# Vehicle Path-Finding using Q-Learning

**James Ni** and **Ashley Alexander-Lee**
{jani,asalexanderlee}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

**Abstract**

Self-driven cars are hot topics in the world of modern artificial intelligence. We explore a similar problem in a simplified setting using a model provided by Pratt and Miller. This model simulates a bicycle-like vehicle on a race track, which we interfaced with our driving intelligence by providing an input of a torque to apply on the front wheel, a torque to apply on the rear wheel, and a steering angle. We attempted to implement the Q-Learning algorithm for estimating expected returns such that the car is able to learn to navigate itself around an oval track. With our implementation, we were able to successfully traverse a fourth of the track; however, progressing further proved to be difficult due to both errors in calculations and unexpected properties of the track.

## 1 Introduction

In this problem, we sought to find a policy that would guide a car simulation around a given track in a model-free way. Ultimately, we decided to use Q-learning to calculate the discounted sum of future rewards, as it meets the stipulation listed above and it allows us to access our optimal policy.

There are a few methods we could have considered in the place of Q-learning. One such method is Bellman's equation, a passive learning model, which calculates the discounted sum of future rewards by estimating the reward and probability matrix using a sample of simulations. However, while this approach is offline, it is also model-based, which is inefficient. We also had the option to use temporal difference learning; however, even though it is a model-free learning algorithm, we are unable to extract the policy or improve it. Instead, we can only evaluate the merit of the current policy. Q-learning allowed us to approach learning in an "active" manner, which permitted us to rise above the limitations of the other methods.

When approaching this Q-learning problem, we were provided with information about the track state and vehicle state such as the vertical outer and inner radii of the track, the horizontal outer and inner radii, the current displacement of the vehicle, and boolean value indicating whether the vehicle has hit the wall. In addition, we had information about the actions we could take at each state, which include front torque, back torque, and steering angle.

The remainder of this paper will be sectioned into five different parts: background, experiments, results, conclusions, and future work.

## 2 Background

### Q-Learning

Q-learning is typically used to find the discounted sum of future rewards in a model-free, policy-retaining way. The idea behind it is that we take an immediate action and act optimally thereafter. The policy is constantly improved through a trial and error process, during which certain actions and states are rewarded and weighted appropriately for the following simulation run.

### Gradient Descent

Gradient descent is the process by which we update each of the weights by observing the distribution of error and iteratively altering the weight such that it approaches the local minimum. We define the error on a point as follows:

$$J = \frac{1}{2}(y - wx)^2$$

where $y$ represents the current Q value, $x$ represents the current feature, and $w$ represents the weight. Since the derivative of the error on the point is the local minimum, if we want to update the weight iteratively, we subtract the learning rate times the local minimum from the weight, as follows:

$$w = w - \alpha(wx - y)x$$

For all of the weights, we can write this equation in terms of Q learning:

$$w_k = w_k + \alpha[R(S_i) + \delta max_{a1} Q(s', a') - Q(s, a)]f_k(s, a)$$

Therefore, we add the current weight to the learning rate multiplied by the current feature and the combination of the reward, the best Q value, and the negative current Q value. This equation will allow the weights to converge on values to reflect the rewards of the system.

### Relating Translational and Rotational Kinematic Equations for Vehicle Motion Prediction

In our experiment, we utilized the kinematic equations for both translational and rotational motion. This allowed us to give the system a predicted next position and velocity as a function of a torque applied onto the wheels of the car. From the kinematic equations, we know that for a constant acceleration,

$$x(t) = x_0 + \dot{x}_0 t + \frac{\ddot{x}t^2}{2}$$

Note that dot notation refers to the time derivative of a variable. Thus, $\ddot{x}$ refers to simple translational acceleration. The rotational analogue of Newton's Second Law of Motion is

$$\tau = I\ddot{\theta}$$

where $\tau$ is the torque applied, $I$ is the moment of inertia, and $\ddot{\theta}$ is angular acceleration. We also know that angular acceleration $\ddot{\theta}$ is related to translational acceleration $\ddot{x}$ by

$$\ddot{\theta} = \frac{a_{Tangential}}{R} = \frac{\ddot{x}}{R}$$

$$\tau = I\frac{\ddot{x}}{R}$$

$$\ddot{x} = \frac{\tau R}{I}$$

where $R$ is the radius of the rotating body, or the wheel in terms of our vehicle.

