RL Reading Group – Prep Session 2

Zhiyu Zhang

09/23/2019

Boston University Division of System Engineering

BOSTON UNIVERSITY

Overview

- Recap of last session
- Policy optimization and its benefit
- Vanilla algorithms
  - Vanilla policy gradient
  - Vanilla actor-critic
- The drawbacks of policy optimization
- Natural policy gradient
- Trust-region policy optimization (TRPO)

Recap of last session

- Problem of interest: optimal control of finite MDPs
- Two “good” objective functions we can maximize:
  - Discounted infinite horizon cumulative reward; no restriction on policies
  - Undiscounted infinite horizon cumulative reward; policies are proper
- Classical solution techniques: DP
  - Important structure: monotonicity and contraction of DP operators
  - Result:
    - Bellman equation has a unique solution that can be asymptotically reached by iteratively applying Bellman operator (value iteration)
    - Policy iteration (PI) generalizes value iteration, with the same guarantee

Recap of last session

- Drawbacks of DP
  - Curse of dimensionality
  - Requires stochastic model in the explicit form

- If we have large state & action spaces: approximate DP
- PLUS, if we only have simulation / real experiences: RL
  - DP only “exploits”; RL needs to trade off exploration & exploitation
    - Value function becomes action-value function
    - Deterministic policies become stochastic policies
  - Basic idea of RL algorithms: “Generalized PI”, iterates between policy evaluation and policy improvement

Recap of last session

- On-policy vs. Off-policy algorithms
  - On-policy: the policy that samples the training data is the same as the policy to be evaluated / improved
  - Off-policy: the opposite; generalizes the on-policy algorithms

- Value function based RL algorithms
  - Monte Carlo: use the idea of MC integration to estimate action value functions
  - Temporal Difference
    - MC with bootstrapping: use the old estimate to generate new ones; Ex. SARSA
  - Off-policy generalization:
    - One step off-policy: Q learning; Otherwise, importance sampling

Recap of last session

- From tabular cases to function approximation
  - Almost everything tabular has good infinite sample guarantee (converges to optimum)
  - Finite sample performance is not well understood (regret)
  - Value function methods with function approximation is problematic
    - The “deadly triad”: function approximation, off-policy, bootstrapping
    - How to choose loss function for the function approximation (Ex. DNN)? No labels!
    - Result: need to have something to “stabilize” these algorithm
      - (Policy optimization; Actor-critic)

The difficulty of RL

- A naive, personal view:
  - The objective is not a classical statistical quantity; hard to convert into a stat problem (will see)
  - It's an active learning problem
    - We (through a black box) choose the training data
    - Therefore, training data is not IID
  - It's also an online learning problem: we often care about online performance
  - See Sec 4.2 of [1]

[1] Szepesvári, Csaba. "Algorithms for reinforcement learning." Synthesis lectures on artificial intelligence and machine learning 4, no. 1 (2010): 1-103.

Policy optimization

- Our task is to generate a stationary (stochastic) policy π for the agent

$$a_t \sim \pi(\cdot \mid s_t)$$

- Consider the value function approach:
  - In DP, only exploitation, the policy can simply be greedy w.r.t. the value function
  - In RL, we only have an inaccurate estimate of the value function; the policy is often "approximately" greedy
    - ϵ greedy, Boltzmann exploration, UCB type exploration, ...

- Tuning these exploration strategies is not trivial
- They are hard to "converge" to stochastic policies
  - In adversarial settings like games, the optimal strategy is often stochastic

Policy optimization

- Consider policy optimization: directly parameterizing the policy itself
- With finite action space A, a popular choice is the softmax parameterization

$$\pi_\theta(a|s) = \frac{\exp(\phi(s,a,\theta))}{\sum_{a'} \exp(\phi(s,a',\theta))}$$

  $\phi(s,a,\theta)$ is the output of a function parameterized by θ (Ex. DNN)
- This is very expressive: can approach a greedy policy or remain siginificantly random

- If action space is continuous, the policy take the form of a Gaussian mixture model, with means and covariances represented by $\phi(s,a,\theta)$
- We focus, as last time, on finite MDPs

