Controlling a Dynamic Marble with Reinforcement Learning

Swetha Varadarajan

December 5, 2014

Contents

Abstract

From [1], **"Reinforcement learning is an area of machine learning inspired by behaviourist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward"**. This report deals with applying this concept in controlling a marble's motion.

1 Introduction

The problem of interest is to control a marble on a one-dimensional track that has a mass and real-valued position and velocity values. By conventional definition the states required for this task will be two-dimensional namely the position and velocity. Actions are forces on the marble. In addition to this, the desired position of the marble is also used as one of the state value. The following section describes the code and the final section talks about the results obtained.

2 Python implementation

This section describes the flow of the program followed by the modified parts and the corresponding explanation. The position, velocity and goal determines the position of the marble at each point of time or state and hence these 3 form the state variables. 3 functions have been defined for the initial, next and the reinforcement step. In the initial state, random values (0-10) is assigned for position. Velocity is zero and goal is also random. The next state is updated using a small variational parameter called the delta. The goal remains the same. The parameters of the neural network are fixed to some constant values and the experiment is conducted for several trials. The 4 inputs to the neural network layer is provided to get the best reinforcement value or the action. The delta and its gradient form the objective function for the SCG. The results obtained are standardized and used for plotting and analysing. The following shows the part of the codes that were modified from the provided code.

1. Initial state: defining goal as a random variable between 1 and 9. Used random package.

```python
def initialState():
    g=random.randint(0,9)
    return np.array([10*np.random.random_sample(), 0.0,g])
```

2. Next state: Goal remains the same. So, the copy will do the purpose. No need of any extra change except that the extreme cases be modified to copy the same.

```
 if s[0] < 0:        # Bound next position. If at limits, set velocity to 0.
        s=[0,0,s[2]]
    elif s[0] > 10:
        s = [10,0,s[2]]
```

3. Reinforcement function: Remove the goal and parametrize with s[2].

```
return 0 if abs(s1[0]-s1[2]) < 1 else -1
```

4. Apart from this, the ”repr” function in NeuralNetwork.py is commented out as per the discussion in Piazza.

5. Plotting differences:

   Plot(4,3,2),(4,3,5): The goal varies as 1,5,9 as opposed to only 5 in the initial code. So, this is provided as a parameter.

```
 plt.plot([0,X.shape[0]], [g,g],'--',alpha=0.5,lw=5)
 plt.fill_between([g-1,g+1],[-5,-5],[5,5],color="red",alpha=0.3)
```

   Standardization involves 3 states now. The neural network also has 4 input variables now.

```
qs = net.use(np.array([[s,0,g,a] for a in validActions for s in range(11)]))
 net.draw(["$x$","$\dot{x}$","$g$","$a$"],["Q"])
```

   plot (4,3,12): 3 state variables are involved.

```
 s = [x,0,g]
 xtrace = np.zeros((nStepsPerTrial,3))
```

3 Observations

This section describes the observations obtained are for the following parameters.

```
gamma = 0.5
nh = 5
nTrials = 500
nStepsPerTrial = 1000
nSCGIterations = 10
finalEpsilon = 0.01
```

Figure 1 to 3 shows the plot for goal=1,5 and 9 respectively. The first sub-plot (4,3,1) shows the decay rate versus the number of trials. The second is the goal versus state plot. Third is the list of actions that had been followed. Fourth is the mean reinforcement versus the trial number. Fifth is the action with respect to the position and velocity. Sixth is the neural network layer with 4 inputs and one output. The remaining plots are our area of interest. Plots (4,3,7) and (4,3,10) are the contour and surface plots for the reinforcement value. The maximum value represents the best taken action. It has been plotted with the velocity, position and action as the 3 axes. From the 3 figures corresponding to the 3 goal values, it can be seen that the surface plot has a raised peak surrounding the goal value. This shows that the code worked appropriately. Plots (4,3,8) and (4,3,12) are the contour and surface plots for the actions taken. There are 3 actions namely to move left or right or to remain in center. These are aptly depicted in the plots. The final sub-plot (4,3,12) is the trajectory or the convergence plot. The convergence should happen at the goal point. This convergence rate is more accurate for goal=1 and goal=5. For goal=9, the rate of convergence has decreased. By varying the parameters, the following observations were obtained.

