Reinforcement learning

Summary

This is a section on Reinforcement Learning. It comprises: the concepts of Markov Decision Process, the Bellman formula and Q-Learning algorithm.

Overview

An overview of the Reinforcement Learning field can be found at https://towardsdatascience.com/reinforcement-learning-an-introduction-to-the-concepts-applications-and-code-ced6fbfd882d

RL is a non-supervised machine learning concept and is based on the idea of a reward signal that the agent uses to evolve to an optimal solution.

Principle: all goals can be described by the maximalization of the anticipated accumulated reward.

Field: RL is about making decision for a move (transition, action, etc.) , i.e. when solving mazes, learning to play a game, self-driving cars, etc.

Markov Decision Process (MDP)

A Markov state is defined by a state in which it is assumed that the current state captures all information from previous states, therefore a transition (or action) on the current state is the same as one would have decided to make that transition by evaluating the entire history of states, i.e. the current state suffices to make a correct decision.

- A Markov process (or Markov chain) is just a sequence of Markov states. Identified by A set of States and a set of actions.
- A Markov Reward process is a Markov Chain, plus a Reward value (a scalar)
- A State Value function, v(S), is the expected long term value of a state.
- The Bellmann Expectation Equation is often used to define a State Value. The Equation takes into account Immediate reward, R(t) and the discounted (gamma) value of the successor state, gamma S(t+1) or s apostrophe

  $$v(s) = R_s + \gamma \sum_{s'} P \; v(s')$$

  Summation of the value estimation (P) of all actions possible in the next state.

An MPD is characterized by the following 5 elements

- S : set of states
- A : set of actions
- P : state transition probability matrix
- R : reward function
- Gamma : discount factor.

Value of an action (Q)

q(s,a) is the value of all actions performed resulting in the current state s.

Dynamic programming

An MPD can be used to solve dynamic programming problems. There are 2 major uses of an MPD

Monte Carlo learning

Is used to evaluate the performance (accuracy) of a policy

Temporal Difference learning

TD learns directly from experience and interaction with the environment. It relies on finetuning a guess of the value function (this is called bootstrapping).

The Q learning algorithm is a TD algoritm. The value of a state is updated (increased/decreased) by the difference of the current value, v(t), and the expected value at t+1, v(t+1)

$$\mathrm{q}(s_t a_t) \mathrel{+}= \alpha \left(r_{t+1} + \gamma MaxArgs_a \, \mathrm{q}(S_{t+1}, a) - \mathrm{q}(s_t a_t) \right)$$

$$\mathrm{V}(S_t) \mathrel{+}= \alpha \left(R_{t+1} + \gamma \mathrm{V}(S_{t+1}) - \mathrm{V}(S_t) \right)$$

So, the value of the next state is estimated using the bellman function (gamma maxargs) and the current value is subtracted to obtain an increment, upon which a learning rate (alpha) is applied.

- Q : q-value or utility of a stet
- S : state (at time t or t+&)
- A: action (at time t or t+1)
- Alpha : learning rate, e.g. 0.7 (higher cause more fluctuation)
- Gamma : discount rate (higher is favour the estimate or utility of next state)
- MaxArgs : maximum of all values over all action in that state

Q Learning algoritm and gridworld

The Q learning formula is restructured as follows

$$\mathrm{q(s,a)} = (1 - \alpha)\mathrm{q(s,a)} + \alpha(R_{t+1} + \gamma MaxArgs_a \, \mathrm{q}(s_{t+1}, a))$$

Typical values

- Alpha = 0.7

- Gamma = 0.99
- R can be used to define a movement cost, i..e set it to -1
- Epsilon : between 0 and 1

```
GridCellDTO targetCell = null;
targetCell = selectNextAction( explore );

// q(s,a) = (1 - alpha) q(s,a) + alpha ( Reward(t+1) + gamma * Max q(s(t+1),a) )
double maxQSAnext = targetCell.getMaxQSA();
double oldqsa = currentCell.getQSA( lastMovementDirection );
double reward = targetCell.getReward();
double newqsa = (( 1 - alpha) * oldqsa) + alpha * ( reward  + (gamma *maxQSAnext)
);
currentCell.setQSA( lastMovementDirection , newqsa );
```

Epsilon greedy

A run can be performed by randomly selecting an action (direction) and continue until the target cell is reached. The q values (one per action or direction) are updated as the process evolves. By performing a number of runs, one will reach on optima.

For example a grid of 3 by 3, starting point left bottom corner and target is right upper corner.

