Reinforcement Learning

8. Deep Deterministic Policy Gradient

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud
Reinforcement learning over continuous actions

- In RL, you need a max over actions
- If the action space is continuous, this is a difficult optimization problem
- Policy gradient methods and actor-critic methods mitigate the problem by looking for a local optimum (Pontryagin methods vs Bellman methods)
- In this class, we focus on Actor-Critic methods
Quick history of previous attempts (J. Peters’ and Sutton’s groups)

- Naive Gradient
- Naive Actor-Critic
- Compatibility condition

- Natural Gradient
- Modern Actor-Critic
- Linear architectures: Least square methods
- Linear architectures: Incremental methods

- NAC/eNAC

- INAC

- Those methods proved inefficient for robot RL
- Keys issues: value function estimation based on linear regression is too inaccurate, tuning the stepsize is critical

Deep Deterministic Policy Gradient

- Continuous control with deep reinforcement learning
- Works well on “more than 20” (27-32) domains coded with MuJoCo (Todorov) / TORCS
- End-to-end policies (from pixels to control) or from state variables

DDPG: ancestors

- Most of the actor-critic theory for continuous problem is for stochastic policies (policy gradient theorem, compatible features, etc.)
- DPG: an efficient gradient computation for deterministic policies, with proof of convergence
- Batch norm: inconclusive studies about importance


General architecture

- Actor $\pi_t(a_t | s_t)$, critic $Q(s_t, a_t | \theta)$
- All updates based on SGD
- Adaptive gradient descent techniques tune the step size (RProp, RMSProp, Adagrad, Adam...)
Training the critic

- Same idea as in DQN, but for actor-critic rather than Q-LEARNING
- Minimize the RPE: 
  \[ \delta_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}) | \theta) - Q(s_t, a_t | \theta) \]
- Given a minibatch of \( N \) samples \( \{s_i, a_i, r_i, s_{i+1}\} \) and a target network \( Q' \), compute 
  \[ y_i = r_i + \gamma Q'(s_{i+1}, \pi(s_{i+1}) | \theta') \]
- And update \( \theta \) by minimizing the loss function

\[
L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i | \theta))^2
\]
Training the actor

Deterministic policy gradient theorem: the true policy gradient is
\[ \nabla_{\mu} \pi(s, a) = \mathbb{E}_{\rho(s)}[\nabla_a Q(s, a|\theta) \nabla_{\mu} \pi(s|\mu)] \] (2)

\[ \nabla_a Q(s, a|\theta) \] is used as error signal to update the actor weights.

- Comes from NFQCA
- \[ \nabla_a Q(s, a|\theta) \] is a gradient over actions
- \[ y = f(w.x + b) \] (symmetric roles of weights and inputs)
- Gradient over actions \( \sim \) gradient over weights

Off-policiness

The actor update rule is

\[ \nabla_w \pi(s_i) \approx 1/N \sum_i \nabla_a Q(s, a|\theta)|_{s=s_i, a=\pi(s_i)} \nabla_w \pi(s)|_{s=s_i} \]

The action from the actor is used:
- To compute the target value \( y_i = r_i + \gamma Q'(s_{i+1}, \pi(s_{i+1})|\theta') \)
- To update the actor

As we have seen, actor-critic is off-policy, but convergence is fragile
Parallel updates

- Updating the critic and the actor can be done in parallel
- One may use several actors, several critics...
- Other state-of-the-art methods: Gorila, IMPALA: parallel implementations without replay buffers


Exploration (hot topic)

- Adding to the action an Ornstein-Uhlenbenk (correlated) noise process or Gaussian noise
- Action perturbation (versus param. perturbation, cf. e.g. Plappert or Fortunato, Noisy DQN)
- Several actors explore more


Tuning hyper-parameters

- Influence of target critic update rate ($\tau$)
  - If $\tau = 1$, no target critic from both sides ($< 1$, $> 1$)
  - In CMC, an optimum $\approx 0.05$ is found (non-standard DDPG code)
- Using Huber loss?
  - On some benchmark, the highest $\delta$ is best, thus no Huber loss
  - Unconclusive results, tuning is problem dependent
- Tuning hyper-parameters is difficult, start from the baselines

In bounded param. domains, the gradient may push beyond boundaries
－ Invert the gradient when the parameter goes beyond the bound
－ Better than gradient zeroing or gradient squashing (using tanh function)
－ Efficient on CMC and Half-Cheetah

Over-estimation bias

- Clipping the target critic from the knowledge of $R_{\text{max}}$ helps
- Several ways to act against an overestimation bias
- TD3: Have two critics, always consider the min, to prevent over-estimation
- Less problem knowledge than target critic clipping
- Gives a justification for target actor: slow update of policy is necessary

Replay buffer management

- Different replay buffer management strategies are optimal in different problems

Any question?

Send mail to: Olivier.Sigaud@upmc.fr
Reinforcement Learning

References