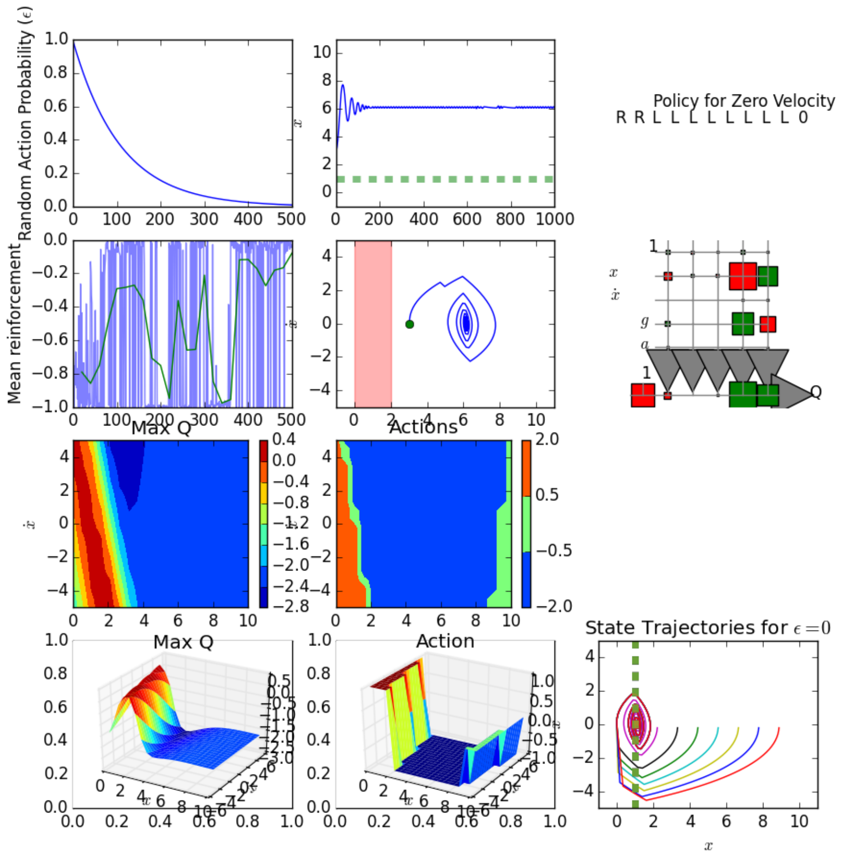

Figure 1: Plots for GOAL=1

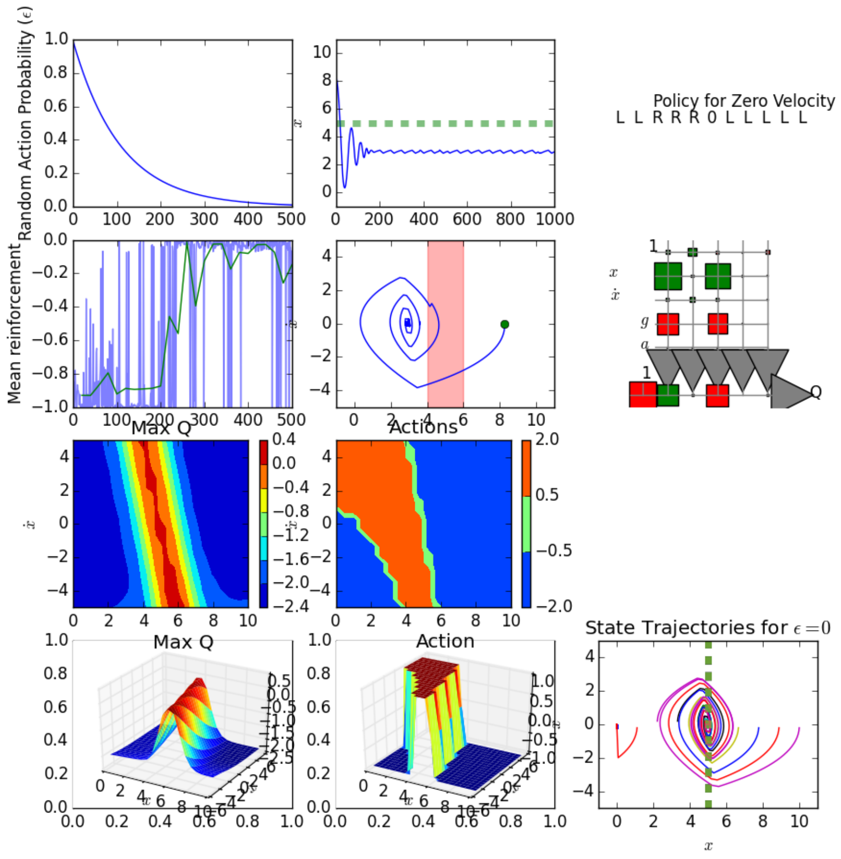

Figure 2: Plots for GOAL=5

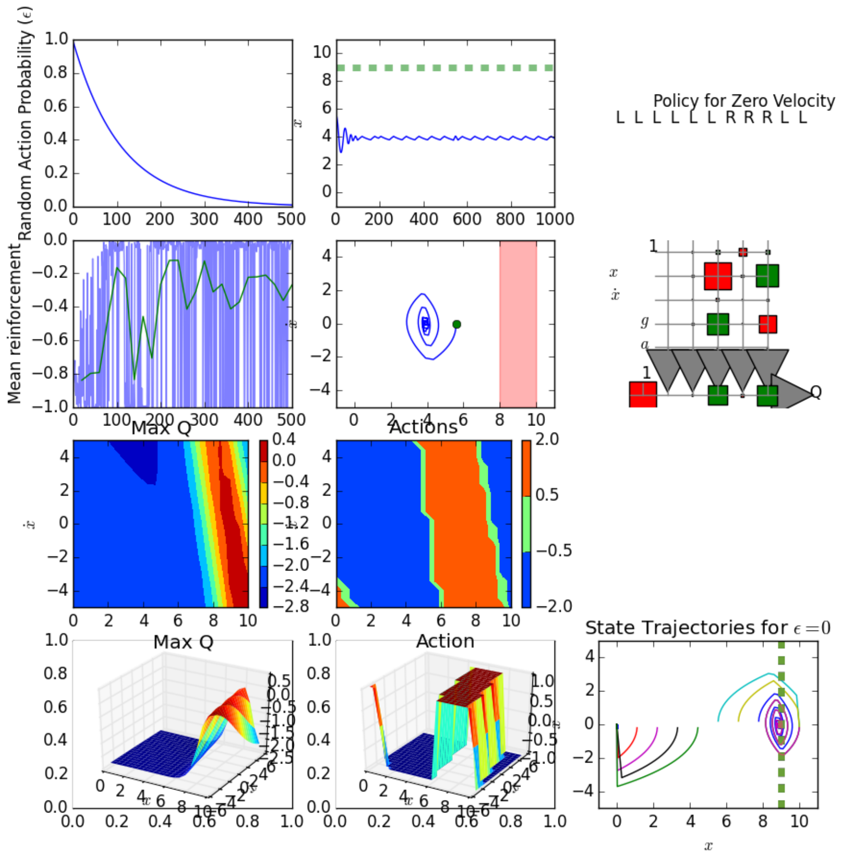

Figure 3: Plots for GOAL=9

1. Number of trials: Increasing it, improved the result.
2. Number of steps per trial: Increasing this to 2000, gave the best result as shown in figure 5.
3. Number of hidden units: When increasing to 10 or decreasing to 1, the performance was poor when compared with hidden units =5.
4. Final epsilon: When changing to 0.1, gave good result for GOAL=9 and to 0.9 yielded a funny graph as shown in figure 4. The convergence rate is not as good as expected and the action contour tells us that the tendency to stay in the same position is very less and the second sub-plot shows an oscillatory motion of the states. This sows that, it is better to have a low value of epsilon.

