Deep Reinforcement Learning
Discussion I:
Neuro-Dynamic Programming

Adalbert Mwombeni

Mechatronics in Mechanical and Automotive Engineering
Mechanical and Process Engineering
University of Kaiserslautern

15.12.2017

Motivation

Let us tell a human to go and pick that cake.

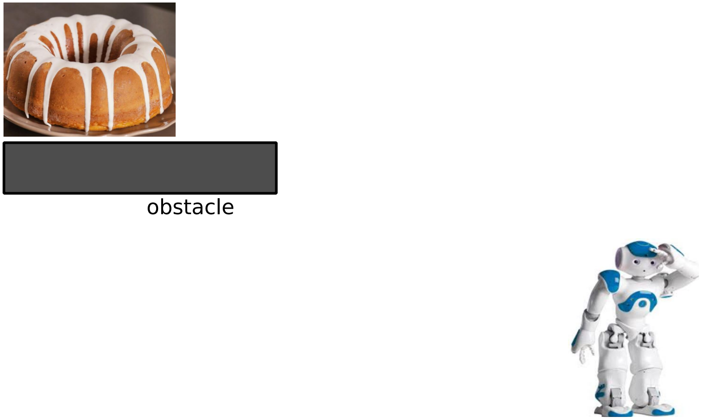
Motivation

Let us tell a robot to go and pick that cake.

- turn left
- go 7 steps
- go straight 14 steps
- grab cake

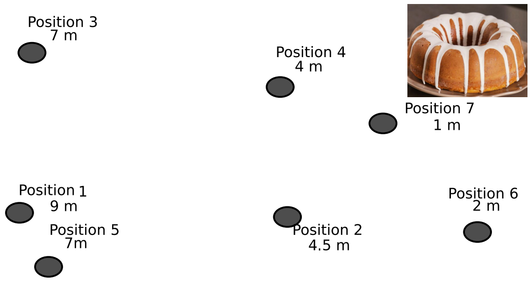
Motivation

Let us tell a robot to go and pick that cake.



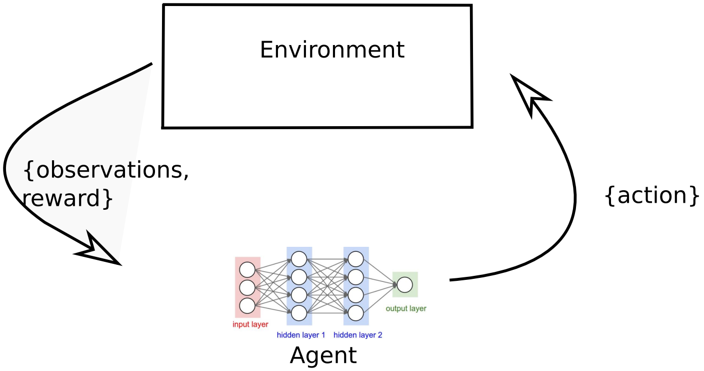
Motivation

Let us tell a robot to go and pick that cake.



- Move around
- Calculate a distance to a cake
- Pick a direction where the distance decreases
- Take a step in that direction
- Repeat

Deep Reinforcement Learning



The agent receives a positive or negative reward at every time step when the goal is not yet reached.

Deep Reinforcement Learning

- Combining deep neural network with reinforcement learning.
- Learn to act from high dimension inputs.
- To deal with credit assignment problem.

An reinforcement learning agent includes following components:

- Policy: it characterizes how the agent picks actions.
- Value function: it shows how good to be in a particular state, how much reward the agent is expected to get.
- Model: it characterizes how the agent thinks the environment works.

Policy π

Policy: a strategy that allows an agent to take the history (observation) and choose an action.
State: a summary of history $(o_1, a_1, r_1, o_2, a_2, r_2, ...)$. S denotes the states or model abstraction of the environment such that:

$$a_t = \pi(S_t)$$

Let us define some quantity:

- That quantity defines the value of being in a particular state S_0 at time 0 (initial decision). S_0 is input.
- 1.We want to predict the sum of all events that will happen in the future. This quantity is called return.

