Reinforcement Learning based Driver Warning System

Arjun Karuvally
University of Massachusetts, Amherst
akaruvally@cs.umass.edu

Abstract

Car warning systems has been exhaustively researched with regards to various aspects to get the optimal time to warn the driver. There has been less work focusing on a reinforcement learning based agent that is applied to such a problem. The ideal time to warn depend on various factors that interplay. The idea to train an RL agent to make decision on the perfect time to warn based on the various features that it observes poses an interesting problem. This paper delves into this problem and seeks to improve car alarm systems in reallife.

1. Introduction

With the advent of autonomous cars, the need to notify drivers of dangers is increasing. The perfect time to notify driver requires information about a myriad of different features of drivers, car and environment. Sometimes early warning systems have negative influence on drivers [3]. Creation of a heuristic based system for warning drivers has its own demerits with the question of reliability of such a system since it affects users directly. Yet this project focuses to show how good the current system is compared to naive approaches to car warning systems and aims to provide a basis on how reinforcement learning agents can be used to make these decisions in real life. This project discusses a simulated environment where the agent works to improve the car warning system. The environment is modelled to best resemble a real world based on data from a freely available dataset FARS(Fatality Analysis Reporting System).

2. Technical Approach

Initially the FARS dataset is taken to create a Bayes net to make decisions on features to output a probability of crash. This is taken as environment. The rewards are modelled on whether the driver crashed or not. A step in and this environment setting is an event of crash. An episode represent one driver. The same experiment is done for multiple drivers. A good estimate of the average metrics are obtained by running multiple trials of the agent on the system. During test time, the exploration parameter(epsilon) is set to zero as it is a safety problem and only the best action is to be taken. Experiments are run for SARSA and Q-Learning agents.

3. Environment

The environment is modelled using FARS(Fatality Analysis Reporting System) dataset. FARS is a freely available dataset of fatal crashes. It has various factors at the time of crashes reported using real life crash scenarios. We are concerned with only some of this data. So only a subset of features is taken. The features we require are:

- age - 3 categories:
  - 18 to 25 years
  - 25 to 65 years
  - above 65 years
- gender - male/female
- state of driver - 4 categories:
  - Intoxicated
  - Tired
  - Distracted
  - Drug Impairment
- number of passengers in car
- equal to or below 5
- above 5
- number of previous crashes
- ttc(time to collision)
- trust

The statistics about these features are taken to model the probability of crash. To model the probability of crash we use the Bayes theorem.

$$P(X/Y) = \frac{P(Y/X) \times P(X)}{P(Y)}$$

Here, Y - features from the dataset(X)

$$P(crash|X) = \frac{P(X|crash)P(crash)}{P(X)}$$

Here $P(X|crash)$ can be infered from the statistics of the dataset. $P(crash)$ can be assumed safely. But the issue is the inference of P(X). In real life the probability of features cannot be infered easily nor sampled easily.

$$P(X) = P(X|crash)P(crash) + P(X|\neg crash)P(\neg crash)$$

One crucial element missing to generate the probability of crash is the probability value of $P(X|\neg crash)$ which presently there are not available datasets. It is difficult to create such a dataset. To get over this issue we take the expected value of $P(crash|X)$ assuming that the value $P(X|\neg crash)$ can take value from 0 to 1 uniformly. This assumption simplifies the calculation of the expected value but can be replaced by the original known value once a dataset is available that has this probability. Using these assumptions.

Let,

$$P(crash) = \alpha$$
$$P(X|crash) = \beta$$
$$P(X|\neg crash) = y$$

Derivation,

$$P(crash|X) = \frac{\alpha\beta y}{\alpha\beta + (1-\alpha)y}$$
$$E[P(crash|X)] = \int_0^1 \frac{\alpha\beta y}{\alpha\beta + (1-\alpha)y} dy$$

On Integration we get

$$E[P(crash|X)] = \frac{\alpha\beta}{(1-\alpha)^2}[(1-\alpha) - \alpha\beta[ln(\frac{\alpha\beta + 1 - \alpha}{\alpha\beta})]]$$

All the experiments are conducted using these assumptions and this expected value of $P(crash|X)$. It can be noted that some of the features are categorical and some are scalar. To accomodate this difference all features are scaled from 0 to 1. With different categories occuring at equal intervals. Experiments using fourier basis expansion was performed but the results were not very satisfactory. Another interesting feature that is taken into account is trust that decreases when the sounded alarm is not necessary, and increases if alarm is reliable. The minimum trust is taken as 0.5 and maximum trust is taken as 0.8. Using the naive system we can expect only 50% trust from the user[2]. Trust determines if the user listens to our alarm or not. The decay factor of trust is taken as 0.15 and the incrementation factor taken as 0.1 [1]. Studies have shown that trust decreases fast and increases slowly. At each time step the RL agent makes decision whether to warn or not.

