Actor-critic methods

Milica Gašić

Dialogue Systems Group, Cambridge University Engineering Department

In this lecture...

The actor-critic architecture

Least-Squares Policy Iteration

Natural actor-critic

Actor-critic methods

- Actor-critic methods implement *generalised policy iteration* - alternating between a policy evaluation and a policy improvement step.
- There are two closely related processes of

  actor improvement which aims at improving the current policy

  critic evaluation which evaluates the current policy

  If the critic is modelled by a bootstrapping method it reduces the variance so the learning is more stable than pure policy gradient methods.

Relation to other RL methods

Value-based methods:

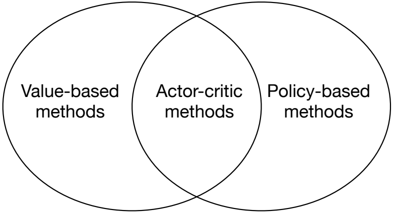
- estimate the value function
- policy is implicit (eg ϵ-greedy)

Policy-based methods

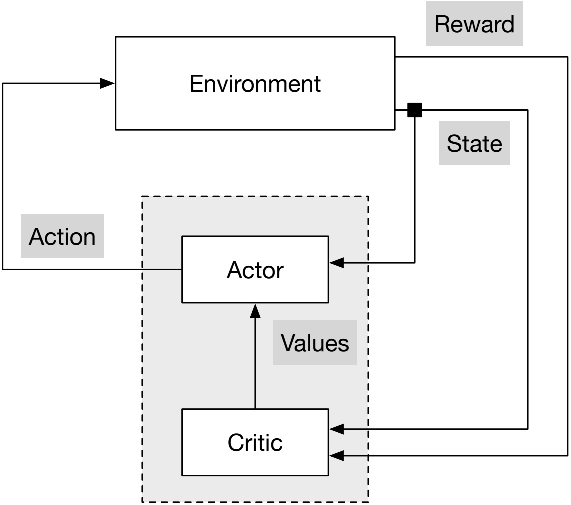
- estimate the policy
- no value function

Actor-critic methods

- estimate the policy
- estimate the value function



Actor-critic architecture



Behaviour vs target policy for actor-critic methods

- The policy used to generate the samples (*behaviour policy*) could be different from the one which is evaluated and improved (*target policy*).
- This allows the critic to learn about the actions not preferred by the target policy and therefore improve the target policy.
- This is impossible to achieve if behaviour policy is the same as target policy and if they are both deterministic.
- In the case that the behaviour and the target policy are the same but stochastic the estimation on low-probability states might be poor.
- If behaviour policy is completely random it might not visit important parts of the space.
- The best choice of the behaviour policy is to add exploration into the target policy.

Implementing actor-critic architecture

Small state-action space The critic is a Q-function estimator and the actor is ϵ-greedy or Boltzmann policy estimated in a tabular way.

Large state-action spaces Both the critic and the actor use function approximation

Implementing a critic

- The critic estimates the value of the current policy – *prediction problem*
- Since the actor uses Q-values to choose actions, the critic must estimate the Q-function
- For small state-spaces we could use tabular TD algorithms to estimate the Q-function (SARSA, Q-learning, etc)
- For large state-spaces we could use LSTD to estimate the Q-function.

Implementing an actor

Policy improvement can be implemented in two ways:

greedy improvement Moving the policy towards the greedy policy underlying the Q-function estimate obtained from the critic

policy gradient Perform policy gradient directly on the performance surface underlying the chosen parametric policy class

Greedy improvement

- For small state-action spaces the policy is greedy with respect to the obtained Q-value
- For large state-action spaces the policy is parametrised and the greedy action is computed on the fly

Least-Squares Policy Iteration

Algorithm 1 Least-Squares Policy Iteration

```
Input: parametrisation of Q(·,·;θ) = θ^T φ(·,·)
Initialise θ arbitrarily
repeat
    π(s) = arg max_a θ^T φ(s,a) {policy improvement}
    θ = LSTD(π, φ, θ) {policy evaluation}
until convergence
```

Policy gradient

- Policy gradient methods perform stochastic gradient descent on the performance surface of the parametrised policy.
- Policy gradient theorem (last lecture) gives

$$\begin{aligned}
\nabla J(\boldsymbol{\omega}) &= E_\pi\left[\gamma^t R_t \nabla_{\boldsymbol{\omega}} \log \pi(a|s,\boldsymbol{\omega})\right] && (1)\\
&= E_\pi\left[\gamma^t Q_\pi(s,a) \nabla_{\boldsymbol{\omega}} \log \pi(a|s,\boldsymbol{\omega})\right] && (2)\\
&= E_\pi\left[\gamma^t \left(Q_\pi(s,a) - V_\pi(s)\right) \nabla_{\boldsymbol{\omega}} \log \pi(a|s,\boldsymbol{\omega})\right] && (3)
\end{aligned}$$

