A Short Introduction To Some Reinforcement Learning Algorithms

By Hado van Hasselt

Continuous Actor-Critic Learning Automaton (Cacla)

Previous -- Up -- Next

Cacla can be viewed as the continuous version of the Acla algorithm. The idea is that again state values are stored in a table or function approximation. However, instead of also storing state-action values, Cacla stores a single value for each state: an approximation for the optimal action. Of course, in problems with multi-dimensional actions, this can be a vector instead of a value.

The idea and implementation of Cacla is very simple. You get the output of your actor. Then, you explore around this value (for instance with gaussian exploration). Then, if the value of the state increases after performing the action, you update your actor towards this action:

$$V_{t+1}(s_t) \xleftarrow{\beta_t} r_t + \gamma V_t(s_{t+1})$$

If

$$V_{t+1}(s_t) > V_t(s_t)$$

then

$$A_{t+1}(s_t) \xleftarrow{\alpha_t} a_t$$

If the state value decreases, the action was not such a good idea, and you do not update the actor.

Advantages

- Can handle both discrete and continuous action spaces.
- Does not require segmentation of state or action space.
- Fast and easy to implement.
- State values are easily extendable to eligibility traces.
- Has been shown to outperform several discrete algorithms (Q-learning, Sarsa, etc.) on problems with discrete action spaces
- Has been shown to outperform several continuous algorithms (CMA-ES, NAC, wire fitting, ADHDP) on problems with continuous action spaces.

Disadvantages

- May be less applicable to problems with a discrete, nominal action space.

Algorithm

The Cacla algorithm in schematic form:

```
Initialize $A_0(s)$, for all $s$
Initialize $V_0(s)$, for all $s$
Select $s_0$
For each step $t = 0, 1, 2, \ldots$:
  Select $a_t$ with exploration around $A_t(s_t)$ (i.e. gaussian exploration)
  Perform $a_t$, observe $r_t$, $s_{t+1}$
  If $s_{t+1}$ is terminal:
    $V_{t+1}(s_t) \xleftarrow{\beta_t} r_t$
    Select new $s_{t+1}$ (starting point for next episode)
  else:
    $V_{t+1}(s_t) \xleftarrow{\beta_t} r_t + \gamma V_t(s_{t+1})$
  If $V_{t+1}(s_t) > V_t(s_t)$:
    $A_{t+1}(s_t) \xleftarrow{\alpha_t} a_t$
```

Comparing the algorithm to Acla shows definite similarities. However, note that Cacla does not update when the state value decreases. This is different from Acla, but was shown to be a better choice for Cacla, since updating away from the action that was selected does not guarantee that you are updating towards an action that is better than the current output of the actor. For a more detailed discussion, see the second publication below. The second publication also shows variations of Cacla and compares the algorithm to other continuous action algorithms.

The output of the actor of Cacla can in principle be any continuous action value or action vector. However, also when these values get rounded to a limited, finite action space, Cacla can outperform discrete algorithms such as Q-learning. See the first publication below for details. See also this page from my recent book chapter. In that chapter, I compare Cacla to state-of-the-art algorithms on a double pole balancing task. The algorithms are the policy-gradient algorithm called natural actor critic (NAC) and the evolutionary strategy algorithm called CMA-ES. Cacla outperforms both those algorithms by a wide margin, quickly reaching better performance levels than either at lower amounts of experience and at lower computational costs. This shows that Cacla is a promising algorithm for continuous domains.

One potential drawback of the Cacla algorithm - and of all other non-linear optimization algorithms - is that the algorithm can get stuck in a relatively poor local optimum. The chance this happens can be reduces by using more than one actor. For details on how this can be done, see Chapter 7 of my dissertation and Section 3.2.4 from the aforementioned book chapter.

Selected relevant publications:

- Hado van Hasselt (2011).
  "Reinforcement Learning in Continuous State and Action Spaces". *A book chapter to be published in* Reinforcement Learning: State of the Art, Springer.
  PDF, HTML
- Hado van Hasselt (2011).
  "Insights in Reinforcement Learning". Ph.D. thesis.
- Hado van Hasselt and Marco Wiering (2009). "Using Continuous Action Spaces to Solve Discrete Problems". Proceedings of the International Joint Conference on Neural Networks, IJCNN 2009, Atlanta, GA, USA, 2009.
- Hado van Hasselt and Marco Wiering (2007). "Reinforcement Learning in Continuous Action Spaces". Proceedings of IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL07), Honolulu, HI, USA, pp. 272-279, 2007.

Quick links:

Previous -- Up -- Next

Contact

My contact data can be found here.