On-policy (state-action) distribution $d_\pi(s,a)$

- This is a bad name: it actually has nothing to do with the "on-policy" concept for RL algorithms
- It depends on the objective function of the problem:
  - For maximizing infinite horizon average reward, $d_\pi(s)$ with a marginalized is the stationary distribution of the MC induced by π (with technical conditions)
  - As we said, average reward problems are harder to deal with, so we consider two easier problems

- For maximizing cumulative γ-discounted reward (the "continuing task"), it is defined as [1]

$$d_\pi(s,a) = (1-\gamma)\pi(a|s)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi, \rho_0)$$

Such a limit always exists; as $\gamma \to 1$, the marginal $d_\pi(s)$ "approaches" the stationary distribution of the MC induced by π

[1] Kakade, Sham, and John Langford. "Approximately optimal approximate reinforcement learning." In ICML, vol. 2, pp. 267-274. 2002.

On-policy (state-action) distribution $d_\pi(s,a)$

- It also works similarly as stationary distribution: [1]

  Remember that $R(s,a,s')$ is the reward function; $R(s,a)$ can be seen as a random variable bounded by R

$$V_\pi = E_{\tau|\pi,\rho_0}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1})\right] = \frac{1}{1-\gamma}E_{(s,a)\sim d_\pi}[R(s,a)]$$

- As for maximizing the undiscounted cumulative reward among proper policies (the episodic task), $d_\pi(s,a)$ is defined in another way (see Page 199, [2] for a rough definition); the idea is the same

- Consider the difference of expected value V_π and $V_{\pi'}$, normally it depends on two part: the change in policy $\pi' - \pi$ and the change in (marginal) on-policy (state) distribution $d_{\pi'}(\cdot) - d_\pi(\cdot)$; however, the derivative $\nabla_\theta V_{\pi_\theta}$ only depends on $\nabla_\theta \pi_\theta$ and not on $\nabla_\theta d_{\pi_\theta}$

[1] Kakade, Sham, and John Langford. "Approximately optimal approximate reinforcement learning." In ICML, vol. 2, pp. 267-274. 2002.
[2] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Policy gradient theorem

- Policy gradient theorem for continuing tasks:

$$\nabla_\theta V_{\pi_\theta} = \sum_{s \in S} d_{\pi_\theta}(s) \sum_{a \in A} Q_{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a|s)$$

- The proof is simple but doesn't fit in one slide, see Page 325, [1]; Need to modify a bit to incorporate discount factor, since it totally ignores the existence of limit issue...
- For episodic case it is the same, with $d_{\pi_\theta}(s)$ defined in another way

- The idea to use the gradient: similar to SGD, write it as the expectation over sample path and use the rollout sample paths to do sample-based gradient ascent

$$\nabla_\theta V_{\pi_\theta} = \sum_{s \in S} d_{\pi_\theta}(s) \sum_{a \in A} \pi_\theta(a|s) Q_{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} = E_{(s,a)\sim d_\pi}\left[Q_{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)\right]$$

[1] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Vanilla policy gradient for episodic tasks

- Consider the episodic tasks, as the gradient estimate is unbiased. Continuing tasks are similar, but updates will be biased.

- In each epoch, rollout π_θ for sevaral episodes, store state-action samples and the cumulative reward of each sample path in the buffer, until the buffer is full
- Between epochs, for each step in the buffer,
  - Calculate an estimate $\hat{Q}_{\pi_\theta}(s, a)$ using Monte Carlo estimation
  - Calculate the gradient estimate $\nabla_\theta V_{\pi_\theta} = \hat{Q}_{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a|s)$
  - Gradient ascent one step, with a some learning rate

- We follow the procedure in [1], but [2] also gives an easy to follow derivation with code

[1] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.
[2] https://spinningup.openai.com/en/latest/

Baseline and Vanilla actor-critic

- Consider the policy gradient form

$$\nabla_\theta V_{\pi_\theta} = \sum_{s \in S} d_{\pi_\theta}(s) \sum_{a \in A} Q_{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a|s)$$