4. Rewards

- For crash without warning a high negative reward (-200) was given.
- For crash even if user was warned a small positive reward(+10) was given.
- For successful warning a high positive reward(+100) was given.
- For false warning(even if there was very less possibility of crash) a high negative reward(-200) was given. False alarms decrease the trust of the driver and eventually leads to accidents in the future.

5. Experimental Setup

5.1. Train

One step in an episode is taken as a single event of a particular driver. At the start of an episode, the attributes of a driver is sampled uniformly randomly from the available features. Trust is initialized at 100%. The step begins with the choosing of an initial ttc(Time to collision). TTC is chosen from a gaussian distribution with mean 5 and standard deviation of 1. The stopping time is taken as 1 sec. This is the time required to stop vehicle after the alarm has been sound. TTC is constrained to the range of [stopping time +1, 8]. This resembles real scenario where the alarm system of car is activated. Initial probability of crash is sampled from the statistics generated from data using the derived formula. Each time step increments this probability of crash. So if ttc is less probability of crash is higher. 1000 such crash events are considered one episode of a driver. Each episode represents different drivers(This method is chosen because the trust parameter varies within an episode and episodes must be independent of one another). Experiment is conducted on 5000 episodes.

5.2. Test

During test time the exploration parameter is set to zero, since it is a safety problem and we dont want the agent to

explore from what it has learned so far. The horizon of an episode is set at 500. The rest of the parameters are similar to train time.

5.3. Variations

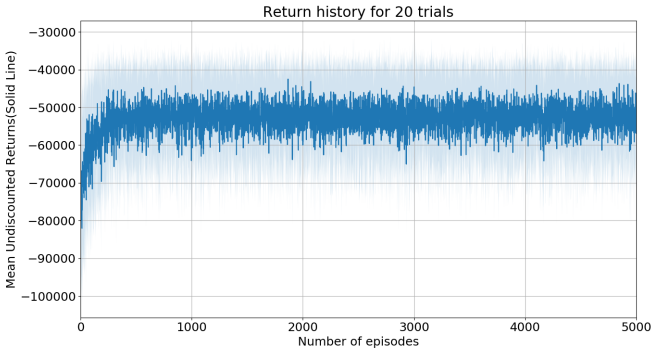
Two experiments were conducted. 20 trials of each experiment was conducted.

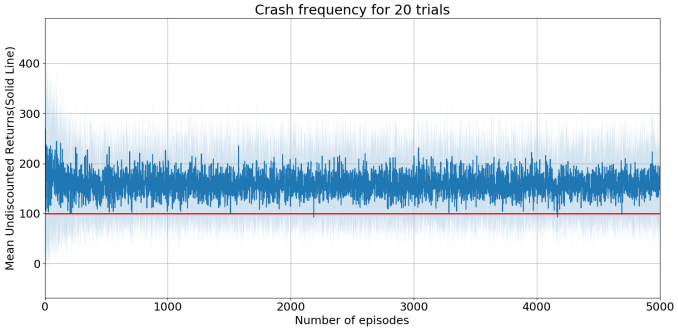
- SARSA and Q-Learning[4] using 0.4 as the prior probability of crash. This value is high compared to what is expected in real life.
- SARSA and Q-Learning using 10^{-5} as the prior probability of crash. This somewhat resembles real life scenarios.

6. Results

6.1. High prior probability

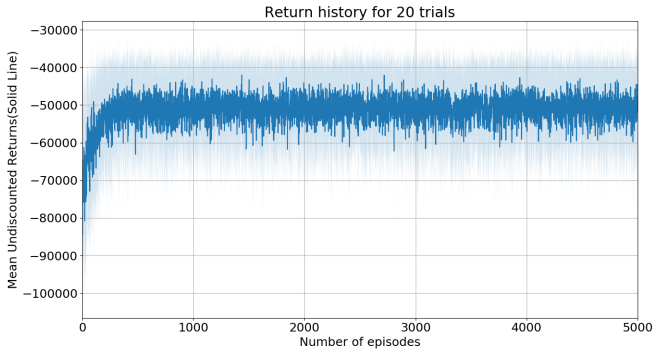
- SARSA based agent on an environment with high prior probability of crash. P(crash)=0.4
  Hyperparameters:
  - alpha: 0.02
  - gamma: 1
  - epsilon: 0.25

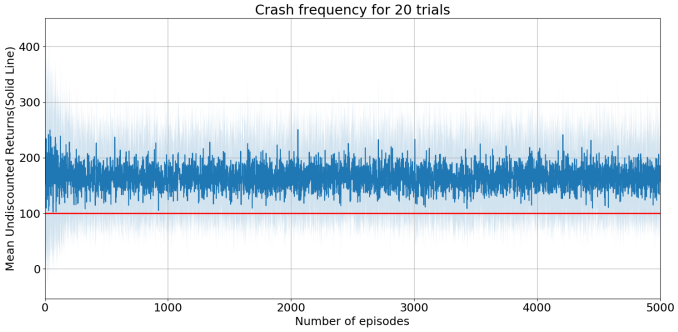




Red Line in crash frequency graph shows the best achievable crash frequency(when trust is 0.8 always). This crash frequency can be achieved only if the agent can perfectly predict the future. It can also be noted that the start value of the crash frequency graph is truncated. This is because the first value is made only after the first episode, which has 1000 steps. By that time the agent has learned to decrease the crash frequency from 500 to 200. The rest can be observed in the graph.

