Title: Mountain car problem

URL: https://en.wikipedia.org/wiki/Mountain_car_problem

PageID: 33998310

Categories: Category:Machine learning

Mountain Car , a standard testing domain in Reinforcement learning , is a problem in which an under-powered car must drive up a steep hill. Since gravity is stronger than the car's engine, even at full throttle, the car cannot simply accelerate up the steep slope. The car is situated in a valley and must learn to leverage potential energy by driving up the opposite hill before the car is able to make it to the goal at the top of the rightmost hill. The domain has been used as a test bed in various reinforcement learning papers.

Introduction

The mountain car problem, although fairly simple, is commonly applied because it requires a reinforcement learning agent to learn on two continuous variables: position and velocity. For any given state (position and velocity) of the car, the agent is given the possibility of driving left, driving right, or not using the engine at all. In the standard version of the problem, the agent receives a negative reward at every time step when the goal is not reached; the agent has no information about the goal until an initial success.

History

The mountain car problem appeared first in Andrew Moore's PhD thesis (1990). [1] It was later more strictly defined in Singh and Sutton's reinforcement learning paper with eligibility traces . [2] The problem became more widely studied when Sutton and Barto added it to their book Reinforcement Learning: An Introduction (1998). [3] Throughout the years many versions of the problem have been used, such as those which modify the reward function , termination condition, and the start state .

Techniques used to solve mountain car

Q-learning and similar techniques for mapping discrete states to discrete actions need to be extended to be able to deal with the continuous state space of the problem. Approaches often fall into one of two categories, state space discretization or function approximation .

Discretization

In this approach, two continuous state variables are pushed into discrete states by bucketing each continuous variable into multiple discrete states. This approach works with properly tuned parameters but a disadvantage is information gathered from one state is not used to evaluate another state. Tile coding can be used to improve discretization and involves continuous variables mapping into sets of buckets offset from one another. Each step of training has a wider impact on the value function approximation because when the offset grids are summed, the information is diffused. [4]

Function approximation

Function approximation is another way to solve the mountain car. By choosing a set of basis functions beforehand, or by generating them as the car drives, the agent can approximate the value function at each state. Unlike the step-wise version of the value function created with discretization, function approximation can more cleanly estimate the true smooth function of the mountain car domain. [5]

Eligibility traces

One aspect of the problem involves the delay of actual reward. The agent is not able to learn about the goal until a successful completion. Given a naive approach for each trial the car can only

backup the reward of the goal slightly. This is a problem for naive discretization because each discrete state will only be backed up once, taking a larger number of episodes to learn the problem. This problem can be alleviated via the mechanism of eligibility traces, which will automatically backup the reward given to states before, dramatically increasing the speed of learning. Eligibility traces can be viewed as a bridge from temporal difference learning methods to Monte Carlo methods. [6]

Technical details

The mountain car problem has undergone many iterations. This section focuses on the standard well-defined version from Sutton (2008). [7]

State variables

Two-dimensional continuous state space.

V e l o c i t y = (– 0.07 , 0.07) {\displaystyle Velocity=(-0.07,0.07)}

P o s i t i o n = (– 1.2 , 0.6) {\displaystyle Position=(-1.2,0.6)}

Actions

One-dimensional discrete action space.

m o t o r = (l e f t , n e u t r a l , r i g h t) {\displaystyle motor=(left,neutral,right)}

Reward

For every time step:

r e w a r d = – 1 {\displaystyle reward=-1}

Update function

For every time step:

A c t i o n = [– 1 , 0 , 1] {\displaystyle Action=[-1,0,1]}

V e l o c i t y = V e l o c i t y + (A c t i o n) ∗ 0.001 + cos ■ (3 ∗ P o s i t i o n) ∗ (– 0.0025) {\displaystyle Velocity=Velocity+(Action)*0.001+\cos(3*Position)*(-0.0025)}

P o s i t i o n = P o s i t i o n + V e l o c i t y {\displaystyle Position=Position+Velocity}

Starting condition

Optionally, many implementations include randomness in both parameters to show better generalized learning.

P o s i t i o n = – 0.5 {\displaystyle Position=-0.5}

V e l o c i t y = 0.0 {\displaystyle Velocity=0.0}

Termination condition

End the simulation when:

P o s i t i o n ≥ 0.6 {\displaystyle Position\geq 0.6}

Variations

There are many versions of the mountain car which deviate in different ways from the standard model. Variables that vary include but are not limited to changing the constants (gravity and steepness) of the problem so specific tuning for specific policies become irrelevant and altering the reward function to affect the agent's ability to learn in a different manner. An example is changing the reward to be equal to the distance from the goal, or changing the reward to zero everywhere and one at the goal. Additionally, a 3D mountain car can be used, with a 4D continuous state space. [8]

References

Implementations

C++ Mountain Car Software. Richard s. Sutton.

Java Mountain Car with support for RL Glue

Python, with good discussion (blog post - down page)

Further reading

Sutton, Richard S. (1996). Mountain Car with Sparse Coarse Coding . Advances in Neural Information Processing Systems. MIT Press. pp. 1038– 1044. CiteSeer x : 10.1.1.51.4764 .

Mountain Car with Replacing Eligibility Traces

"More discussion on Continuous State Spaces". 2000. pp. 903– 910. CiteSeerX 10.1.1.97.9314 .

Gaussian Processes with Mountain Car