We thus find the equation

$$x(t, \tau) = x_0 + \dot{x}_0 t + \frac{R}{2I}\tau t^2 \quad (1)$$

The time derivative of equation 1 gives a predicted velocity. Thus we find

$$\dot{x}(t, \tau) = \dot{x}_0 + \frac{R}{I}\tau t \quad (2)$$

Finally, noting that $R/I$ is a constant, this equation for simple translational motion without friction can give a rough estimate of motion for the wheels of a vehicle.

## 3 Experiments

### System Environment

The elliptical track provided had a semi-major axis radius of 500 units for the inner radius, and 550 for the outer radius. The semi-minor axis for the ellipse inner and outer radius was 300 and 350, respectively. The vehicle begins along the track with a velocity of $V_x = 10$ and $V_y = 0$. Each time step seems to advance the system time by .1. We found these parameters by applying no torque and no change in the steering direction to find that the system does not have friction along the straight line path. By using these measured parameters and then applying small torques, we can then apply equation 2 and isolate the constant $R/I$, which we thus calculated to be $1.10237 \cdot 10^{-3}$. This seems like a reasonable value, given that a vehicle should have a large moment of inertia due to its mass and relatively small wheels.

Table 1: The vehicle does not accelerate or decelerate when travelling in a straight line, Total Torque = 0

| $x$ | $y$ | $V_x$ | $V_y$ | $\theta_{steer}$ |
|---|---|---|---|---|
| 0.0 | 325.0 | 10.0 | 0.0 | 0 |
| 1.0 | 325.0 | 10.0 | 0.0 | 0 |
| 2.0 | 325.0 | 10.0 | 0.0 | 0 |
| 3.0 | 325.0 | 10.0 | 0.0 | 0 |
| 4.0 | 325.0 | 10.0 | 0.0 | 0 |
| 5.0 | 325.0 | 10.0 | 0.0 | 0 |
| 6.0 | 325.0 | 10.0 | 0.0 | 0 |
| 7.0 | 325.0 | 10.0 | 0.0 | 0 |
| 8.0 | 325.0 | 10.0 | 0.0 | 0 |
| 9.0 | 325.0 | 10.0 | 0.0 | 0 |
| 10.0 | 325.0 | 10.0 | 0.0 | 0 |

Table 2: The $x$-axis velocity $V_x$ after 10 steps shows the constant $Radius/Inertia$, Total Torque = 1

| $x$ | $y$ | $V_x$ | $V_y$ | $\theta_{steer}$ |
|---|---|---|---|---|
| 0.0 | 325.0 | 10.0 | 0.0 | 0 |
| 1.00000478295 | 325.0 | 10.0000971435 | 0.0 | 0 |
| 2.00002073415 | 325.0 | 10.0002101756 | 0.0 | 0 |
| 3.00004788561 | 325.0 | 10.0003217009 | 0.0 | 0 |
| 4.00008618959 | 325.0 | 10.0004332261 | 0.0 | 0 |
| 5.00013564607 | 325.0 | 10.0005447506 | 0.0 | 0 |
| 6.00019625494 | 325.0 | 10.0006562745 | 0.0 | 0 |
| 7.0002680162 | 325.0 | 10.0007677983 | 0.0 | 0 |
| 8.00035092985 | 325.0 | 10.0008793222 | 0.0 | 0 |
| 9.00044499588 | 325.0 | 10.000990846 | 0.0 | 0 |
| 10.0005502143 | 325.0 | 10.0011023699 | 0.0 | 0 |

The data in table1 reflects the prediction for no friction, and the data in table 2 shows same sample data for calculating the constant $Radius/Inertia$ of the vehicle. A quick glance gives the constant, as after 10 time steps, the change in the velocity relative to the zero torque applied case is $R/I$ as the applied total torque is 1 in this test data.