- Q-Learning based agent on an environment with high prior probability of crash. P(crash)=0.4





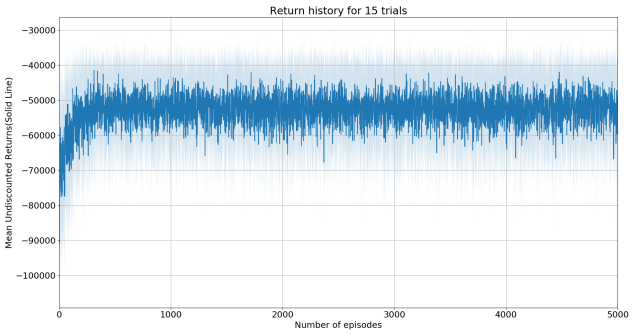
6.2. Low Prior Probability

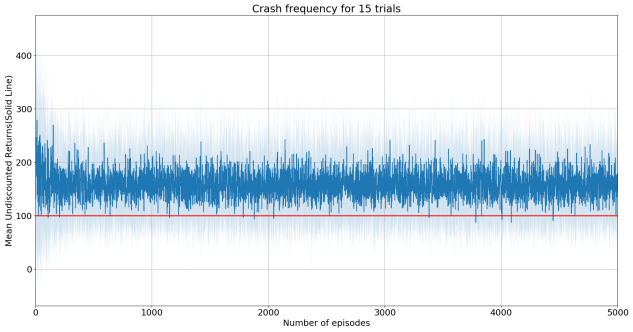
- SARSA based agent on an environment with low prior probability of crash. P(crash)=$1 \times e^{-5}$
- Q-Learning based agent on an environment with low prior probability of crash. P(crash)=$1 \times e^{-5}$

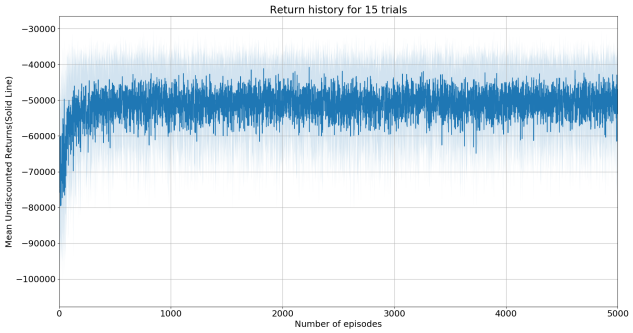
7. Conclusion

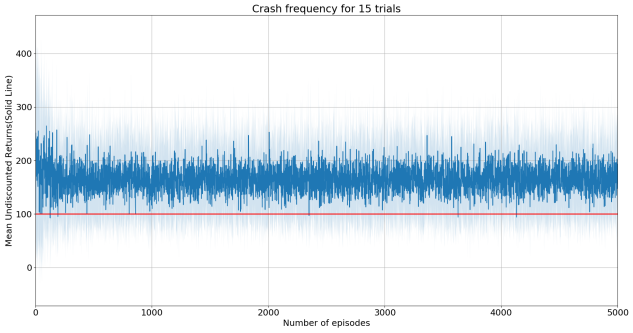
From the conducted experiments, it can be concluded that:

- The reinforcement learning agent was able to capture the right time at which to warn the driver and effectively reducing the number of crashes for a driver over









time. The final number of crashes it was able to obtain is near to optimal value(the red line) of 100 crashes.

- The prior of the number of crashes does not change the learning process much. This could be due to the fact that the dataset has reliable info that overshadows the initial prior crash probability.

- Both SARSA and Q-Learning both exhibited similar learning capabilities.

  It can be concluded that with proper information about the probability of not crashing derived using real world experimentation, reinforcement learning could be applied to the task of driver warning system.

References

[1] G. Abe and J. Richardson. The effect of alarm timing on driver behaviour: an investigation of differences in driver trust and response to alarms according to alarm timing. *Transportation Research Part F: Psychology and Behaviour*, 7:307 – 322, 2004.

[2] J. P. Bliss and S. A. Acton. Alarm mistrust in automobiles: how collision alarm reliability affects driving. *Applied Ergonomics*, 34:499 – 509, 2003.

[3] M. Maltz and D. Shinar. Imperfect in-vehicle collision avoidance warning systems can aid drivers. *Human Factors*, 46(2):357–366, 2004. PMID: 15359683.

[4] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.