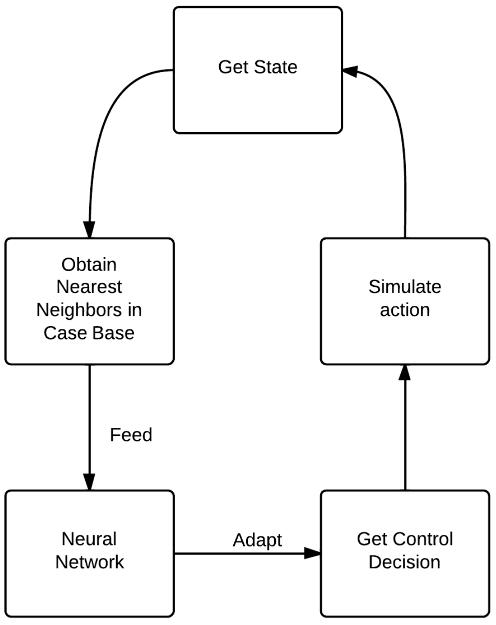

Fig. 3. Flow Diagram

to include relevant information about neighboring cars (eg. distance to nearest car etc.) to make more informed decisions. We assume that the tracks are of the same category, but different geometric structure only i.e we learn across different tracks of the same type *road* as opposed to learning on a road track, and playing on a *dirt* track. We spent a significant amount of time in analyzing the influence of various parameters that the AI car had access to, involving those of the car and the track. The attributes in the problem space of the case base are those obtained after pruning the insignificant parameters.

Conclusion

In this exercise, we experiment with the application of Case-based reasoning in cruise control for AI Racing cars.