### Exploration

We wanted to encourage exploration so that we did not inadvertently narrow our future options by choosing the best option in each episode. Therefore, for every tick of the system's clock, or, in other words, for every episode, we calculated the value of $\epsilon$ which is our likelihood of choosing the best policy over a random policy. If we chose a random probability and it was smaller than $\epsilon$, then we would choose a random policy. As our policy improved, exploration would become less and less necessary, meaning $\epsilon$ needed to decay over time. Therefore, our equation for $\epsilon$ was

$$\epsilon = \frac{1}{episode}$$

where one episode a complete run of the vehicle until it reaches it's goal or it crashes. For our random policy, we

chose to generate random, discrete values for the torque and the steering angle in the hopes that it would generate enough diversity that the agent could learn properly.

## Reward

For the agent to desire to progress along the track, we generated a reward function that could be evaluated at each state of the system. The number of states of the system is not finite, so it was much more appropriate to use a function that evaluated the desirability of each individual state. Our primary aim was for the system to successfully travel along the ellipse track, thus our reward function consisted of a positive value for distance travelled radially, a reward for each velocity component in the appropriate direction, which also subsequently punishes the system if the signs of the velocity are not appropriate for its appropriate position, and a punishment for being off the center of the track. An inverse of the speed squared was also subtracted from the total reward to discourage unreasonably slow speeds. Thus, the reward function we developed was

$$R(s) = \vec{r} \cdot \Delta\vec{\theta(s)} + V_y(s) + V_x(s) - \Delta\vec{r_{walls}} - \frac{C}{|\vec{v(s)}|^2} \quad (3)$$

We set $C = .1$ as it seemed that the simulation stopped when the speed was less than .03 from experimental tests. An error code reported that the stopping condition was $|v| < .001$, so we used a higher value such that we could punish low speeds at an earlier point. In the experimental runs conducted, multiplicative constants were also attached to the other components of the equation to create different proportions of rewards while testing.

Note that in the case of the vehicle crashing into the wall by going into or out of the bounds of the track, the reward function would return a large negative constant penalty for the state. In our experiment, we utilized values of $-250$, $-2500$, and $-25000$ in our experiment. The value used on a few of our best results was $-25000$. We also punished the system coming to a complete halt, and thus leading to the simulation stopping, with this same negative reward value.

## Features

In order to perform Q-Learning, it is necessary to create an equation that can give some type of understanding or approximation of the system in which the agent resides and how an action affects the system. We utilized a linear approximation strategy by defining a set of features, normalized to return a value between 0.0 and 1.0 to describe some aspect of the current state and some particular action taken.

These features are multiplied by a corresponding weight value that allows the algorithm to learn which feature to emphasize. Each feature is normalized differently, such that some will return values closer to 1.0 when the system is most benefited while others will return closer to 0.0. The weights associated will allow the algorithm to decide which features lead to better outcomes and which lead to worse outcomes. Noting that our set of actions consists of a steering angle and applied torque on both front and rear wheels, our features are as follows:

