Deep Reinforcement Learning with Double Q-learning

Friday, February 03, 2017 12:05

Background

- The popular Q-learning algorithm is known to overestimate action values under certain conditions
  - because it includes a maximization step over estimated action values, which tends to prefer overestimated to underestimated values
  - Its irrespective of the source of approximation error (i.e. not because of the neural net or other approximation method)
- This paper answers the following:
  - whether, in practice, such overestimations are common,
  - whether they harm performance,
  - whether they can generally be prevented

- They adapt the Double Q-Learning algorithm to DQN and propose a specific adaptation

- Regular DQN:
  - We want to be able to estimate Q(s,a) - the true value of action a in state s

$$Q_\pi(s,a) \equiv \mathbb{E}\left[R_1 + \gamma R_2 + \ldots \mid S_0 = s, A_0 = a, \pi\right] ,$$

  - Given optimal Q we can easily find the policy
  - The standard Q learning update for the parameters after taking action A_t in state S_t and observing the reward R_{t+1} and resulting state S_{t+1} is given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(Y_t^{\mathrm{Q}} - Q(S_t, A_t; \boldsymbol{\theta}_t))\nabla_{\boldsymbol{\theta}_t} Q(S_t, A_t; \boldsymbol{\theta}_t) . \quad (1)$$

    Where the target is defined as:

$$Y_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t) .$$

  - In DQN paper they introduced 2 interesting ideas:
    - Fixed target - holding another network whose parameters were updated from the online network in a lower frequency. This network was used to calculate the target :

$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-) .$$

    - Experience Replay - observed transitions are stored for some time and sampled uniformly from this memory bank to update the network
      - Not quite clear how this works

Double Q-learning

- Note that the max operator in DQN uses the same values both to select and to evaluate an action
  - We choose the a that maximizes the Q and take this same value of Q to construct the target towards which we tune Q .
  - What if we chose a according to some Q but then use another Q to evaluate Q(s,a) for the a that we found ?
  - Basically it means doing the following
    - In Q learning (still not difference from above, just a clearer notation:)

$$Y_t^{\mathrm{Q}} = R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t) .$$

    - And now apply the separation to get the double Q learning. Note the different set of parameters:

$$Y_t^{\mathrm{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t) .$$

  - This second set of weights can be updated symmetrically by switching the roles of θ and θ'

Double DQN

- We already have a separate network for the target network in DQN. Can we use this target network to evaluate the action and the online network to choose it ?
- i.e. instead of having another network θ' for the double Q, use the target network θ^- that we used in the DQN and get the double DQN :

$$Y_t^{\mathrm{DoubleDQN}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t), \boldsymbol{\theta}_t^-) .$$

- The update of the target network remains the same as in DQN (periodically, once every few frames, copy weights form the online network)