INTERNATIONAL JOURNAL OF ADVANCE RESEARCH AND DEVELOPMENT

***(Volume3, Issue4)***

Available online at: www.ijarnd.com

# Training an autonomous car

**Udit Jaitly[1], Abhishek Gupta[2], Shallu Bashambu[3]**

*[1,2,3]Student, Maharaja Agrasen Institute of Technology, Rohini, Delhi*

## ABSTRACT

***This paper aims to simulate an autonomous car which is rewarded according to the actions it takes. The car learns from the actions it does and the rewards it receives for them.***

***The project consists of an Artificial Intelligence system, i.e. an agent acts upon the environment in a virtually simulated environment.***

***A simulator is designed which consists of multiple roads and their intersections. At each intersection, there is a traffic light. Random traffic is also generated on random roads. The agent, i.e. the car, responds to the environment (Traffic lights, Traffic). The agent is rewarded accordingly to whatever decision it makes.***

***Using the results discussed in this paper, any vehicle will be able to cruise through traffic based on real time environment. Furthermore, thanks to the multiple research papers, studies and forum that helped us complete this.***

**Keywords:** *Artificial Intelligence, Reinforcement Learning.*

## 1. INTRODUCTION

The Internet of Things is maybe the most imaginative advancements of our circumstances. The troublesome innovation has conquered any hindrance amongst creative energy and reality. Because of the IoT, brilliant, associated autos, which were before a bit of fiction, are soon going to journey the streets. Genuinely, shrewd autos are the fate of the car business and what's to come is considerably nearer than we might suspect. The savvy auto time will check a fresh start for the car business and it possibly practically around the bend. Self-governing driving will be the standard, with associated autos incorporated with cell phones running on the streets. These autos will tackle numerous advances, for example, GPS for empowering route and directing, counterfeit consciousness for permitting self-learning capacities and information and investigation making an interpretation of and preparing data vigorously. Driving will be more secure, quicker witted and more pleasurable that it has ever been.

Comfort is, maybe, the primary offering purpose of associated vehicles. These vehicles will be controlled by advanced sensors and PCs. Path following and stopping help are different types of accommodation that associated vehicles offer. This will lessen the reliance on human drivers, therefore clearing path for driverless autos. Things will turn out to be substantially simpler for the elderly and the debilitated as well. Furthermore, savvy autos will utilize machine learning innovations to mechanize highlights, for example, temperature control and speed control. They can even be customized to play the travelers' most loved music to make the ride engaging.

Other than comfort, associated autos will offer abnormal state of street security. Keen innovations can control them superior to human drivers. The reason is that the likelihood of blunder and diversion is irrelevant with sensors and PCs. Savvy autos will be empowered with highlights, for example, impact evasion and speed control. They will have sensors to caution the drivers in the event that they distinguish anything out of order, for example, the driver snoozing off while driving. These autos will be eco-accommodating as well, as contamination control will be an essential component for them.

Machine-to-machine correspondence would surface as vehicle-to-vehicle correspondence with associated autos. These vehicles will have the capacity to trade data identified with courses, routes, activity, conditions and speed. This component will be a wellbeing add-on for associated autos as vehicles will approach applicable notices at suitable time. It will be conceivable to turn away accidents with auspicious V2V notices.

## 2. RELATED WORK

Reinforcement learning (RL) is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms.

In the operations research and control literature, the field where reinforcement learning methods are studied is called approximate dynamic programming. The problem has been studied in the theory of optimal control, though most studies are concerned with the existence of optimal solutions and their characterization, and not with the learning or approximation aspects.

In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality. However, our contribution

**(a)** Has an environment that is formulated as a Markov decision process (MDP), as many reinforcement learning algorithms for this context utilize dynamic programming techniques.

**(b)** Reinforcement Learning differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. Instead the focus is on online performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

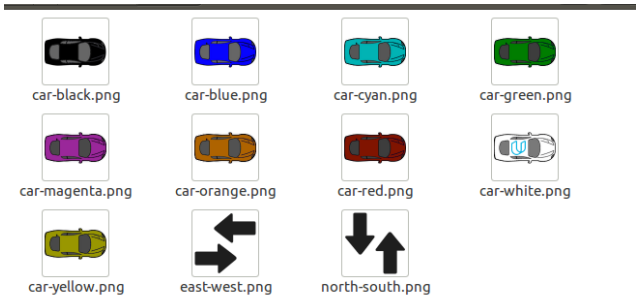
**(c)** Detailed explanation on how the exploration vs. exploitation trade-off in reinforcement learning has been most thoroughly studied through the multi-armed bandit problem and in finite MDPs.