- *Constant feature*: We use a constant feature which always evaluates to 1.0 in order to provide a balancing constant to the total value of the linear equation.
- *Steering angle*: Our function evaluates the steering angle planned for the vehicle to steer relative to the angle expected for it to continue moving in the same direction as the curvature of the ellipse.
- *Front wheel torque*: Evaluates the applied front wheel torque relative to some maximum torque.
- *Rear wheel torque*: Evaluates the applied rear wheel torque relative to some maximum torque. As we apply the same torque to both wheels, the associated weight of the two torques should converge to a value close or exactly the same value.
- $V_x$: Evaluates the difference of the expected x-axis velocity component relative to the expected value at the position of the ellipse due to the curvature.
- $V_y$: Evaluates the $y-axis$ analogue to the previous feature.
- *Velocity tangent to ellipse*: Evaluates the velocity towards the walls of the track. A tangential velocity of 50 would suggest that the system will unavoidably crash into the wall at the next time step.
- *Distance travelled*: Evaluates the distance travelled at the current point relative to the total distance of the track.
- *Displacement from the center*: Evaluates the difference between the distance from the inner and outer wall, such that a minimum value of 0 exists along with a maximum value of 50.
- *Predicted* $V_x$: A predicted analogue of the $V_x$ feature, for a predicted next position. Predictions were made using kinematic equations.
- *Predicted* $V_y$: A predicted analogue of the $V_y$ feature.
- *Predicted distance travelled*: A predicted analogue of distance travelled.
- $\Delta\theta_{Steering}$: Evaluates the change in the steering angle relative to some maximum possible change in the steering angle. This comes from the idea that generally large sudden changes in steering leads to drifting or other undesirable behavior in real cars.
- *Low speed*: Evaluates the inverse of the speed with a multiplicative constant proportional to the stopping condition of the simulation. This feature is an attempt to encourage the vehicle to continue moving forward. A speed too low would lead to the simulation ending and is unideal in the case of an actual racing vehicle.

## Procedures and Parameters

We initialized the weight values of each function to a random value within the range of $[-1.0, 1.0]$. We expected that the values would converge to values appropriate to

accurately model each specific feature of the system through the use of gradient descent after running enough trials. We chose to run a full simulation using a single set of weight values and then updating the values after reaching a crash or traversing the track fully. In the main run discussed in results, which travelled about $\frac{\pi}{2}$ radially around the track, we assigned the learning rate parameter and the discount factor to .2 and .5, respectively.

We tested a finite number of actions by testing discrete values of torque applied per time step and change in the steering angle. For main trial producing figure 1, we utilized torque applied on the front wheel and rear wheels between $[-5.0, 5.0]$ in steps of 1.0. The same torque is applied on both wheels. The range of the change in angle per simulation step was $[-\frac{\pi}{4}, \frac{\pi}{4}]$ in steps of $\frac{\pi}{32}$. We utilized a low torque as our primary concern at the time was for the vehicle to travel along the path as far as possible, with the assumption that we simply needed our car to continually move forward. The small angle increment but large range allows the vehicle to make small adjustments such that it can steer itself along the center line as best as possible. The reward function was slightly modified, further multiplying the constant of the inverse speed term by 20, so the term is effectively $\frac{2}{|\vec{v}|^2}$. We also multiplied the deviation from the center line term by 2 in this trial.

## 4 Results

We performed many different experiments with a variety of parameters, and some parameters performed better than others. To reiterate, along with the previously mentioned parameters and range, we utilized a crash and/or halting punishment value of $-25000$. Defined under these conditions, the result of the final reward before crashing for the figure was 693.89, with the path seen in figure 1.

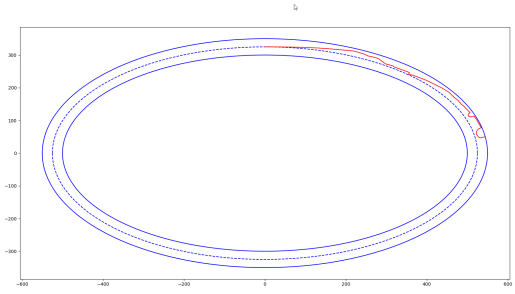

Figure 1: One of the best trials around the path.

