Continuous Deep Q-Learning with Model-based Acceleration

presented by Jason TOKO

背景与动机

- 传统Q-learning应用于连续动作空间时，每一步求解复杂的非线性函数最大值会相当麻烦。
  - 常用AC方法来解决连续动作空间的问题（如：DDPG），但AC方法需要构建并训练Actor和Critic两个网络。
- 对于model-free的RL算法，其采样复杂度会随FA的维度变高而变高。
  - 采用task-specific的特征表示方式可提高采样效率，但限制了学习的范围且需要大量的先验知识。
  - 采用model-based的RL方法也可提高效率，但是策略学习受到模型限制。
  - 大部分现实任务中，学习一个好策略比学习一个好模型要简单。

背景与动机

- 由此，文中提出了两个完备的算法：

- 连续的Q-learning算法——Normalized Advantage Function(NAF)
- 基于imagination rollouts的NAF算法

NAF算法

- Normalized Advantage Function
  - Q函数：

    $$Q(\boldsymbol{x}, \boldsymbol{u}|\theta^Q) = A(\boldsymbol{x}, \boldsymbol{u}|\theta^A) + V(\boldsymbol{x}|\theta^V)$$

  - A函数：

    $$A(\boldsymbol{x}, \boldsymbol{u}|\theta^A) = -\frac{1}{2}(\boldsymbol{u} - \boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))^T \boldsymbol{P}(\boldsymbol{x}|\theta^P)(\boldsymbol{u} - \boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))$$

  - 其中P为正定矩阵，即有$x^T P x > 0$，P矩阵可被分解为：

    $$\boldsymbol{P}(\boldsymbol{x}|\theta^P) \quad = \quad \boldsymbol{L}(\boldsymbol{x}|\theta^P)\boldsymbol{L}(\boldsymbol{x}|\theta^P)^T$$

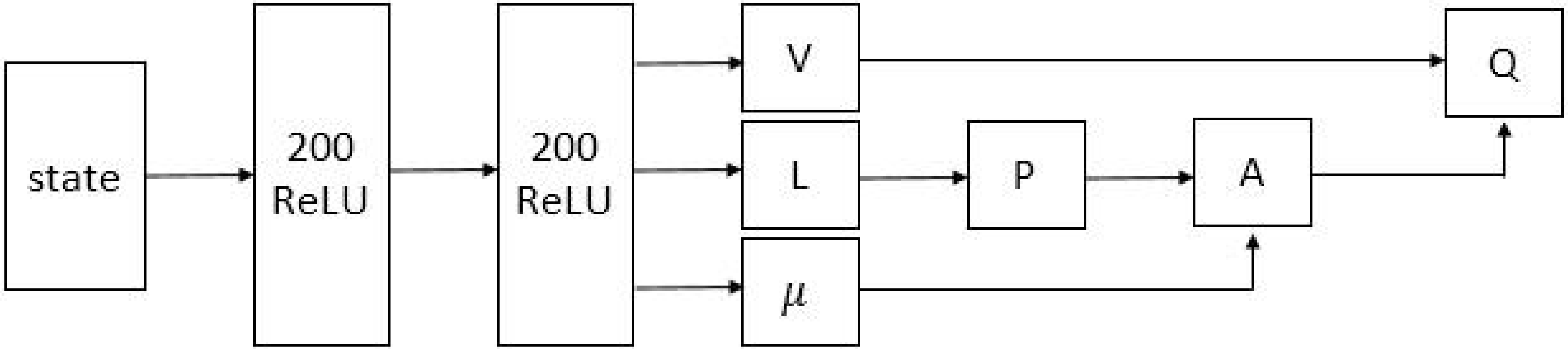
    - L是对角线为正数下三角矩阵（实际中对角线进行了exp指数化）

NAF算法

Algorithm 1 Continuous Q-Learning with NAF

Randomly initialize normalized Q network $Q(\boldsymbol{x}, \boldsymbol{u}|\theta^Q)$.
Initialize target network Q' with weight $\theta^{Q'} \leftarrow \theta^Q$.
Initialize replay buffer $R \leftarrow \emptyset$.
for episode=1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $\boldsymbol{x}_1 \sim p(\boldsymbol{x}_1)$
  for t=1, T **do**
    Select action $\boldsymbol{u}_t = \mu(\boldsymbol{x}_t|\theta^\mu) + \mathcal{N}_t$
    Execute $\boldsymbol{u}_t$ and observe r_t and $\boldsymbol{x}_{t+1}$
    Store transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1})$ in R
    for iteration=1, I **do**
      Sample a random minibatch of m transitions from R
      Set $y_i = r_i + \gamma V'(\boldsymbol{x}_{i+1}|\theta^{Q'})$
      Update θ^Q by minimizing the loss: $L = \frac{1}{N}\sum_i(y_i - Q(\boldsymbol{x}_i, \boldsymbol{u}_i|\theta^Q))^2$
      Update the target network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
    end for
  end for
end for

