COSC 522 Project 5

Reinforcement Learning in Self-Driving Cars

Yangsong Gu

11/23/2021

Objectives

The objective of this project is to have an in-depth understanding of supervised vs. unsupervised learning by a simple realization of reinforcement learning.

Simulator used

CARLO is a 2-D driving simulator, a low-budget version of the popular CARLA. Two scenarios were provided in this simulator, an intersection and a circular road. You task is to develop reinforcement learning techniques for steer control to avoid vehicle crashes.

Task 1

Plot the convergence curve (Figure 1. Only need to provide one curve after many trials). Apply the trained BPNN on the pima test set. Generate a bar chart (Figure 2 or a table) summarizing the classification accuracy on pima using various classification approaches you've practiced so far.

Task 1.1 preprocess

Firstly, we randomly divide each category of the pima training samples into training (70%) and validation (30%). After that, 70% of class 0 and 1 sample were concatenated together to compose a new training set. The rest of training set composes the validation set. It should be noticed that the original dataset is standardized to make the input consistent with previous algorithm, we are training the neural network though. **Secondly**, to perform the Nielson's code directly, the training, validation and test set were prepared as the input format of Nielson's code. Specifically, the training label was vectorized while the validation and test set label were not.

Task 1.2 Tune the network

Using the strategy discovered in project 4, I set the network layers from [7, 4, 2], [7, 8, 4, 2], [7, 16, 8, 4, 2], and other geometric progression fashion. Finally, the network structure is set as [7, 256, 128, 64, 32, 16 ,8, 2]. For other parameters, the minibatch size is set as 1 which is stochastic Gradient descent, the epoch is 250, learning rate is 0.7. Noticed that these values are determined after multiple trails.

Task 1.3 Performance of validation set.

Figure 1 demonstrates the convergence curve of the validation set, with y-axis being the overall accuracy. As we can see in figure 1, the overall accuracy fluctuates a lot at the beginning of training, while after 160 epochs, the accuracy tends to be stable and finally, the accuracy stabilized at 63.93%. By calling the evaluate function on test set, I got the overall accuracy of test set, which is 73.80%

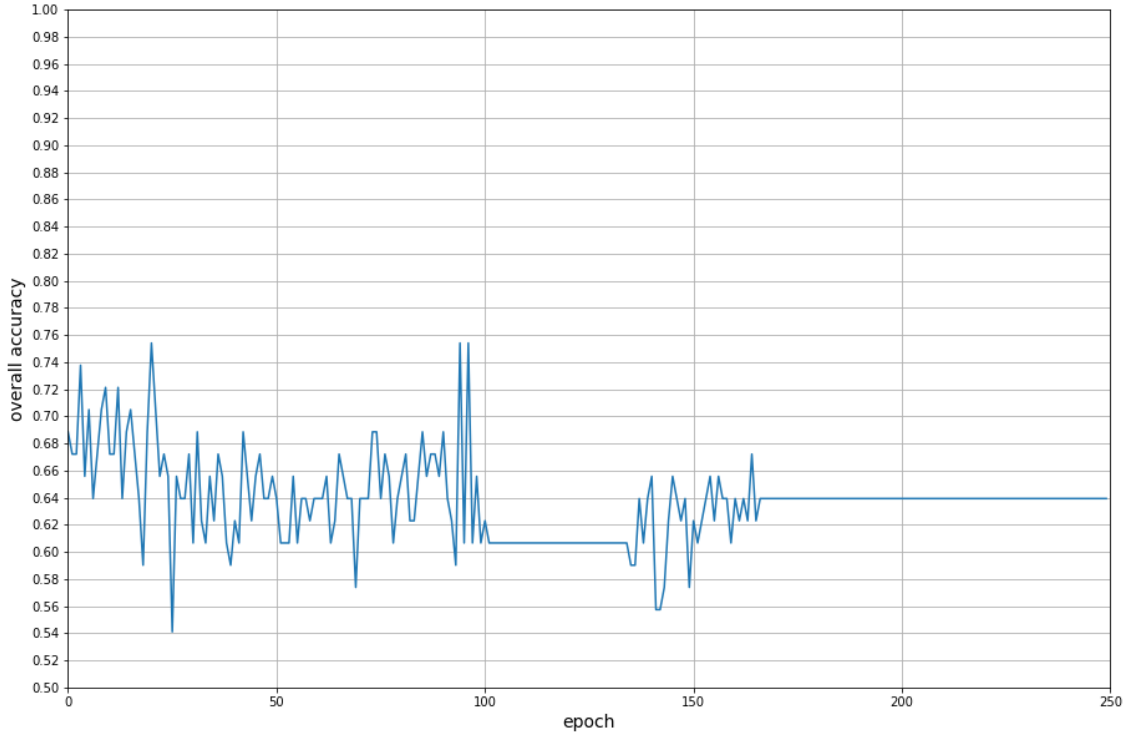

Figure 1. Overall accuracy of validation set.