While our Q-value approximation equation managed to converge to find a path to this distance several times, as seen in figure 3 and figure 2, finding these partial solutions was also not consistent or unique to this parameter combination. The other figures utilized different weights and parameters, yet still found a similar path. As weights were also initialized randomly, it provided some inconsistency in initial

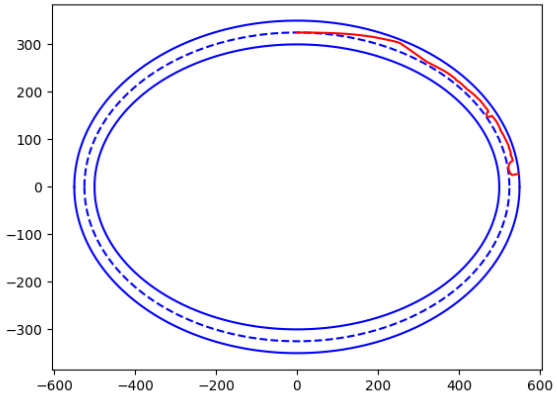


Figure 2: Another example of traversing a fourth of the track.

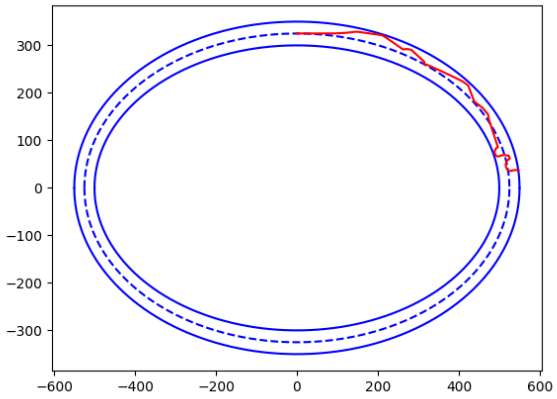


Figure 3: Traversing a fourth of the track slightly different from the other two runs.

conditions.

The weight values within the simulation also had a tendency to occasionally perform large jumps which often forced the weights to switch from positive to negative or vice-versa and increase in an order of magnitude despite small learning rates. This could suggest that the system was reaching a state where it was performing well and expected to continue to perform well, but then suddenly reached a crash. While this may be a result with our function as a result of our features, it is also possible that this results from some aspect of the track that the vehicle is travelling along. These fluctuations cause the weight values of the linear approximation equation to be unable to truly converge.

## 5 Conclusions

In conclusion, while the linear approximation we constructed for the expected reward of each state was able to direct the vehicle along the path for a fourth of the full track, it seems to be inadequately equipped to fully traverse the length of the track. The vehicle did not travel in a clean looking ellipse, but stayed within the bounds of the track for the initial fourth of the track. The function utilized seems to be able to approximate the general shape of the ellipse for the track, but was unable to adapt to the changing environment of the track, as we will detail in the following section.

## 6 Future Work

We discovered that our approximation of the shape of the elliptical track is a clear over-estimate of the size of the track, and thus the features, functions, and rewards related to the elliptical shape of the track do not accurately reflect the proper expected values of angles and velocity for a specific portion of the track. While we spent some time attempting to create a working function for approximating the shape of the ellipse, it was mostly unsuccessful. Thus, this is one major avenue of improvement available for future work on this assignment.

Further investigations into the actual performance of the vehicle on the track while seeking to modify our prediction function to include friction led us to discover and believe that the track terrain rapidly transitions as the course reaches the rightmost apex. We constructed a forced policy for the vehicle, choosing its steering angle as $\theta_{steering} = tan^{-1}(\frac{y}{.81601x}) - \frac{\pi}{2}$. The constant with the $x$-axis position was decided empirically to test for a good approximation of the elliptical shape. This value in particular travels further than .81600 or .81602 and any other deviation.

We applied no torque, and found that the vehicle does experience a friction force along the track, despite the previous findings of no friction along the straight line direction. More interestingly, the vehicle rapidly speeds up after reaching the apex of the curve. It quickly crashed into a wall. This acceleration with no applied torque is shown in table 3. It seems to suggest that this portion of the track is on a slope or some other terrain which causes an acceleration of the vehicle. One suggestion could be to implement separate policies for each portion of the track or to stick to a policy after certain points. Of course, it is also necessary to improve upon our prediction function by including a friction term as well. The result of these test policies can be seen in figures 4 and 5.

