SCS_3547_006 Term Project

By: Phil King
ID: qq345692
August 4th, 2020

Mastering a driving game agent

Home brewed driving game

You win by completing a lap

You can die by running out of gas or hitting a wall

You reach checkpoints for points and to refill your gas tank

wasd controls: a and d rotate the car 15 degrees, w and s move the car one unit

Deep Q Learning

State: what the car sees (length of vision lines)

Action: w, a, d, wa, wd are selectable by the agent

The neural network functions as a function approximator

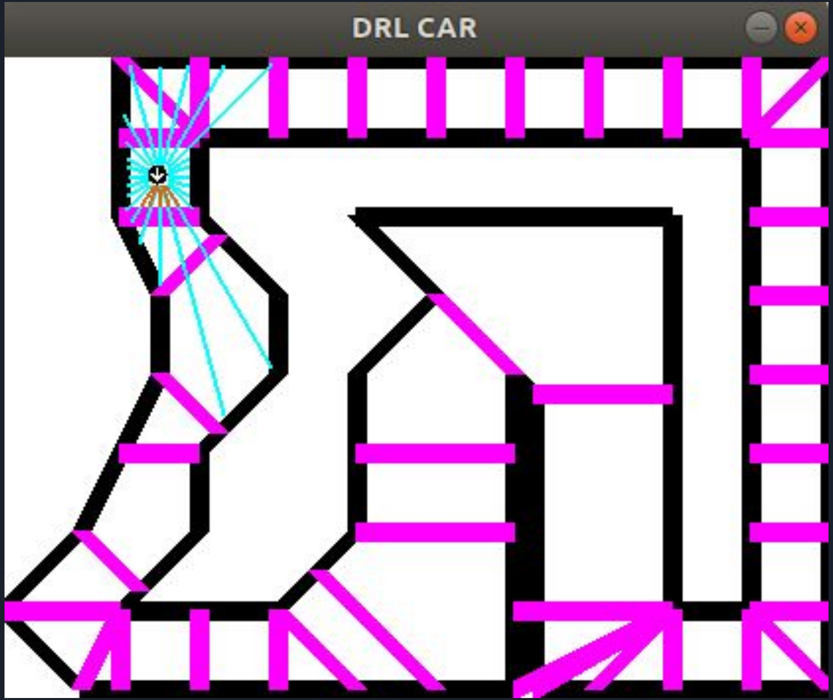
Reward: +1 if hits a checkpoint, -1 if agent dies, -0.01 as a movement cost

History: store every state into a dictionary with values for every action

At each step use Q algorithm to update reward values in the history

Periodically train the neural network on the history dataset

Visualizing the environment and the agent



Black circle: car

Arrow represents the direction

Black lines: Walls of the track

Purple lines: checkpoints

Blue lines: how the car sees walls

Brown lines: how the car sees checkpoints (can only see checkpoints in front of it)

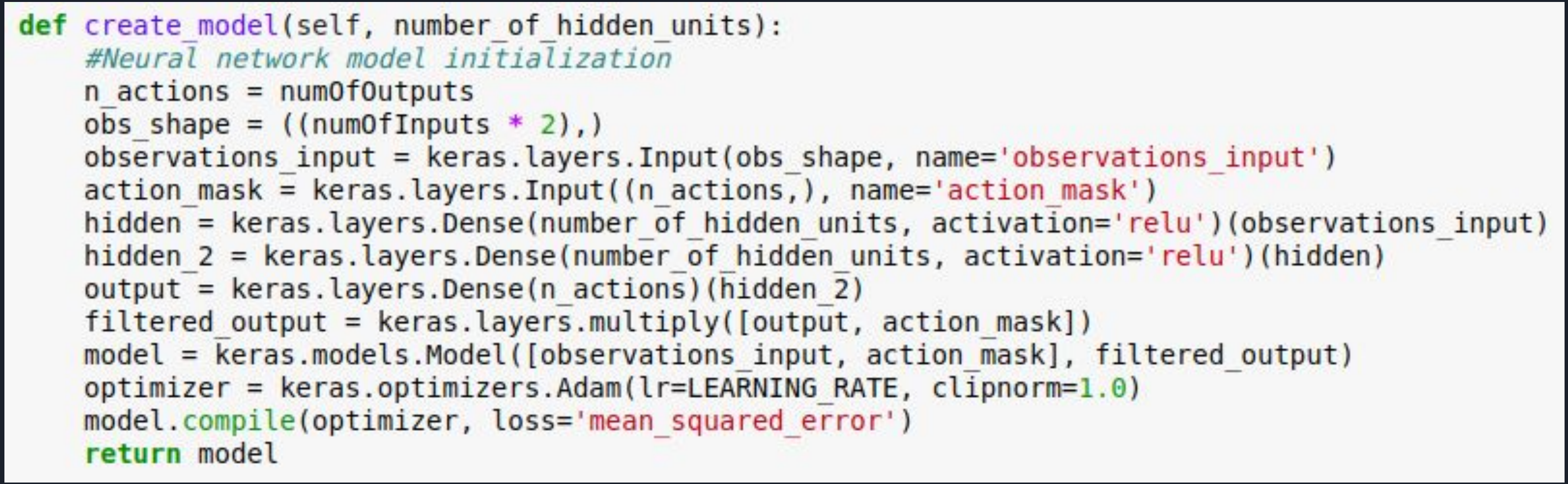
Neural Network Model

LEARNING_RATE = 0.005

numOfInputs = 24 (number of vision lines)

numOfOutputs = 5 (number of action choices)

number_of_hidden_units = 24



```python
def create_model(self, number_of_hidden_units):
    #Neural network model initialization
    n_actions = numOfOutputs
    obs_shape = ((numOfInputs * 2),)
    observations_input = keras.layers.Input(obs_shape, name='observations_input')
    action_mask = keras.layers.Input((n_actions,), name='action_mask')
    hidden = keras.layers.Dense(number_of_hidden_units, activation='relu')(observations_input)
    hidden_2 = keras.layers.Dense(number_of_hidden_units, activation='relu')(hidden)
    output = keras.layers.Dense(n_actions)(hidden_2)
    filtered_output = keras.layers.multiply([output, action_mask])
    model = keras.models.Model([observations_input, action_mask], filtered_output)
    optimizer = keras.optimizers.Adam(lr=LEARNING_RATE, clipnorm=1.0)
    model.compile(optimizer, loss='mean_squared_error')
    return model
```

Training and experiment specifics

Success is when the neural network gets the car to complete a lap with no stochasticism

Stochastic move selection: starts at 0.4 and caps at 0.9. Goes up as best attempt gets better

Every 150 episodes the NN will be trained and tests the performance of an no stochasticism attempt

Q learning step and discount parameters: 0.99

Train time is not consistent: varies between 20m-24h

This attempt took 71m of training with ~2600 episodes

Demonstration

Short url: shorturl.at/jtFM0

Full url: https://www.youtube.com/watch?v=gMF-J8oYVWM

Thanks for listening