NAF算法网络结构



NAF算法

- NAF算法特点：
  - 简化了Q-learning计算最大值的复杂操作，但同时网络结构也变得复杂。
  - 实现了连续动作空间的控制算法
  - 相比于AC方法，只需一个网络，算法更简单，采样效率更高。

Imagination Rollouts

- 实验表明，即使在正确模型下，只给予算法“good”的动作对算法的改良还是微乎其微的，算法也需要体验“bad”的动作。（略）
- 在现实任务中，“bad”动作会增加大量数据，且会导致一定的危险。
- Imagination Rollouts实际上是在环境的模型上进行rollout，并将rollout轨迹上的所有样本放入replay buffer中。
- 常用的model-based方法有：iLQG、Dyna-Q

基于Imagination Rollouts的NAF算法

Algorithm 2 Imagination Rollouts with Fitted Dynamics and Optional iLQG Exploration

Randomly initialize normalized Q network $Q(\boldsymbol{x}, \boldsymbol{u}|\theta^Q)$.
Initialize target network Q' with weight $\theta^{Q'} \leftarrow \theta^Q$.
Initialize replay buffer $R \leftarrow \emptyset$ and fictional buffer $R_f \leftarrow \emptyset$.
Initialize additional buffers $B \leftarrow \emptyset$, $B_{old} \leftarrow \emptyset$ with size nT.
Initialize fitted dynamics model $\mathcal{M} \leftarrow \emptyset$.
for $episode = 1, M$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $\boldsymbol{x}_1$
  Select $\mu'(\boldsymbol{x}, t)$ from $\{\mu(\boldsymbol{x}|\theta^\mu), \pi_t^{iLQG}(\boldsymbol{u}_t|\boldsymbol{x}_t)\}$ with probabilities $\{p, 1-p\}$
  for $t = 1, T$ **do**
    Select action $\boldsymbol{u}_t = \mu'(\boldsymbol{x}_t, t) + \mathcal{N}_t$
    Execute $\boldsymbol{u}_t$ and observe r_t and $\boldsymbol{x}_{t+1}$
    Store transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1}, t)$ in R and B
    if $\mod(episode \cdot T + t, m) = 0$ and $\mathcal{M} \neq \emptyset$ **then**
      Sample m $(\boldsymbol{x}_i, \boldsymbol{u}_i, r_i, \boldsymbol{x}_{i+1}, i)$ from B_{old}
      Use $\mathcal{M}$ to simulate l steps from each sample
      Store all fictional transitions in R_f
    end if
    Sample a random minibatch of m transitions $I \cdot l$ times from R_f and I times from R, and update $\theta^Q, \theta^{Q'}$ as in Algorithm 1 per minibatch.
  end for
  if B_f is full **then**
    $\mathcal{M} \leftarrow$ FitLocalLinearDynamics(B_f) (see Section 5.3)
    $\pi^{iLQG} \leftarrow$ iLQG_OneStep($B_f, \mathcal{M}$) (see appendix)
    $B_{old} \leftarrow B_f$, $B_f \leftarrow \emptyset$
  end if
end for

基于Imagination Rollouts的NAF算法

- 算法细节：
  - 算法前期，imagination rollouts在Q函数较差时较为有用，后期作用相对变小，因此算法在一定的迭代次数后将放弃imagination rollouts。
  - 拟合动态模型时，只需要在当前样本集中学习局部模型即可，而不必学习环境的全局模型。
  - 模型表示为 $p_t(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\mathbf{F}_t[\boldsymbol{x}_t; \boldsymbol{u}_t] + \mathbf{f}_t, \mathbf{N}_t)$，每隔n步从replay buffer中重新拟合模型的高斯分布。
- 算法特点：
  - 结合了model-free的泛化性和model-based的高采样效率，加快了RL算法的学习。

总结

- 提出NAF算法，将Q-learning算法拓展到连续动作空间
- 通过实验表明了model-based方法的缺陷
- 提出一种结合imagination rollouts的model-based方法，加速了model-free RL算法的学习