## 3. SOFTWARE USED ANALYSIS

Let's go one by one, analyzing why we need these different software.

Since we will be using NumPy to implement pandas, let's go through the software needed to run a Jupyter Notebook or an iPython notebook.

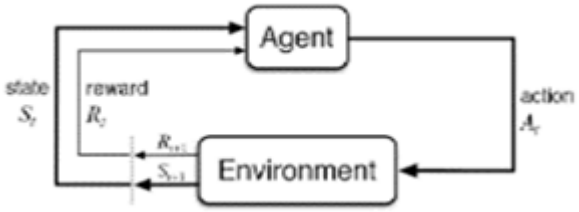
**Anaconda:** For large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment.




**NumPy:** The foundational library for scientific computing. Introduces objects for multidimensional arrays and matrices, as well as routines that allow us to perform advanced mathematical and statistical functions.

**SciPy:** Collection of algorithms and high-level commands for manipulating and visualizing data, includes functions for computing integrals numerically, solving differential equations, optimization, and etc.

**Pandas:** Adds data structures and tools that are utilized for practical data analysis in statistical data. Pandas works well with incomplete, messy, and unlabeled data (i.e., the kind of data one is likely to encounter in the real world), and provides tools for shaping, merging, reshaping, and slicing datasets.




**iPython:** Extends the functionality of Python's interactive interpreter with a souped-up interactive shell that adds introspection, rich media, shell syntax, tab completion, and command history retrieval.

**Matplotlib:** The standard Python library for creating 2D plots and graphs. It's pretty low-level and requires more commands to generate nice-looking graphs and figures. However, the flip side of that is flexibility. With enough commands, one can make just about any kind of graph you want with matplotlib.

**PyGame:** Cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.


## 4. IMPLEMENTATION

The studies shown in this paper have been performed by being carried out in a simulated environment, each one of them referring to a specific usage scenario.

We start to explore this area by initializing a basic driving agent. The first thing we did was to run the simulation to get an understanding of the game. Observations made based on the agent's initial behavior as it took random actions included either the agent being blocked at one intersection for multiple turns because it wanted to turn right at a green light, which was not allowed or the agent went around in loops. Next we assessed whether the vehicle eventually made it to its destination or not. It was also analyzed that the agent can move through the rightmost side of the grid to end up on the left side of the grid. Similarly, it can move through the topmost side of the grid and reappear at the bottom and vice versa.

Next, we moved to the part where information was received by the driving agent. To pass on information and to store it in a format so that the agent can take an informed decision, we decided to implement states as follows and they are appropriate.

| **States** | **Why it's appropriate** |
|---|---|
| Where we want to go next to get to our destination | Give us the optimal next action we should take to reach our destination. |
| Traffic light | Traffic lights will give part of the constraints |
| Oncoming (cars) | Similar to traffic lights, oncoming cars (straight ahead) are a constraint on our actions. |
| What the car immediately to the left wants to do | If the car to the left is going to turn right, you don't want to turn left and crash into it. |
| What the cars immediately to the right wants to do | If the car to the right is going to turn left, you don't want to turn right and crash into it. |

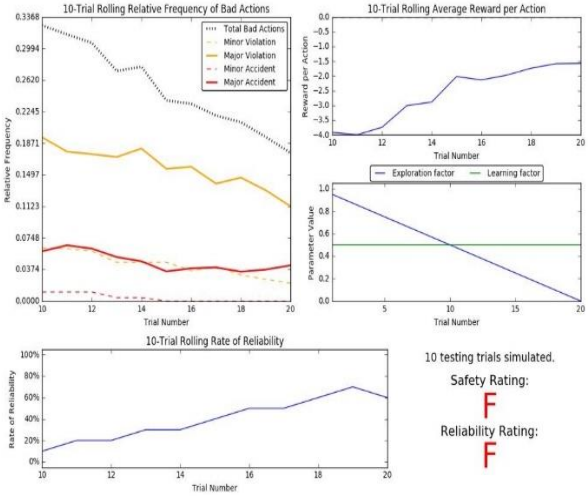
Our analysis now turns to understanding the total states that exit for the vehicle within the environment at any given instance. We check whether the number of states appear reasonable considering that our algorithm's aim is to learn and make informed decisions. If they seem low, we add mode states to include all possible attribute value for the states of the vehicle.

Moving ahead, we finally implement the learning algorithm and add it to the driving agents operations. The agent chooses the best action to take by choosing the action with the maximum state value for the corresponding state. It chooses a random action if there are multiple actions where the resulting state value is the maximum possible value. The agent also chooses a random action with a defined probability. This allows it to escape if it 'gets stuck' in some suboptimal local optima. The initial learning rate is high so the agent learns quickly. As time goes on, the agent becomes more confident with what it's learned and is less persuaded by new information.