.

- $\sum r_t\left(S_t, \pi(S_t), S_{t+1}\right)|S_0$

Let us define some quantity:

- That quantity defines the value of being in a particular state S_0 at time 0 (initial decision). S_0 is input.
- 1.We want to predict the sum of all events that will happen in the future. This quantity is called return.
- 2.We do not know future states. We need to do the computation regardless how future states will be.

.

- $\mathbb{E}_{S_1,S_2,S_3,\ldots}\left[\sum r_t\left(S_t, \pi(S_t), S_{t+1}\right)|S_0\right]$

Let us define some quantity:

- That quantity defines the value of being in a particular state S_0 at time 0 (initial decision). S_0 is input.
- 1.We want to predict the sum of all events that will happen in the future. This quantity is called return.
- 2.We do not know future states. We need to do the computation regardless how future states will be.
- 3.The sum of rewards may be go till infinity time steps.

.

- $\mathbb{E}_{S_1,S_2,S_3,\ldots}\left[\sum_{t=0}^{\infty} r_t\Big(S_t, \pi(S_t), S_{t+1}\Big)|S_0\right], \qquad 0, \infty$

Let us define some quantity:

- That quantity defines the value of being in a particular state S_0 at time 0 (initial decision). S_0 is input.
- 1.We want to predict the sum of all events that will happen in the future. This quantity is called return.
- 2.We do not know future states. We need to do the computation regardless how future states will be.
- 3.The sum of rewards may be go till infinity time steps.
- 4. In order to prevent that the sum blows up, let us use the discount factor $\gamma \in (0, 1)$.
- $\mathbb{E}_{S_1, S_2, S_3, \ldots} \left[\sum_{t=0}^{\infty} \gamma^t r_t \left(S_t, \pi(S_t), S_{t+1} \right) | S_0 \right]$

Value Function

$$V^{\pi}(S_0) = \mathbb{E}_{S_1, S_2, S_3, \ldots} \left[\sum_{t=0}^{\infty} \gamma^t r_t \Big(S_t, \pi(S_t), S_{t+1} \Big) | S_0 \right] \quad (1)$$

- $a_t = \pi(S_t)$
- S_0 is the initial state, it is given. One takes S_0 and evaluates its value, ...
- The value function V depends on policy π. Starting in a good place and using a bad policy may lead to bad results.

How to compute such equation?

Content

1. Dynamic Programming
2. Q-Learning Simulation
3. Neuro-Dynamic Programming

Value Function

- The value function by assuming a particular policy $V^{\pi}(S_0)$,

$$V^{\pi}(S_0) = \mathbb{E}_{S_1,S_2,S_3,\ldots}\left[\sum_{t=0}^{\infty} \gamma^t r_t\Big(S_t, \pi(S_t), S_{t+1}\Big)|S_0\right]$$

- The value function by assuming the optimal policy is $V^{*}(S_0)$.
- V^{*} is the best value function (optimal value)
- Since the function is depending on π, this will be changed till the optimal value is obtained:

$$V^{*}(S_0) = \max_{\pi} \mathbb{E}_{S_1,S_2,S_3,\ldots}\left[\sum_{t=0}^{\infty} \gamma^t r_t\Big(S_t, \pi(S_t), S_{t+1}\Big)|S_0\right]$$

Optimal Value Function

$$V^*(S_0) = \max_{\pi} \mathbb{E}_{S_1,S_2,S_3,\dots} \left[\sum_{t=0}^{\infty} \gamma^t r_t \Big(S_t, \pi(S_t), S_{t+1} \Big) | S_0 \right]$$

- 1. Step: Take the zero term out of the bracket, with $a_0 = \pi(S_0)$

$$V^*(S_0) = \max_{\pi, a_0} \mathbb{E}_{S_1,S_2,S_3,\dots} \left[r_0(S_0, a_0, S_1) + \sum_{t=1}^{\infty} \gamma^t r_t \Big(S_t, \pi(S_t), S_{t+1} \Big) | S_0 \right]$$