- Notice that if replace $Q_{\pi_\theta}(s,a)$ with a function of s only, the RHS is 0; Ex. $V_{\pi_\theta}(s)$
- The policy gradient can be written as

$$\nabla_\theta V_{\pi_\theta} = E_{(s,a)\sim d_\pi}\left[(Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s))\nabla_\theta \log \pi_\theta(a|s)\right]$$

- It doesn't change the expectation, but can dramatically decrease variance
- Define advantage function as

$$A_{\pi_\theta}(s,a) = Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s)$$

[1] provides an $TD(\lambda)$ type algorithm to estimate $A_{\pi_\theta}(s,a)$ efficiently

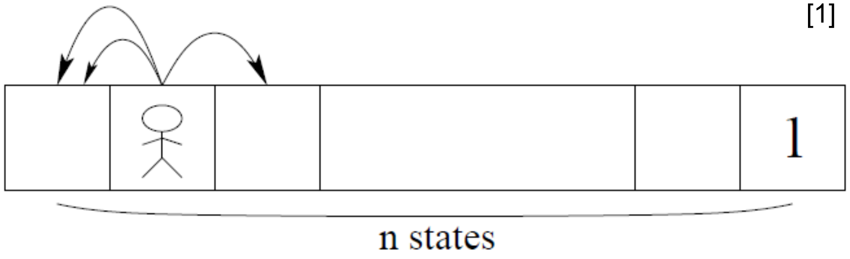
- If $Q_{\pi_\theta}(s,a)$ and $V_{\pi_\theta}(s)$ are learned using TD methods with function approximation, we call the algorithm actor-critic: actor for parameterized policy, critic for parameterized value function

[1] Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-dimensional continuous control using generalized advantage estimation." arXiv preprint arXiv:1506.02438 (2015).

The drawbacks of policy gradient

- Policy optimization methods are intrisically on-policy, which means when starting with a bad policy, it is hard to improve

[1]



- Example
  - Three actions; equal prob.
  - The first passage time from the left most state to the right most state is $3(2^n - n - 1)$
  - Should rollout an extremely long sample path before policy improvement
- One possible solution
  - Exploration start: make ρ_0 more uniform
- Still largely unsolved; Sample complexity is a lot worse than value function based methods
  - For simple benchmark tasks like Atari games, a few million samples is considered “efficient”

[1] Kakade, Sham, and John Langford. "Approximately optimal approximate reinforcement learning." In ICML, vol. 2, pp. 267-274. 2002.

Natural policy gradient

- The vanilla policy gradient looks good, but has the following two early noticed problems:
  - Conceptually, the gradient is non-covariant, which doesn't make sense
    An affine change of coordinates leads to a different effective gradient direction
  - Practically, training is often slow
    The gradient direction is the steepest w.r.t. L_2 norm, but not necessarily the correct "natural" norm we care about

- From the optimization point of view, this is the same motivation that generalizes gradient ascent to Newton's method: it is covariant, and steepest w.r.t. the correct (Hessian) norm

- Similarly, we want to find a matrix to weight the gradient direction to make it covariant and (hopefully) faster

[1] Kakade, Sham M. "A natural policy gradient." In Advances in neural information processing systems, pp. 1531-1538. 2002.
[2] Bagnell, J. Andrew, and Jeff Schneider. "Covariant policy search." (2003).

Natural policy gradient

- From now on, I keep the discussion very vague as I don't understand it well... The paper itself is also kinda "high-level"...
- The discussion relies on another (very interesting) view of policy optimization:

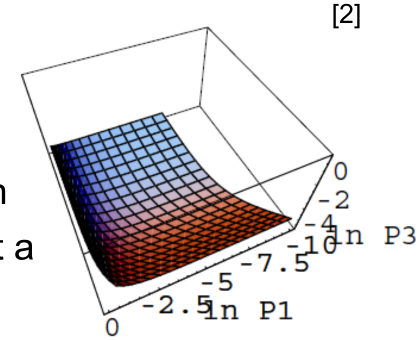
  Our selection of policy parameter θ induces a distribution of sample path $p(\tau|\theta)$; the objective funciton is essentially the form of an expectation over $p(\tau|\theta)$

  $$V_{\pi_\theta} = E_{\tau \sim p(\cdot|\theta)}[Total\_R(\tau)]$$

  regardless of the specific objective we choose (discounted / undiscounted / average)