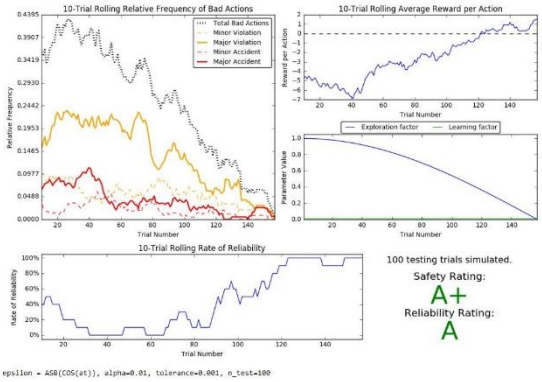
 

The agent is more likely to take actions corresponding to the successive waypoint when those are viable because it receives a higher reward if it can execute that action without penalty and is learning to behave in ways that maximizes total expected reward. The agent is less likely to take actions that result in penalties (crashing into cars, making illegal moves or making moves that are legal but are not equal to next waypoint so they are further from the destination) because those usually do not maximizes reward. It does learn to make legal but 'incorrect' moves rather than crash into a car. The agent reaches the destination more frequently: after implementing learning algorithm, it reaches the destination in time over 80% of the time, whereas while it was moving randomly it reached the destination in time less than 10% of the time. The agent does not just move randomly in loops any more.

As the agent gains experience, it is less likely to go around in loops or get penalized for going against traffic rules or crashing into other cars.




We improve the learning within the driving agent further by running successive trails over each configuration. The metrics thus obtained is maintained in a separate file. The statistical summarization, which follows, helps us alter the list of successive configurations and repeat trials until satisfactory results are obtained. The metrics considered were the total number of successful outcomes because unsuccessful outcomes (Trail aborted as car did not make it in time) indicates inefficiency, the average buffer (Time left / Initial deadline) which indicates how efficient the driving agent was and the average number of incorrect actions per trial (penalties of negative reward) because this indicates that an action was unsafe.




## 5. CONCLUSION

Using sensor feedback and machine learning in a vehicle, it can be given a supplementary skill to identify the environment and take actions accordingly.

An optimal policy would be successful in almost all trials (the exceptions being where it is impossible for some reason). That is, it would attain close to 100 successes per 100 trials. It would be efficient and thus approach the theoretical maximum buffer of 0.8 (tentative). It would maximize net reward and thus likely incur close to zero negative penalties.

Comparing our driving agent to the optimal policy, it had a 99.12 average success per 100 trials, a 0.5926 average buffer (proportion) per trial and a 0.164 negative penalties.

Judging by the Average Successes per 100 trials, our policy is close to the optimal policy. As noted before, it's hard to know what the optimal policy's average buffer would be so although there is a gap, we don't know how great the gap between our policy and the optimal one is. There are still a significant number of penalties occurring (violations of traffic rules or crashing). This is suboptimal.

The penalties occur because the agent has had little (99) or no (94) previous experience in this state. These are usually states where oncoming, right, or left are not blank.

We then conclude that our policy is efficient but not nearly as safe as it could be.

This paper aims to make people understand as to how such a system can add to the learning abilities of an autonomous vehicle.

## 6. REFERENCES

[1] Wikipedia, "Reinforcement Learning", Website, URL: https://en.wikipedia.org/wiki/Reinforcement_learning

[2] Sainbayar Sukhbaatar, Arthur Szlam, Gabriel Synnaeve, Soumith Chintala, Rob Fergus, "MazeBase: A Sandbox for Learning from Games (2015) "Research Paper, Cornell University Library, URL: https://arxiv.org/abs/1511.07401

[3] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, Sergey Levine, and "Continuous Deep Q-Learning with Model-based Acceleration" (2016), Research Paper, Cornell University Library, and URL: https://arxiv.org/abs/1603.00748, 2016

[4]Tutorials Point, "Artificial Intelligence - Overview", Website, URL:

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_overview.htm, 2017.

[5] YouTube, "Pygame (Python Game Development) Playlist", Website, URL: https://www.youtube.com/watch?v=K5F-aGDIYaM&list=PL6gx4Cwl9DGAjkwJocj7vlc_mFU-4wXJq, 2015.

[6] Scienceblogs, "Reinforcement and punishment", Website, URL: http://scienceblogs.com/cognitivedaily/2007/08/06/basic-concepts-reinforcement-a/, 2016