Task 1.4 Performance comparison

Table 1 summarizes the performance of different algorithms we performed on Pima dataset, with the last row being the BPNN results. Comparing with other models, the overall accuracy of BPNN (i.e., 73.80%) is below the average overall accuracy, which is inferior to Case 1, Case 2, Case 3, KNN, PCA(5) + Case3, PCA(5) + KNN, FLD + Case3 and FLD + KNN. Similarly, the class 0 accuracy only outperforms the KNN, PCA(1)+ Case3, PCA(1)+ KNN, PCA(5)+Case3 and PCA(5) + KNN. Again, the class 1 accuracy is also not prominent. In a summary, the BPNN does not improve the performance while it is inferior to many previous models.

Table 1 Performance comparison.

	Overall acc.	**Acc. of Class 0**	**Acc. of class 1**
Case 1	74.10%	70.64%	75.78%
Case 2	76.81%	72.48%	78.92%
Case 3	74.10%	61.47%	80.27%
KNN (k=12)	78.01%	57.80%	87.89%
wta ($\eta = 0.05$)	71.39%	72.48%	70.85%
PCA(1)+ Case 3	73.19%	57.80%	87.00%
PCA (1) + KNN	70.18%	49.54%	80.27%
PCA(5) + Case 3	77.41%	57.80%	87.00%
PCA (5) + KNN	76.51%	56.88%	86.10%
FLD + Case3	80.42%	54.13%	93.27%
FLD + KNN	78.01%	64.22%	84.75%
BPNN	**73.80%**	**61.47%**	**79.82%**

Task 2.

Run the two example codes using either command line "python example_intersection.py" or simply create a new notebook and run from there. Study the code and get yourself familiar with the simulation setup of the two scenarios.

Task 3.

Modifications to the circular road scenario.

Task 3.1 Human Control

Figure 2 shows the error under human controller with respect to time. The blue dashed line labels the one circle. The vehicle running one circle cost about 129 simulate seconds.