  By changing θ, we move the distribution $p(\tau|\theta)$ on the "distribution manifold"

[2]



  Example: we have three possible sample paths; the distribution of sample paths can be parameterized by two parameters, therefore the the distribution manifold is "dim 2" embedded in $\mathbb{R}^3$ (axes showing log probability; this is just a log coordinate transform of the probability simplex in $\mathbb{R}^3$)

[1] Kakade, Sham M. "A natural policy gradient." In Advances in neural information processing systems, pp. 1531-1538. 2002.
[2] Bagnell, J. Andrew, and Jeff Schneider. "Covariant policy search." (2003).

Natural policy gradient

- If $\theta \in \mathbb{R}^n$, then the distribution manifold is "dim n" inside the space with dimension as the size of all possible sample paths. (A "dim n" subset of the high dimensional probability simplex)
- We want to establish Riemannian structure on this manifold. "Equivalently", we want to define a Matrix-weighted inner product on its tangent space: $< u, v > = u^T G v$.
- If we have an G, then the steepest ascent direction can be obtained from a duality argument [2]

$$\delta\theta \propto G^{-1} \nabla_\theta V_{\pi_\theta}$$

- How to choose G? There are plenty of possible choices that make the gradient covariant
- In this algorithm, they propose the Fisher information matrix

$$G_{ij} = E_{p(\tau|\theta)}\left[\frac{\partial \log p(\tau|\theta)}{\partial \theta_i} \frac{\partial \log p(\tau|\theta)}{\partial \theta_j}\right]$$

[1] Kakade, Sham M. "A natural policy gradient." In Advances in neural information processing systems, pp. 1531-1538. 2002.
[2] Bagnell, J. Andrew, and Jeff Schneider. "Covariant policy search." (2003).

Natural policy gradient

- Motivation of this choice:
  - [1, 3] Chentsov (or Cencov) characterization theorem: “the Fisher information metric is the only metric that is invariant under a family of probablistically meaningful mappings termed congruent embeddings by a Markov morphism” (???)
  - [3] Fisher information is the Hessian of KL divergence; and KL divergence is the objective of another optimization problem (MLE) on the same distribution manifold

    The MLE problem is defined as: we have $\{\tau_i\}_{i=1}^{N} \sim p(\tau|\theta^*)$, IID, we want to maximize

$$l(\theta) = \sum_{i=1}^{N} \log p(\tau_i|\theta)$$

    It is the same as minimizing $D_{KL}(p_{data}(\cdot)||p(\cdot\,|\theta))$ as $N \to \infty$

    The MLE problem is useful to us in the sense that we want the on-policy distribution of our new policy to be close to that of the old one (not so large information loss)

[1] Lebanon, Guy. "Axiomatic geometry of conditional models." IEEE Transactions on Information Theory 51, no. 4 (2005): 1283-1294.
[2] Kakade, Sham M. "A natural policy gradient." In Advances in neural information processing systems, pp. 1531-1538. 2002.
[3] Bagnell, J. Andrew, and Jeff Schneider. "Covariant policy search." (2003).
[4] Amari, Shun-Ichi. "Natural gradient works efficiently in learning." Neural computation 10, no. 2 (1998): 251-276.
[5] Peters, Jan, Katharina Mulling, and Yasemin Altun. "Relative entropy policy search." In Twenty-Fourth AAAI Conference on Artificial Intelligence. 2010.

Natural policy gradient

- [2, 4] Consider the MLE problem defined just now

  Batch MLE has very good statistical property: it is consistent and asymptotically efficient (variance converges to the Cramer-Rao lower bound)

  Consider the online optimization problem: each time we are give one sample path τ_i, we use it to make a gradient step and discard it (SGD)

  (Thm 2, [4]) It is shown that online MLE using steepest ascent weighted by Fisher information matrix and 1/n stepsize is asympotically efficient (asymptotically achieves the same performance as batch MLE)

  The argument from this: Fisher information is a very good Riemannian metric for another useful optimization problem (MLE) on the same distribution manifold, therefore it incorporates some geometric structure that can be useful in policy optimization