  PROOF
- **Advantage function** $A_\pi(s,a)$ is defined as

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

Compatible function approximation

actor policy that takes actions parametrised with $\boldsymbol{\omega}$, for example

$$\pi(a|s,\boldsymbol{\omega}) = \frac{\exp(\boldsymbol{\omega}^\mathsf{T}\boldsymbol{\psi}(s,a))}{\sum_{a'}\exp(\boldsymbol{\omega}^\mathsf{T}\boldsymbol{\psi}(s,a'))}$$

critic Advantage function that evaluates the actor parametrised with $\boldsymbol{\theta}$

$$\gamma^t A(s_t,a,\boldsymbol{\theta}) = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\phi}(s_t,a)$$

such that the choice of the approximation is *compatible* with the policy parametrisation: if $\boldsymbol{\omega}$ changes $\boldsymbol{\theta}$ changes too.

Limitations of vanilla policy gradient

- Vanilla policy gradient methods are not always stable as (large) changes in the parameters can result in unexpected policy moves.
- Convergence can be very slow.

(More in next lecture.)

Natural actor-critic [Peters and Schaal, 2008]

- Uses compatible function approximation for actor and critic
- A modified form of gradient – *natural gradient* is used to find the optimal parameters

Natural Policy Gradient

- Advantage function is parametrised with parameters $\boldsymbol{\theta}$ such that the direction of change is the same as for the policy parameters $\boldsymbol{\omega}$

$$\gamma^t \nabla_{\boldsymbol{\theta}} A(s_t, a, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\omega}} \log \pi(s_t, a, \boldsymbol{\omega})$$

- Then by replacing

$$\gamma^t A(s_t, a, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\omega}} \log \pi(s_t, a, \boldsymbol{\omega})^{\mathsf{T}} \boldsymbol{\theta}$$

  in Eq 3
- It can be shown

$$\boldsymbol{\theta} = G_{\boldsymbol{\omega}}^{-1} \nabla_{\boldsymbol{\omega}} J(\boldsymbol{\omega})$$

  where $G_{\boldsymbol{\omega}}$ is the Fisher information matrix

$$G_{\boldsymbol{\omega}} = E_{\pi(\boldsymbol{\omega})} \left[\nabla \log \pi(\mathbf{b}, a, \boldsymbol{\omega}) \nabla \log \pi(\mathbf{b}, a, \boldsymbol{\omega})^{\mathsf{T}} \right]$$

- $\boldsymbol{\theta}$ is the natural gradient of $J(\boldsymbol{\omega})$

Natural gradient [Amari, 1998]

- Distance in Riemann space: $|d\omega|^2 = d\omega^\mathsf{T} G_\omega d\omega$, where G_ω is a metric tensor
- Direction of steepest descent in Riemann space for some loss function $L(\omega)$ is $G_\omega^{-1} \nabla_\omega L(\omega)$
- If ω is used to optimise the estimate of a probability distribution $p(x|\omega)$ then the optimal metric tensor is Fisher information matrix as this give distances invariant to scaling of the parameters.

$$G_\omega = E(\nabla \log p(x|\omega) \nabla \log p(x|\omega)^\mathsf{T})$$

- It can be shown that $KL(p(x|\omega)||p(x|\omega + d\omega)) \approx d\omega^\mathsf{T} G_\omega d\omega$

Episodic Natural Actor Critic

Algorithm 2 Episodic Natural Actor Critic

1: Input: parametrisation of $\pi(\boldsymbol{\omega})$
2: Input: parametrisation of $\gamma^t A(\boldsymbol{\theta}) = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\phi}$
3: Input: step size $\alpha > 0$
4: Initialise $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$
5: **repeat**
6: Execute the episode according to the current policy $\pi(\boldsymbol{\omega})$
7: Obtain sequence of states s_t, actions a_t and return R
8: **Critic evaluation** Choose $\boldsymbol{\theta}$ and J to minimise $(\sum_t \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\phi}(s_t, a_t) + J - R)^2$
9: **Actor update** $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \alpha\boldsymbol{\theta}$
10: **until** convergence

In practice the update is not performed after every episode but rather after a number of episodes to improve stability and efficiency.

Summary

- Actor-critic methods implement generalised policy iteration where the actor aims at improving the current policy and the critic evaluates the current policy.
- For large state-action spaces, both the actor and the critic are parametrised functions.
- The actor and the critic can be estimated using compatible function approximation, where their parameters depend on each other and are estimated using stochastic gradient descent.
- Instead of the vanilla gradient which has low convergence rates, the natural gradient can be used and this yields natural actor-critic algorithm.

Next lecture

- Deep reinforcement learning
- To prepare for the next lecture please read
  - *Mastering the game of Go with deep neural networks and tree search*, http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html
  - *Mastering the game of Go without human knowledge*, https://www.nature.com/articles/nature24270

References I

Amari, S.-I. (1998).
Natural gradient works efficiently in learning.
Neural Comput., 10(2):251–276.

Peters, J. and Schaal, S. (2008).
Natural actor-critic.
Neurocomputing, 71(7):1180–1190.