These discoveries suggest that the current simple Q-Learning algorithm is insufficient in describing the complete track. It is unlikely that any combination of weights, values, and features can adequately describe the terrain that seems to exist along the track. It may still be possible for the agent to utilize Q-Learning to traverse the track with a multiple, split policy Q-Learning, but it seems exceedingly difficult to describe the system with a single set of weights and features combinations throughout. These are the areas in which we suggest further work can be done.

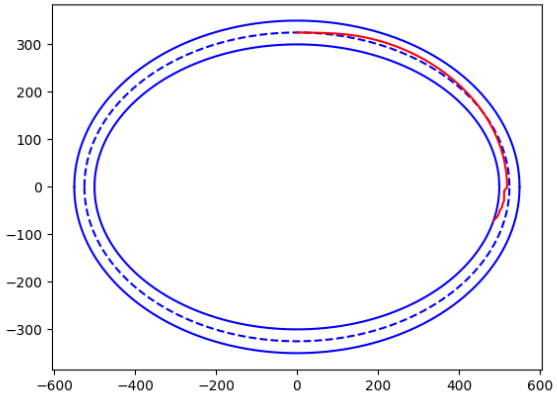


Figure 4: No applied torque, with $x$ multiplied by .81601. We see the vehicle continue to travel forward, in contrast to other multiplicative constants applied when calculating the angle.

## 7 Contributions

Together we developed the initial ideas about the organization of our code and the first feature and reward functions. James went on to write the code for the simRedux, features, reward, and Qval function, while Ashley wrote the structural code for the QFunc class. For the paper, James wrote the abstract, part of Background, part of Experiments, Results, and the second draft of Conclusions, and Future Work. Ashley wrote the Introduction, part of Experiments, part of Background, the first draft of Conclusions, Contributions, and Acknowledgements. We both proofread and edited the paper.

## 8 Acknowledgements

We would like to thank Dr. Ramanujan for his help during office hours. Also, we would like to acknowledge Caleb and Arthur, for inspiring the beginnings of testing a baseline elliptical policy to test the track properties

Table 3: Acceleration with No Torque

| x | y | $V_x$ | $V_y$ | Angle |
|---|---|---|---|---|
| 487.9311321 | -69.12312116 | -3.096369367 | -2.627732563 | -1.742690881 |
| 487.617486 | -69.38670907 | -3.169738153 | -2.642404611 | -1.743442249 |
| 487.2963182 | -69.65168088 | -3.246484183 | -2.655550067 | -1.744200386 |
| 486.9672843 | -69.91787875 | -3.326731596 | -2.667073991 | -1.744965072 |
| 486.6300274 | -70.18513495 | -3.410598978 | -2.676873389 | -1.745736065 |
| 486.2841794 | -70.45327095 | -3.498198659 | -2.684836922 | -1.746513108 |
| 485.9293612 | -70.72209665 | -3.589635943 | -2.690844726 | -1.747295919 |

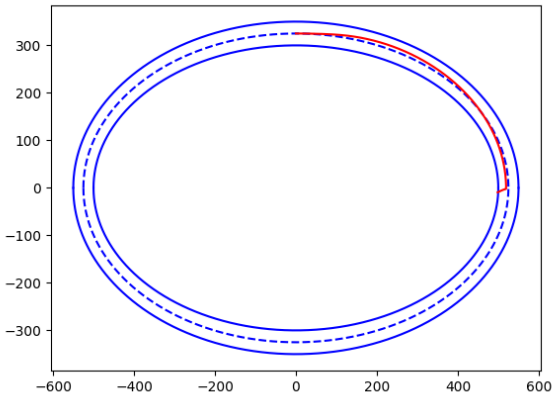


Figure 5: No applied torque policy, with $x$ multiplied by .81602. The vehicle quickly strays from its expected course.