- 2. Step: Linearity of the expectation

$$V^*(S_0) = \max_{a_0} \mathbb{E}_{S_1} \left[r_0(S_0, a_0, S_1) + \max_{\pi} \mathbb{E}_{S_2,S_3,\dots} \left\{ \sum_{t=1}^{\infty} \gamma^t r_t \Big(S_t, \pi(S_t), S_{t+1} \Big) | S_1 \right\} | S_0 \right]$$

Value Function

- 3. Step: Divide and multiply by γ

$$V^*(S_0) = \max_{a_0} \mathbb{E}_{S_1}\left[r_0(S_0, a_0, S_1) + \underbrace{\gamma \max_{\pi} \mathbb{E}_{S_2, S_3, \dots} \left\{ \sum_{t=1}^{\infty} \gamma^{t-1} r_t \Big(S_t, \pi(S_t), S_{t+1} \Big) | S_1 \right\}}_{\text{The value like the one started with at beginning}} | S_0 \right]$$

$$= \max_{a_0} \mathbb{E}_{S_1}\left[r_0(S_0, a_0, S_1) + \gamma V^*(S_1) | S_0 \right]$$

- 4. Step: Bellman's equation

$$V^*(S) = \max_{a} \mathbb{E}_{S'}\left[r(S, a, S^{'}) + \gamma V^*(S^{'}) | S \right]$$

- S: the current state
- $S^{'}$: the future state

Value Function

- 5. Step: Multiply the learning rate to η and add the value function of any possible value function.

$$V^{\pi}(S) + \eta V^{\pi}(S) = \mathbb{E}_{S'}\left[r(S, a, S^{'}) + \gamma V^{\pi}(S^{'})|S\right]\eta + V^{\pi}(S)$$

$$V^{\pi}(S) = V^{\pi}(S) + \eta\left[\mathbb{E}_{S'}\left\{r(S, a, S^{'}) + \gamma V^{\pi}(S^{'})|S\right\} - V^{\pi}(S)\right]$$

- 6. Step: To use the sample instead of expectation.

$$V^{\pi}_{t+1}(S) = V^{\pi}_t(S) + \eta\Big[r(S, a, \tilde{S}^{'}) + \gamma V^{\pi}_t(\tilde{S}^{'})|S - V^{\pi}_t(S)\Big] \qquad (2)$$

- The algorithm can be implemented as equation (2).
- The policy will be constructed by using value function.
- Since the number of sample goes to infinity, the iteration converges to the optimal value. The convergence proof is given by Watkins (1989).

Action-Value Function

- The value-action function depends not only on initial state but also on action that agent has taken.
- The way to move from value function to action-value function is by taking the maximum of value function.

$$V(S) = \max_{a'} Q(S, a')$$

$$V^*(S) = \max_{a} \mathbb{E}_{S'} \left[r(S, a, S') + \gamma V^*(S') | S \right]$$

- Why action-value function?

Action-Value Function

$$V^*(S) = \max_a \mathbb{E}_{S'}\left[r(S, a, S^{'}) + \gamma \max_{a^{'}} Q^*(S^{'}, a^{'})|S\right] \tag{3}$$

- The equation (3) is possible regardless that the policy is optimal or not.
- Since:

$$V(S) = \max_{a^{'}} Q(S, a^{'})$$

$$Q^*(S, a) = \mathbb{E}_{S'}\left[r(S, a, S^{'}) + \gamma \max_{a^{'}} Q^*(S^{'}, a^{'})|S, a\right] \tag{4}$$

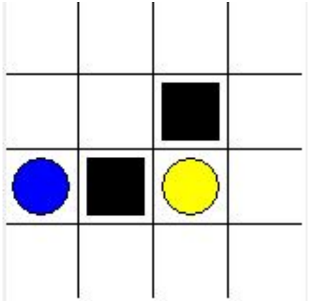
- The equation (4) is exactly the same as the one shown by the value function but instead of looking into value of state, here we have to deal with the state and action.
- This gives the immediate way of how to act in the environment.
- The agent acts in the environment by choosing action that maximizes the Q-function.
- By doing so, the agent finds the maximum expected reward.