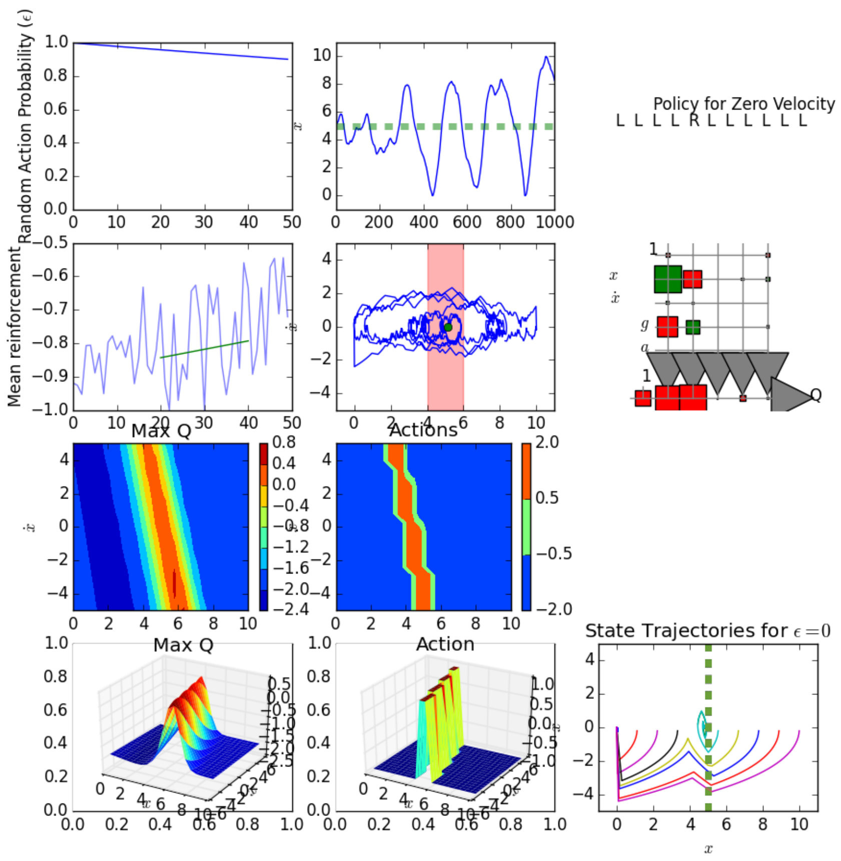

Figure 4: Plots for GOAL=5,epsilon=0.9

5. Gamma: Decreasing to 0.1, for GOAL=5, the states kept oscillating between 3 and 2 Increasing the value to 0.9 yielded better result.

Best results were obtained by retaining the original parameters and increasing the number of steps per trial to 2000.

4 Conclusion

Thus, the neural network based reinforcement learning has been applied to control the dynamic motion of a marble placed on a platform to the desired position.

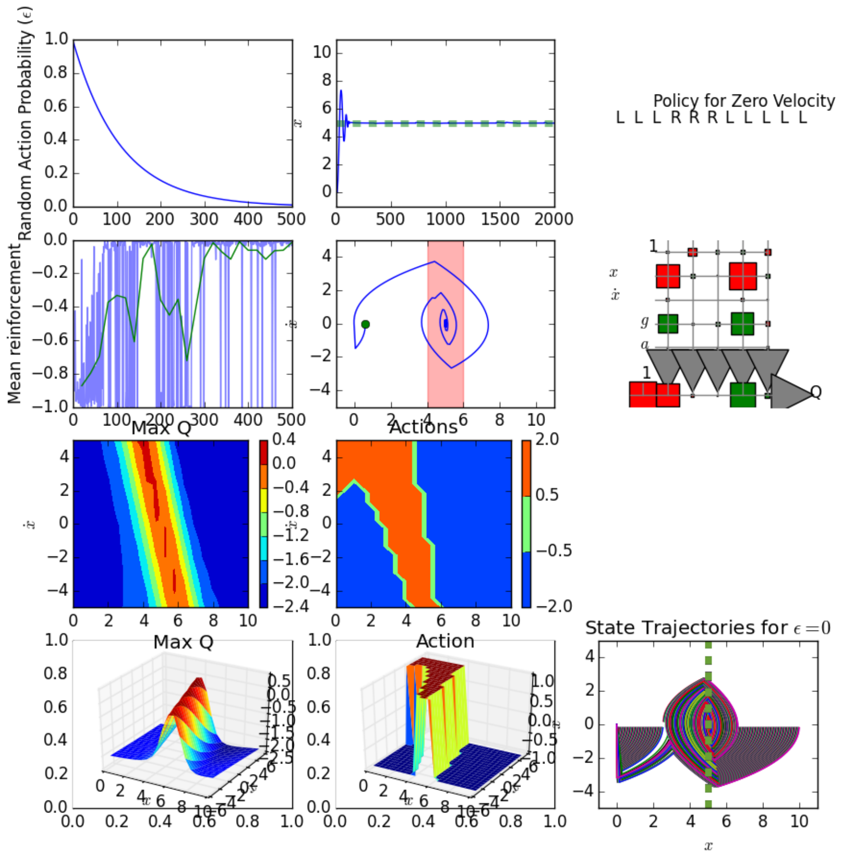

Figure 5: Plots for GOAL=5,steps per trial=2000

References

[1] http://en.wikipedia.org/wiki/Reinforcement_learning