[1] Lebanon, Guy. "Axiomatic geometry of conditional models." IEEE Transactions on Information Theory 51, no. 4 (2005): 1283-1294.
[2] Kakade, Sham M. "A natural policy gradient." In Advances in neural information processing systems, pp. 1531-1538. 2002.
[3] Bagnell, J. Andrew, and Jeff Schneider. "Covariant policy search." (2003).
[4] Amari, Shun-Ichi. "Natural gradient works efficiently in learning." Neural computation 10, no. 2 (1998): 251-276.
[5] Peters, Jan, Katharina Mulling, and Yasemin Altun. "Relative entropy policy search." In Twenty-Fourth AAAI Conference on Artificial Intelligence. 2010.

Natural policy gradient

- [1] as the first paper on natural policy gradient shows the idea, but the formulation has problem..
  - Possibly by luck, the authors used the correct form in the empirical evaluation part, and the performance is very strong
- [2] is very readable and corrects the problem in [1]'s formulation
- The resulting Fisher information matrix gives a truly covariant gradient, but its definition doesn't have reward function in it; therefore it is for sure not optimal!

- The above discussion highlights one difficulty in RL: the objective function is hard to be incorporated into a statistical framework
  - One recent idea in RL is following this line: maximum entropy RL, or soft actor-critic
  - They use a heuristic method to cast the value function into a statistical quantity; then policy optimization becomes inference in a graphical model; empirical performance is strong [3]

[1] Kakade, Sham M. "A natural policy gradient." In Advances in neural information processing systems, pp. 1531-1538. 2002.
[2] Bagnell, J. Andrew, and Jeff Schneider. "Covariant policy search." (2003).
[3] Levine, Sergey. "Reinforcement learning and control as probabilistic inference: Tutorial and review." arXiv preprint arXiv:1805.00909 (2018).

How to choose stepsizes?

- This is when the MLE problem becomes useful

  Large stepsizes means that we are not “protected” by policy gradient theorem: the change in value function is determined by not only the change of policy, but the change of on-policy distribution

- A natural idea is: we can pick the largest stepsize that still keeps the change of on-policy distribution reasonable (relative entropy policy search)
- This leads to the following constrained optimization problem

$$\max_{\Delta\theta} \; E_{(s,a)\sim d_{\pi_\theta}(s)\pi_{\theta+\Delta\theta}(a|s)}[R(s,a)]$$
$$\text{s.t} \quad E_{s\sim d_{\pi_\theta}(s)}[D_{KL}(\pi_\theta||\pi_{\theta+\Delta\theta})] \leq \epsilon$$

  where $d_{\pi_\theta}(s)$, the on-policy distribution of the old policy, is used as an approximation of $d_{\pi_\theta+\Delta\theta}(s)$, the on-policy distribution of the new policy. In practice, it is replaced by the empirical distribution; it would be beneficial to use another “distance” here, like Wasserstein

[1] Peters, Jan, Katharina Mulling, and Yasemin Altun. "Relative entropy policy search." In Twenty-Fourth AAAI Conference on Artificial Intelligence. 2010.

Trust Region Policy Optimization

- Take a first order approximation of the objective function, take a second order approximation of the constraint; the problem becomes

$$\max_{\Delta\theta} \quad \Delta\theta^T \nabla_\theta V_{\pi_\theta}$$
$$\text{s.t} \quad \frac{1}{2}\Delta\theta^T G \Delta\theta \leq \epsilon$$

where $\nabla_\theta V_{\pi_\theta}$ is the policy gradient, G is the Fisher information matrix

- The analytical solution is

$$\Delta\theta = \sqrt{\frac{2\epsilon}{\left(\nabla_\theta V_{\pi_\theta}\right)^T G^{-1} \nabla_\theta V_{\pi_\theta}}} G^{-1} \nabla_\theta V_{\pi_\theta}$$

- This is exactly the natural policy gradient direction, with a certain stepsize
- In practice it is solved by conjugate gradient, since we don't want to invert G directly

[1] Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization." In International conference on machine learning, pp. 1889-1897. 2015.
[2] https://spinningup.openai.com/en/latest/algorithms/trpo.html