Q-Learning

$$Q^*(S,a) = \mathbb{E}_{S'}\left[r(S,a,S^{'}) + \gamma \max_{a^{'}} Q^*(S^{'},a^{'})|S,a\right]$$

- By doing the procedure like the one done for the case of value function: multiplying the learning rate and add the Q-function, lead to Q-learning algorithm.

$$Q^{\pi}_{t+1}(S,a) = Q^{\pi}_t(S,a) + \eta\left[r(S,a,\tilde{S}^{'}) + \gamma \max_{a^{'}} Q^{\pi}_t(\tilde{S}^{'},a^{'})|S,a - Q^{\pi}_t(S,a)\right] \qquad (5)$$



Neuro-Dynamic Programming

- In the following, we consider the Q-function as $Q(S, a, w)$, where w is the parameter.
- Since we want to proximate Q-function, we are no longer dealing with Bellman equation, instead, we are dealing with approximated Bellman equation.
- So, approximated Dynamic Programming.
- Introduce a loss function that shows how far we are from Bellman.

$$L(w_i) = \mathbb{E}_{S,a}\left\{\left(\underbrace{\mathbb{E}_{S'}\left[r(S,a,S^{'}) + \gamma \max_{a^{'}} Q(S^{'}, a^{'}, w_{i-1})\right]}_{A} - \underbrace{Q(S,a,w_i)}_{B}\right)^2\right\}$$

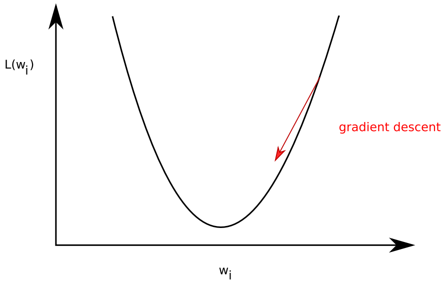
Neuro-Dynamic Programming

- A: The equivalent Q-function one would get, if one applies Bellman updates. A is the target.
- B: B is Neural network Q-function.
- The target has parameters that are not at the same time step level as the neural network time step level.
- The main idea is to use the parameters already produced by neural network.
- The goal is to minimize the loss function.
- The neural network is an Actor.
- The target is the critic.

$$y_i = \mathbb{E}_{S'}\left[r(S, a, S^{'}) + \gamma \max_{a^{'}} Q(S^{'}, a^{'}, w_{i-1}) \right]$$

Neuro-Dynamic Programming

- The agent acts in the environment by maximizing the value of the Q-function, $Q(S, a)$.
- The critic evaluates, if the action taken, was good or bad.
- By following the direction of gradient, the parameter w_i has to be found, which minimizes the error. This is the main role of neural network.



Neuro-Dynamic Programming

$$L(w_i) = \mathbb{E}_{S,a}\left\{\left(\underbrace{\mathbb{E}_{S'}\left[r(S,a,S^{'}) + \gamma \max_{a^{'}} Q(S^{'},a^{'},w_{i-1})\right]}_{A} - \underbrace{Q(S,a,w_i)}_{B}\right)^2\right\}$$

$$\nabla_{w_i} L(w_i) = \mathbb{E}_{S,a,S'}\left\{\left(r(S,a,S^{'}) + \gamma \max_{a^{'}} Q(S^{'},a^{'},w_{i-1}) - Q(S,a,w_i)\right)\nabla_{w_i} Q(S,a,w_i)\right\}$$

- The action is chosen according to:

$$\hat{a} = arg \max_{a} Q(S,a,\hat{w}) \tag{6}$$

Neuro-Dynamic Programming

- The reinforcement algorithm with neural network by using function approximation consists of choosing action according to the policy and then updating parameters w_i.
  - Start in state S.
  - Choose a.
    - By choosing action, it is possible to use a mechanism different from the one used to find the best action.
    - Once the parameter is found, the action is chosen as by equation (6).
    - During the learning procedure, the action can be chosen other way, e.g ϵ-greedy.
  - The environment gives $S^{'}$.
  - Update w by using derivative of loss function.

Thank you for your attention