Tutorial 3

In this document we explain how to use the submission template and the contest package. We also give you an example of how to generate data and how to save the state of your neural network.

An example of a controller with everything explained in this tutorial can be found here: http://mac360.few.vu.nl:9090/cicourse2016/resources/ExampleController.7z

The submission template consists of the following 4 classes:

- DefaultDriver
- DefaultDriverAlgorithm
- DefaultDriverGenome
- DefaultRace

I. DefaultDriver

DefaultDriver contains methods that can be used to control a car. This class extends AbstractDriver which is an abstract class in the contest package.

DefaultDriver contains the following 3 methods:

- controlWarmUp(Action action, SensorModel sensors)
- controlQualification(Action action, SensorModel sensors)
- controlRace(Action action, SensorModel sensors)

These 3 functions can be used to control different types of races which you can set up on your computer (WARMUP, QUALIFYING and RACE).

It is also possible to control the car directly with the control method. The current implementation (in AbstractDriver) is:

```
public void control(Action action, SensorModel sensors) {
    if(this.getStage() == Stage.WARMUP) {
        this.controlWarmUp(action, sensors);
    }

    if(this.getStage() == Stage.QUALIFYING) {
        this.controlQualification(action, sensors);
    }

    if(this.getStage() == Stage.RACE) {
        this.controlRace(action, sensors);
    }
}
```

You can rewrite this function with you own code (don't forget @Override):

```
@Override
public void control(Action action, SensorModel sensors) {
     action.steering = getSteering(sensors);
     action.accelerate = getAcceleration(sensors);
     action.brake = getBraking(sensors);
}
```

To enable automated clutch, automated gearbox, automated recovering and ABS, you can use the following code in your DefaultDriver class:

```
import cicontest.torcs.controller.extras.ABS;
import cicontest.torcs.controller.extras.AutomatedClutch;
import cicontest.torcs.controller.extras.AutomatedGearbox;
import cicontest.torcs.controller.extras.AutomatedRecovering;

this.enableExtras(new AutomatedClutch());
this.enableExtras(new AutomatedGearbox());
this.enableExtras(new AutomatedRecovering());
this.enableExtras(new ABS());
```

The method loadGenome(IGenome genome) gives you an example of how to load a trained neural network from memory. This function has as input a serializable class called Igenome which is just an empty class with a serialVersionUID. The class DefaultDriverGenome from the submission template is an example of such a class.

Since many groups found it unclear, we will show you how to write your own code to save the state of your neural network in a .mem file.

Let's say you have a class NeuralNetwork which you will be using to control your car with. You have generated data and trained the neural network to drive on a circuit. The weights, parameters and other objects after training need to be stored somewhere. To do this, you have to make this class Serializable and declare a serialVersionUID by declaring a field named "serialVersionUID". From within this class you can save the state of the object as shown in the following example:

- **Create a NeuralNetwork class and a NeuralNetwork object:**

```
public class NeuralNetwork implements Serializable {
      NeuralNetwork(){ }
      private static final long serialVersionUID = -88L;
}

 NeuralNetwork neuralNetwork = new NeuralNetowrk();
```

- **Train your network**
- **Save state of the class (NeuralNetwork) (http://www.java2s.com/Tutorial/Java/0180__File/Savingandrestoringthestateofclasses.htm)**

```
ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("path/to/memory/mydriver.mem"));

out.writeObject(this);
```

- **Restoring objects from a serialized state (www.javacoffeebreak.com/articles/serialization/)**

  You can use the following code to restore the state of a neural network that has been trained earlier. You can call this code from your DefaultDriver class and use the restored state of your neural network to control your driver.

```
// Read from disk using FileInputStream
FileInputStream f_in = new FileInputStream("memory/mydriver.mem");
```

```
// Read object using ObjectInputStream
  ObjectInputStream obj_in = new ObjectInputStream (f_in);

// Read an object
  NeuralNetwork neuralNetwork = (NerualNetwrok) obj_in.readObject();
```

II. DefaultRace

The class DefaultRace was included to give you an example of how to create different types of races. You can customize this class as you wish. The following example shows you how to set up a custom race:

```
Race race = new Race();
race.setTrack("road", "aalborg");
race.setTermination(Race.Termination.LAPS, 2);
race.setStage(Controller.Stage.RACE);
race.addCompetitor(new DefaultDriver());
Boolean withGUI = true;
RaceResults results;
if(withGUI) {
    results = race.runWithGUI();
} else {
    results = race.run();
}
```

III. DefaultDriverGenome

This class is meant as an example of how to create a Serializable class. You can create your own Serializable class as follows:

```
public class NeuralNetwork implements Serializable {
       NeuralNetwork(){ }
       private static final long serialVersionUID = -54534L;
}
```

IV. DefaultDriverAlgorithm

DefaultDriverAlgorithm represents the main class that you will be using to train a controller on your computer. You can change the main method and any other method in this class as you wish.

V. FAQ:

How to generate training data?

You can set up a race with your DefaultDriver and use the built-in methods from the class DriversUtils to get training data. In your DefaultDriver class you have to implement the following method:

```
public void controlRace(Action action, SensorModel sensors) {

    action.steering = DriversUtils.alignToTrackAxis(sensors,0.5);

    if(sensors.getSpeed() > 60.0D) {
        action.accelerate = 0.0D;
        action.brake = 0.0D;
    }

    if(sensors.getSpeed() > 70.0D) {
        action.accelerate = 0.0D;
        action.brake = -1.0D;
    }

    if(sensors.getSpeed() <= 60.0D) {
        action.accelerate = (80.0D - sensors.getSpeed()) / 80.0D;
        action.brake = 0.0D;
    }

    if(sensors.getSpeed() < 30.0D) {
        action.accelerate = 1.0D;
        action.brake = 0.0D;
    }
    System.out.print(action.steering+"action.steering");
    System.out.print(action.accelerate+"action.accelerate");
}
```

A second option is to use a keyLogger or a keyListener to control your driver with your keyboard. (http://www.java2s.com/Tutorial/Java/0260__Swing-Event/HowtoWriteaKeyListener.htm)

You can train your neural network while you are in a race or you can choose to save the data to a file and use that data to train your neural network after you finish racing. To train a good controller you will need to generate enough data from several circuits.

How to create a valid jar file?

The jar file should contain the following folders and files:

- A folder with the name META-INF which contains the MANIFEST.MF.
- The MANIFEST.MF should contain the following :
    - Driver: path.to.DefaultDriver
    - Main-Class: path.to.DefaultDriverAlgorithm
    - Class-Path: . YourPackages.jar (if you are using any external libraries).
- A folder with the name “memory” which contains a .mem file.
- You code (.java and .class).

How change the view during a race with GUI?

- Use the keys F1 - … - F12