The grid is optimally traverse by following the direction with the greatest q-value.

```
[          7,8110        ]  [          8,9000        ]  [                        ]
[  7,8110          8,9000]  [  7,8110         10,0000]  [                        ]
[          6,7329        ]  [          7,8110        ]  [                        ]

[          7,8110        ]  [          8,9000        ]  [          10,0000       ]
[  6,7329          7,8110]  [  6,7329          8,9000]  [  7,8110          8,9000]
[          5,6656        ]  [          6,7329        ]  [           7,8110       ]

[          6,7329        ]  [          7,8110        ]  [           8,9000       ]
[  5,6656          6,7329]  [  5,6656          7,8110]  [  6,7329          7,8110]
[          5,6656        ]  [          6,7329        ]  [           7,8110       ]
```

Epsilon is a value between 0 and 1. It is used to decide whether the process will explore or exploit its previously gathered knowledge. The decision is just a generating a random value and comparing it to the epsilon value. If greater then a random move is made, else the move corresponds to the direction with the largest q-value. (this is performed in the selectNextAction() function in the above Java code)

Appendix

https://deeplizard.com/learn/video/mo96Nqlo1L8

Also look for the online courses by Jacon Schrum, Southwestern university.

deeplizz
https://urldefense.proofpoint.com/v2/url?u=https-3A__www.youtube.com_watch-3Fv-3DqhRNvCVVJaA&d=DwIDaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=kMTStfUdA_2sHCZBG-Ae91vQMig-R_aOm2otssp8Bzo&m=csILdQ0u1tNX0xbiH_ncBHkOlMke55HypmCvq715C60&s=PILQxMbVk2MghibrTeeFyYmQWb7kk5DEFihtQZA_8sw&e=
https://urldefense.proofpoint.com/v2/url?u=https-3A__www.youtube.com_watch-3Fv-3Dmo96Nqlo1L8&d=DwIDaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=kMTStfUdA_2sHCZBG-Ae91vQMig-R_aOm2otssp8Bzo&m=csILdQ0u1tNX0xbiH_ncBHkOlMke55HypmCvq715C60&s=qNe7749bNwKXuXXwImjXLhwg307qNv1P1MkDEkvw7Bg&e= (1- alpha) aanpak
https://urldefense.proofpoint.com/v2/url?u=https-3A__www.youtube.com_watch-3Fv-3DHGeI30uATws&d=DwIDaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=kMTStfUdA_2sHCZBG-Ae91vQMig-R_aOm2otssp8Bzo&m=csILdQ0u1tNX0xbiH_ncBHkOlMke55HypmCvq715C60&s=EaB2mvO_HtDXbu8f9u2j-UqwTrsQLbN_uIGCgFO8Wi8&e= (python)

Jacob Scrum
https://urldefense.proofpoint.com/v2/url?u=https-3A__www.youtube.com_watch-3Fv-3D3T5eCou2erg&d=DwIDaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=kMTStfUdA_2sHCZBG-Ae91vQMig-R_aOm2otssp8Bzo&m=csILdQ0u1tNX0xbiH_ncBHkOlMke55HypmCvq715C60&s=0uLmt9GlFd3uD-CAizWA_NaTi8g_mJORgJPxlhXPoVY&e= (expected values)
https://urldefense.proofpoint.com/v2/url?u=https-3A__www.youtube.com_watch-3Fv-3DbHeeaXgqVig&d=DwIDaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=kMTStfUdA_2sHCZBG-Ae91vQMig-R_aOm2otssp8Bzo&m=csILdQ0u1tNX0xbiH_ncBHkOlMke55HypmCvq715C60&s=lCe5oSxv7WdxnBB-pi1cq_JyiWYHPALy4tMbeKHzrfI&e= (rare video q value wordt uitgeld)
https://urldefense.proofpoint.com/v2/url?u=https-3A__www.youtube.com_watch-3Fv-3D1XRahNzA5bE&d=DwIDaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=kMTStfUdA_2sHCZBG-Ae91vQMig-R_aOm2otssp8Bzo&m=csILdQ0u1tNX0xbiH_ncBHkOlMke55HypmCvq715C60&s=YaGYlsdcD6InuBFKyp9BQNmpr47Lk-F0TAoLk-ZogGM&e= (berekenen van de q value)
https://urldefense.proofpoint.com/v2/url?u=https-3A__www.youtube.com_watch-3Fv-3DXrxgdpduWOU&d=DwIDaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=kMTStfUdA_2sHCZBG-Ae91vQMig-R_aOm2otssp8Bzo&m=csILdQ0u1tNX0xbiH_ncBHkOlMke55HypmCvq715C60&s=RlPPdilOPSOAH6HKlktLFU-pSo4oIq3S97tWGSfIdbU&e= (gamma, learning rate)