Machine Learning Engineer Nanodegree Capstone Proposal

Charlotte Dieter-Ridder, May, 5th, 2017

Implementing a learning agent for the “Mountain Car problem”

Domain Background

Reinforcement Learning (RL) is a fascinating area of machine learning:

Wikipedia (https://en.wikipedia.org/wiki/Reinforcement_learning): "*Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward*."

A problem is modelled as interaction between environment and agent. The agent is the learner. It gets feedback from the environment. Based on its status and on the feedback the agent chooses it next action. The feedback a reward, and the agent tries to maximize the rewards earned over the time.

Other than supervised learning RL need no large data sets. Instead the agent uses the feedback from the environment to learn how to act in an optimal manner.

RL got a lot of publicity when Google's DeepMind created AlphaGo won a Go game against the world greatest Go player. For real live there are other areas more important. In [Sutton,Barto] as areas are mentioned:
"*Beside impressive results in games (Go, checkers, backgammon, atari) there are physical control tasks for robots, dynamic random access memory (DRAM) for computers, personalized web services such as the delivery of news articles or advertisements, modelling the modeled the thermal soaring problem (birds, gliders)*."

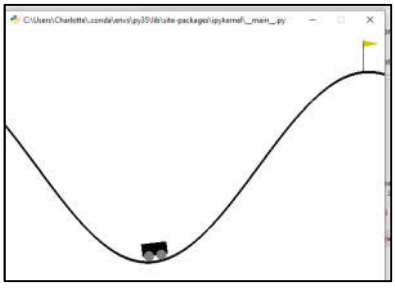
RL algorithms are able to approximate complex optimization problems which cannot be solved otherwise because of their complexity (e.g. Backgammon has around 10^{20} states).

For me this was the most fascinating part of the course but my feeling is that I'm still at the very beginning in this area. To gain some more experience I'll work on a "classic problem" in this area. I hope to get a better understanding of possible solutions and about RL in general.

Problem Statement

One of the classic RL optimization problems is the "mountain car" problem:

In [Wiki_MountainCar] the problem and its history are described:
"*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.*"



This problem has some very interesting aspects:

- The status of the car is continuous: velocity and position. They must be converted first to a discrete number of states before the known RL algorithms can be used.
- Before reaching the top of the hill the car has to drive up the neighbor to get enough momentum. The agent has increase the distance to the goal before reaching it.
- Only if the top of the hill is reached, the reward will be “1”. Otherwise it will be always “-1”

Datasets and Inputs

As this is a RL problem, no data sets will be necessary. The agent will learn from the environment. To base the project on a common environment I will use the implementation provided by [OpenAI].

Solution Statement

I plan to implement first the agent as described in [Sutton,Barto] based on python and its standard libraries and understand how changes of the parameters affect the effectivity of the learner. As a second step (optional) I would like to implement an agent as well based on tensorflow to get a feeling of the differences.

Benchmark Model/Evaluation Metrics

MountainCar-v0 is considered "solved" when the agent obtains an average reward of at least -110.0 over 100 consecutive episodes. Evaluation will be done by uploading the agent to OpenAI. They will provide a standardized evaluation.

Project Design

For the project, I see as steps:

- **Preparation:**
  Get the environment running and understand the “classic” approach for the solution. According to [Sutton,Barto] using “Semi-Gradient Sarsa” is recommended. It differs by several aspects from the Q-Learning algorithm as discussed during the course.
- **Implement this solution “straight forward”:**
  based on python and the typical libraries (numpy, pandas, ..)
- **Implement the solution based on Tensor flow:**
  Tensor flow is a quite different development environment promising to handle matrix, neural networks faster and promise an easy migration to a GPU.
- Comparison of the two implementations

Appendix

[Sutton,Barto]: Reinforcement Learning: An Introduction, Second edition, in progress, Draft, Richard S. Sutton and Andrew G. Barto, 2016

[OpenAI]: https://gym.openai.com

[wiki_MountainCar]: https://en.wikipedia.org/wiki/Mountain_Car