Project Title: 2D Self Driving Car
Project Author: Daniel Newman

Aim:

The aim of this experiment is to explore the use of Q Learning under a simple application for teaching an automated bot to navigate a 2D world and respond to Reinforcement Stimulus.

Hypothesis:

Using Q Learning, I expect to see a dramatic response to negative stimuli from the bot and a behaviour that adapts quickly and fluidly to a changing environment. The bot should demonstrate an aptitude for avoidance of its negative stimuli and effectively follow the path set out for it.

Experiment

Description

The experiment is a small pygame window that the user can interact with in order to section out an area to provide negative stimulus, this is referred to as 'sand' within the environment. The bot is designed to receive a positive reward from its experience when it reaches the top left or bottom right of the screen, and can be diverted using the sand to create roads & obstacles for the bot to avoid.

Requirements

Python
Python Libraries

- Numpy
- SciPy
- Reinforcement Learning Library
  - Tensorflow
  - PyTorch
  - Theano
  - etc
- OpenCV
- OpenAI Gym

Python related IDE

- Spyder
- Jupyter
- Pycharm

Experiment Specific Requirements

- Kivy
- Pygame

Steps to Reproduce

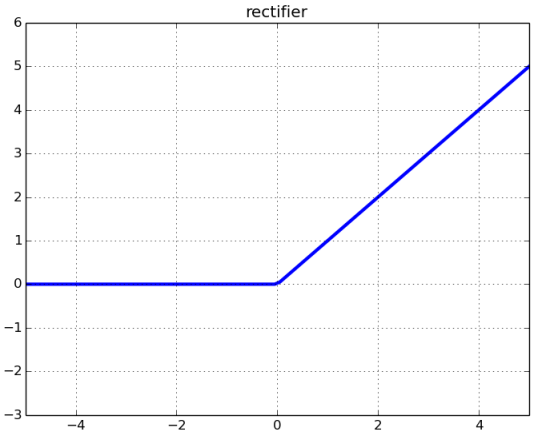
This is a simple Neural network with no image processing, it is propagated through a single hidden layer of 30 neurons before determining the appropriate action. Our hidden layer is configured with the rectifier activation method, which linearly increases as the input value increases over 0.

```
class Network(nn.Module):

    def __init__(self, input_size, nb_action):
        super(Network, self).__init__()
        self.input_size = input_size
        self.nb_action = nb_action
        self.fc1 = nn.Linear(input_size, 30)
        self.fc2 = nn.Linear(30, nb_action)

    def forward(self, state):
        x = F.relu(self.fc1(state))
        q_values = self.fc2(x)
        return q_values
```

Our neural network inputs are fed by 3 sensors on our vehicle, one in the centre and one of each side of the vehicle to determine any obstacles approaching the vehicle as it moves throughout the space. Our outputs determine whether the vehicle turns left or right or continues straight ahead, depending on what the neural network determines is the best option. The action is then reinforced through our reward function, which gives a positive or negative reward depending on where the vehicle happens to be at that frame. This allows us to sculpt the route the car will travel as it moves along its path, as a valid path will promote positive reward values over negative.



Results:

From the experiment we can define an environment for the AI to navigate around, avoiding obstacles and determining the best known path from one point to another. Over time the bot learns through exploring and reacting to its environment, promoting the required behaviour from the bot. Our bot will avoid the coloured sand and attempt to navigate from one point to another.

Discussion

When creating an artificial intelligence, we need to determine the methods by which it will respond to its environment, what its goals are within that environment and how close it is to achieving its goal. The way this is approached informs how the model is created, most models for AI are rigid and do not generalise well to new environments. Our model is based on a biological approach, derived from the study of the human brain, which can specialise and generalise at a rapid rate, without explicitly being informed about the capacities of its environment. We call this particular model a neural network, and it works similarly to the neurons in our brain. As new ideas are formed through exploring our environment, new connections are being formed between each neuron, which we refer to in our experiment as a node. These connections flow in between layers of nodes in an ever expanding network of connections, as these connections are grown and repeated, they refine and become more efficient. The same thing happens in code. As the neural network samples more data, these layers of nodes interconnect and make new connections which affect the overall value, so in order to find out how close the current output value is to the intended value, we compare what the neural network things or predicts to what the value was in reality, this creates our loss function. This loss function allows the neural network to see how far off it was from reality, reinforcing its model until it becomes correct, this happens as the model approaches zero. However the nature of the neural network also becomes its limitation, as over-training the network leads to an inability to adapt and generalise correctly and leads to what's called 'over-fitting'. This can be seen in our Q-Learning experiment if the bot is left in the same state for a while, if the environment is cleared of obstacles it takes some time before the bot reacquaints itself with the entire environment, preferring to travel its well developed path instead.

It would be interesting to see how to approach the same problem in a 3D environment using human behaviour to facilitate training data, it would be possible to create a simulation and investigate Imitation Learning for our data collection, but would require future investigation.

Conclusion

To conclude, Q-Learning serves as a fundamental starting point for Reinforcement Learning, however its limitations only facilitate its use in the simplest environments.