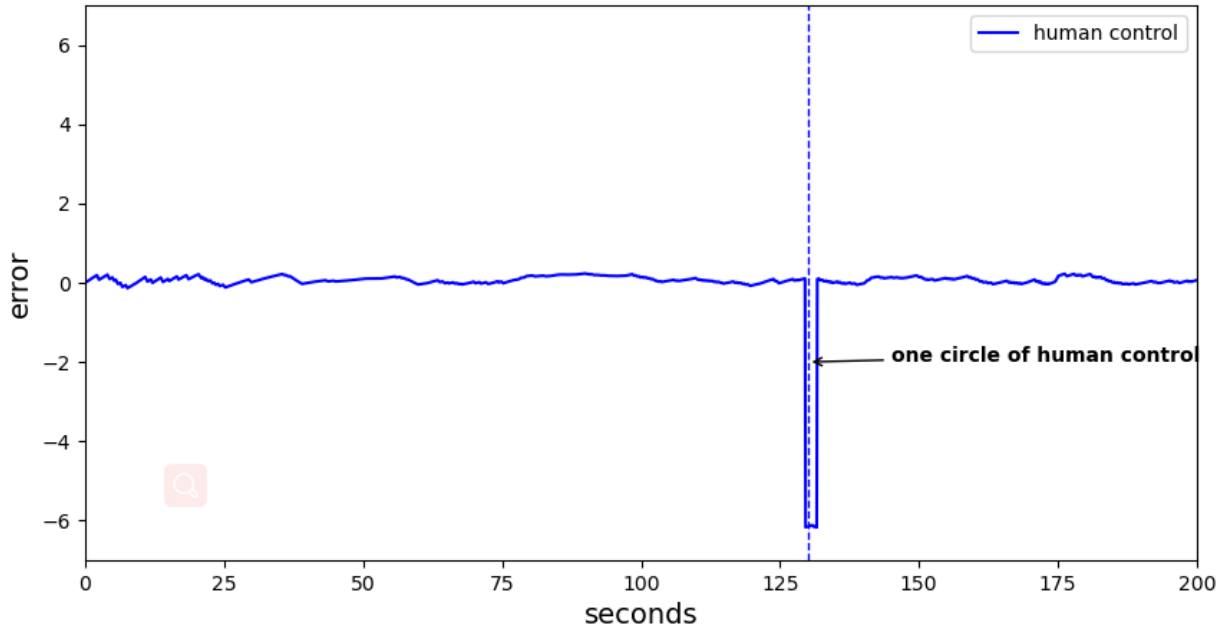

Figure 2 Error of human controller

Task 3.2 Default auto Control

Figure 3 demonstrates the error of default auto controller. The vehicle running one circle elapsed about 42 simulate seconds.

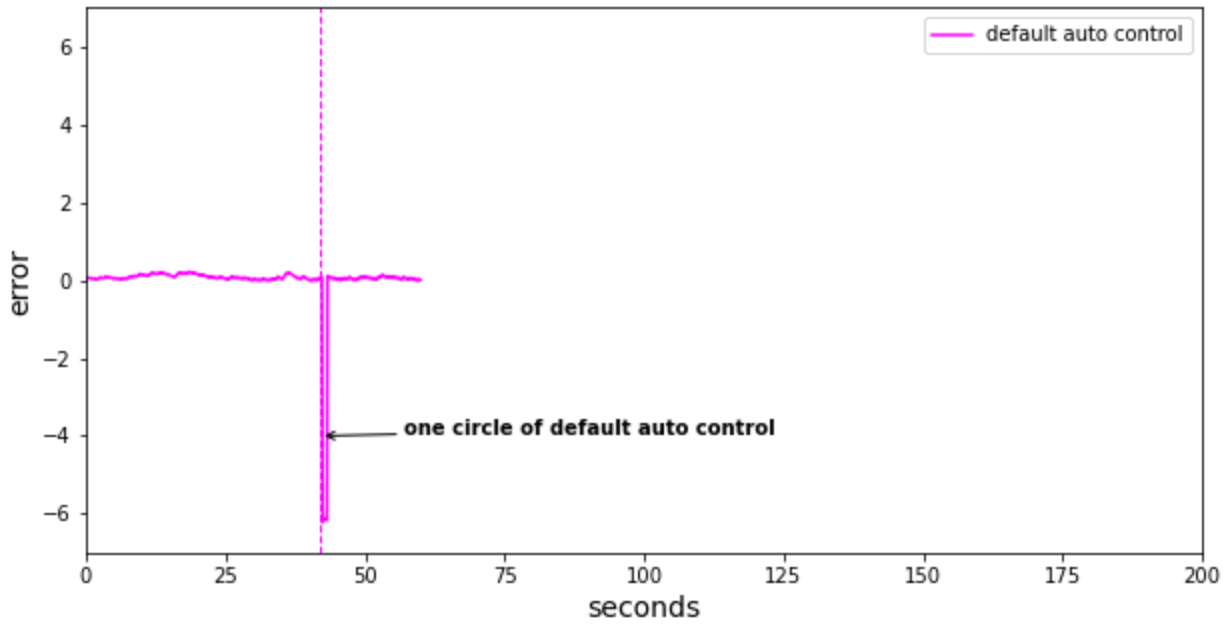

Figure 3 Error of default auto controller

Task 3.3 PID controller

Figure 4 demonstrates the error of PID controller. As is shown in the figure, the vehicle running a circle took about 50 simulate seconds. The error is almost 0 during, and fluctuation is very fine.

