Machine Learning Engineer Nanodegree

Capstone Proposal

Sachin Gupta

March 3, 2018

PROPOSAL

Domain Background

Table based Q-Learning (note: Not Deep Q-Learning) is a type of reinforcement machine learning. Q-Learning was first detailed in a Cambridge PhD thesis by Christopher Watkins in 1989. It is inspired by how you would normally train an animal or child. You positively reward desirable behavior and punish (negative reward) undesirable behavior.

The next problem is in the rate at which out Q-Table grows with complexity.

This is what prompted the use of an artificial neural network to approximate the behavior of a Q-Table. In 2013 DeepMind released ‘Playing Atari With Deep Reinforcement Learning’ and 2015 ‘Human-Level Control Through Deep Reinforcement Learning’.

Proposal Statement

My project proposes a deep learning approach to the ‘Train a Smart cab’ problem (Training project under the reinforcement learning section of the Machine Learning Engineer Nanodegree Program).

The agent was originally coded using the (Q-table based) Q-Learning Algorithm which encountered the growing Q-table problem due to presence of a large number of states as mentioned in the domain background section if this proposal.

Datasets and Inputs

The environment and simulator have been taken from the Train a Smart cab project, Reinforcement Learning, MLND, Udacity.

The simulator simulates an environment with a high randomness.

The state of the agent is defined in terms of the following inputs:

1. Waypoint: Denoted the direction to the destination as per the shortest path from the current location. Number of possible values = 3 (left, right, forward).
2. Input-light: Denotes the status of the traffic lights. Number of states = 2 (Red, Green).
3. Input-oncoming: Denoted the intended direction of motion of the traffic in the front. Number of possible values = 4 (Left, Right, Forward, None).
4. Input-left: Denoted the intended direction of motion of the traffic in the left. Number of possible values = 4 (Left, Right, Forward, None).
5. Input-right: Denoted the intended direction of motion of the traffic in the right. Number of possible values = 4 (Left, Right, Forward, None).

Based on the state the agent is supposed to take one of the following actions.

1. Forward
2. Right
3. Left
4. None

Solution Statement

This ideology of this project is to tackle the problem of the rate of growth of the Q-table which has to contain policies for 384 possible states hence, several models based on deep-reinforcement learning algorithms will be trained and compared.

Benchmark Model

Since the main objective of this project is to improve the performance if the Q-learning model trained in the Reinforcement Learning section of MLND. It will involve a comparative study of various and models and hence the performance will be evaluated relative to the original q-learning agent and the new deep-reinforcement learning models.

Evaluation Metrics

The driving agent will be evaluated on two metrics: Safety and Reliability as per the following table:

Grade	Safety	Reliability
A+	Agent commits no traffic violations, and always chooses the correct action.	Agent reaches the destination in time for 100% of trips.
A	Agent commits few minor traffic violations, such as failing to move on a green light.	Agent reaches the destination on time for at least 90% of trips.
B	Agent commits frequent minor traffic violations, such as failing to move on a green light.	Agent reaches the destination on time for at least 80% of trips.
C	Agent commits at least one major traffic violation, such as driving through a red light.	Agent reaches the destination on time for at least 70% of trips.
D	Agent causes at least one minor accident, such as turning left on green with oncoming traffic.	Agent reaches the destination on time for at least 60% of trips.
E	Agent causes at least one major accident, such as driving through a red light with cross-traffic.	Agent fails to reach the destination on time for at least 60% of trips.

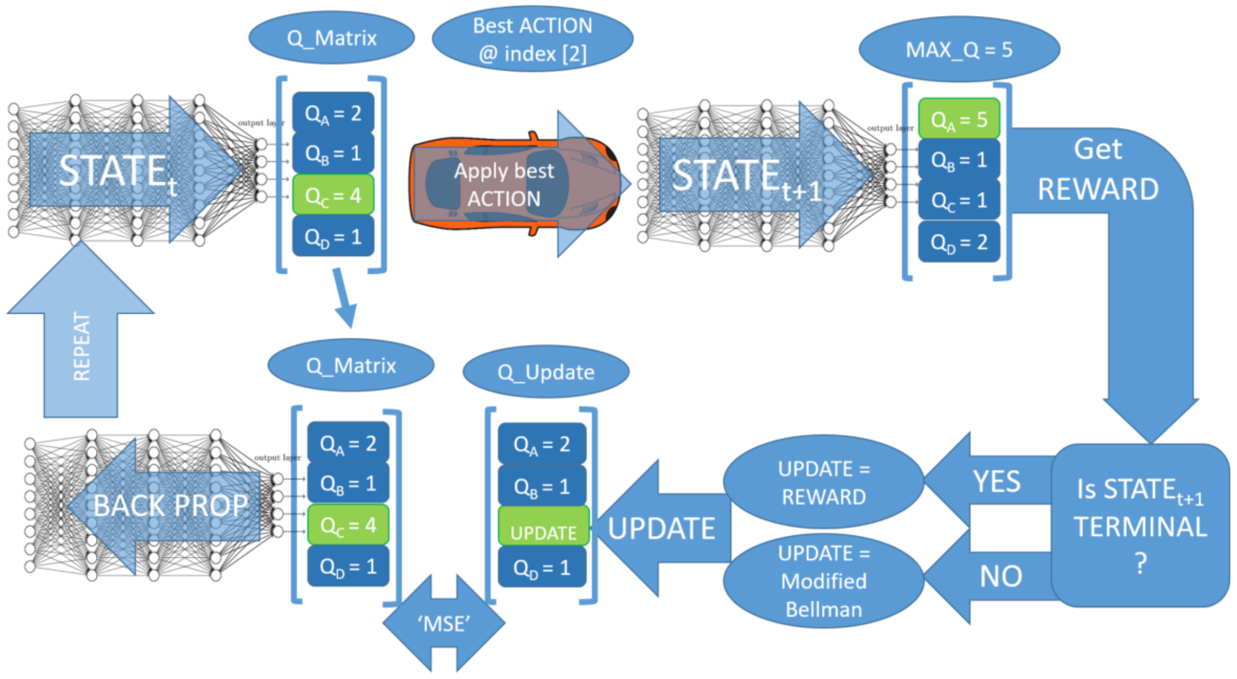
Along with this some important factors under consideration will be the training time and training efficiency in terms of arriving at an optimal policy for maximum number of states possible.

Project Design

The project involves:

1. An environment in which the agent operates.
2. A simulator that simulates that environment.
3. The agent originally trained on Table based Q-learning algorithm.
4. Agents trained on one or more deep-reinforcement learning methods.

The flowchart for a deep-reinforcement learning (here, deep q-learning) agent:



Algorithm:

```
initialize state, epsilon, alpha, epsilon
while training:
  observe state
  feed forward state through NN to get q_matrix
  if (epsilon > random_float_between_0_and_1):
    select random action
  else:
    action_idx = index of the element of q_matrix with
largest value
apply action at action
  observe next_state
  observe reward
  if next_state is terminal:
    q_update = reward
  else:
    feed forward next_state through NN to get
q_matrix_next
    observe the largest q value (q_max) in
q_matrix_next
```

```
  q_update = reward + gamma*q_max

make copy of q_matrix (q_target)
q_target[action_idx] = q_update
loss = MSE(q_matrix, q_update)
optimize NN using loss funciton
state = next_state
epsilon -= decay_rate
```