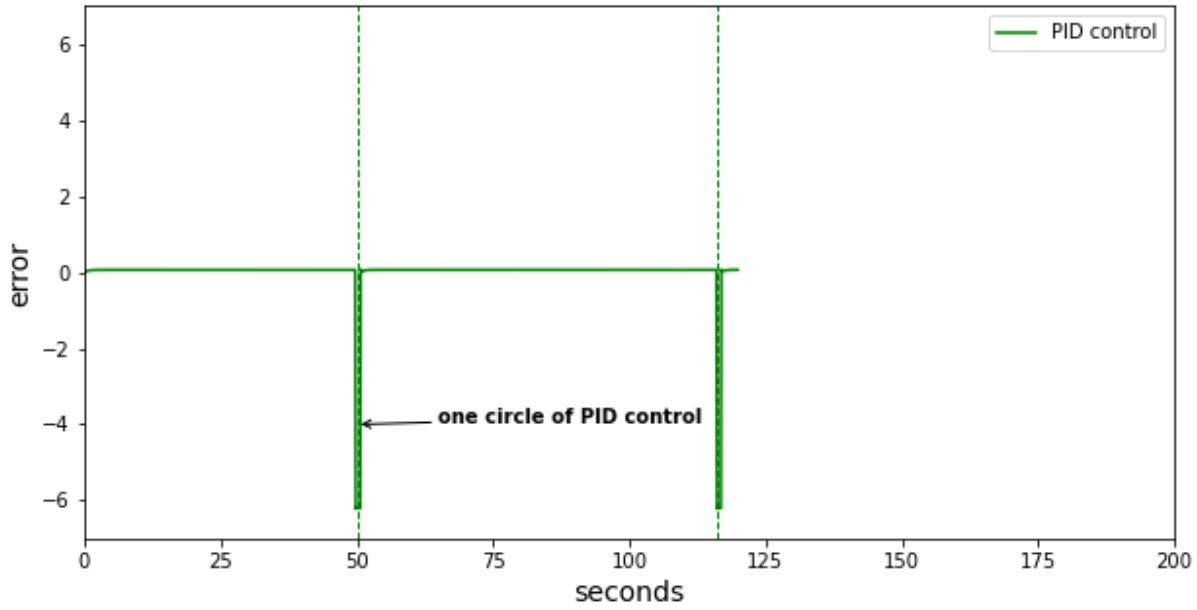

Figure 4. Error of PID controller.

Task 3.4 Reinforcement Learning controller

Figure 5 demonstrates the error change of reinforcement learning controller. Basically, the reinforcement learning trained the policy firstly based on maximum Q value (i.e., reward), and apply the policy to instruct the action. It takes about 61 seconds to finish one circle.

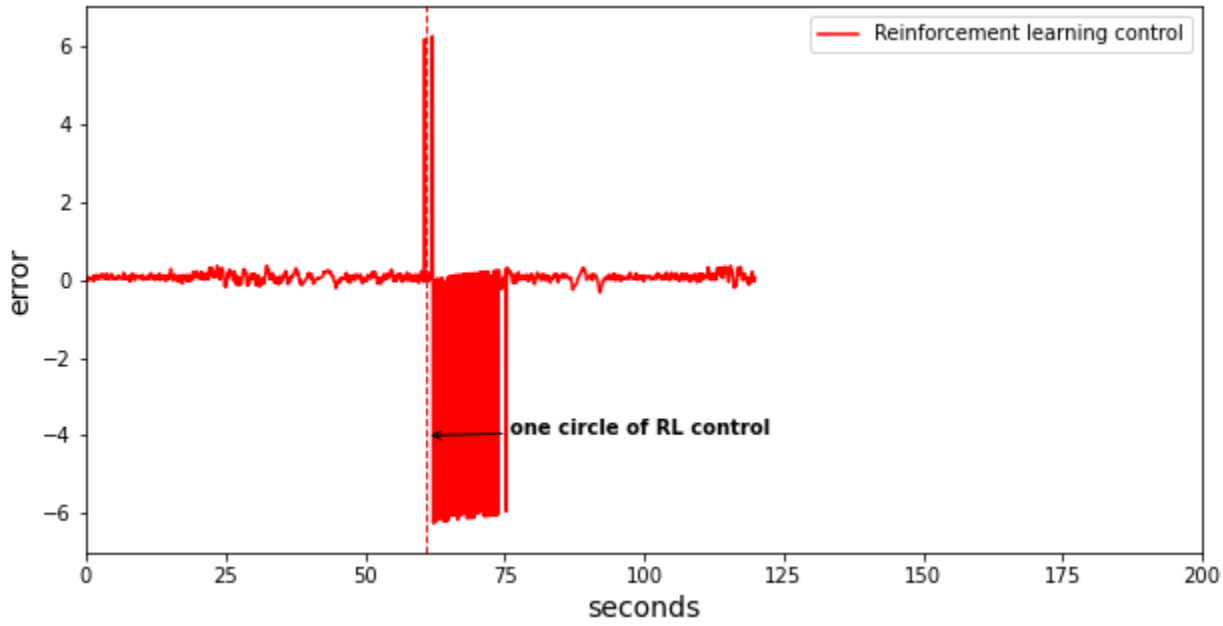

Figure 5. Error of RL controller.

Task 3.5 Comparison of different controller

Figure 6 presents the error change of previous controllers, that is human controller, default auto controller, PID controller, and reinforcement learning controller. The dashed lines indicate time elapsed for the one circle. Looking at figure 6, we have serval findings.

- Different controllers have totally different run time of finishing one circle. For instance, under the default auto controller, the vehicle takes the least time to finished one circle, but it should be noticed that the vehicle didn't stay on the dedicated lane (i.e., lane 1) throughout the first circle. Among smart controllers, with the supervision of PID controller, the vehicle takes the least time to finish one circle, followed by reinforcement learning controller. Not surprisingly, it takes much longer time for a vehicle to finish one circle under human control than other controllers.
- Different controllers produce the different oscillation characteristics of error curve. Due to the overlap of different error curve, we can look at the above single controller's curve. It is not difficult to find that PID generates the most smooth and stable error curve, followed by default auto controller and reinforcement learning based controller.
- It is worth noting that the reinforcement learning cases display the thick fluctuation which lasts 15 seconds (i.e., 61~75 secs) during transition period, indicating that vehicle needs to take much time to return to normal track between the transition of successive

circles. This behavior is totally different from other controllers. Other controllers can quickly find the correct heading at the beginning of next round.

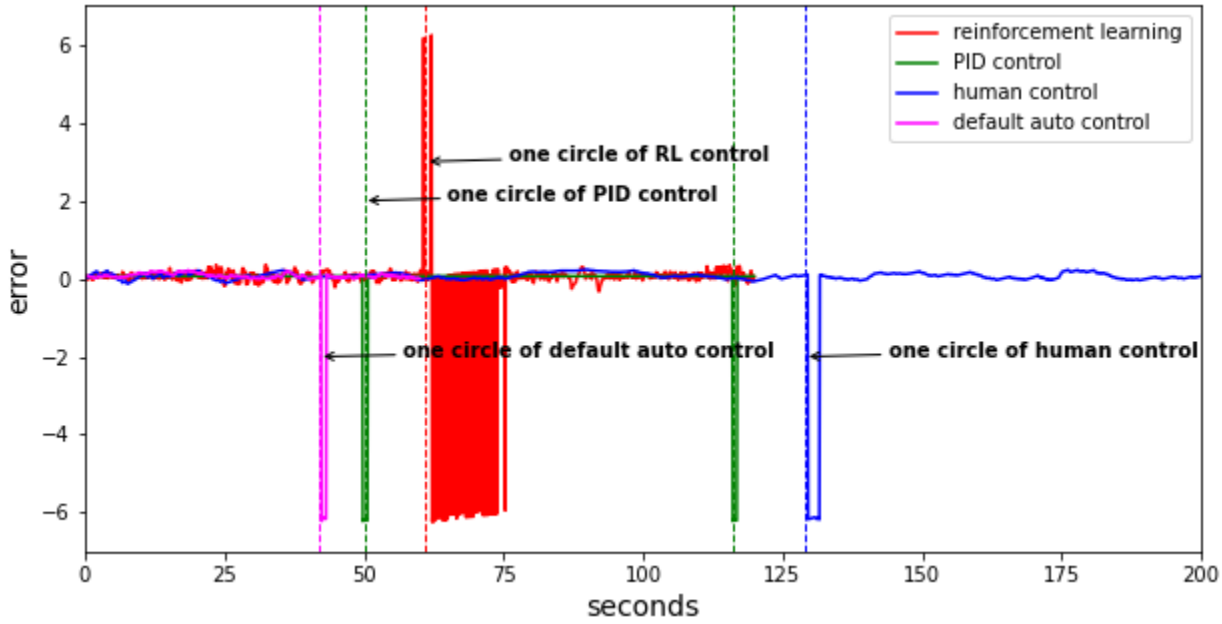

Figure 6. Summarization of all controllers.

Task 4 Bonus 1

Design a 2-car scenario such that not only each car needs to be driving on their dedicated lanes, but they also cannot crash into each other when changing lanes.