强化学习及其应用

Reinforcement Learning and Its Applications

第五章 值函数逼近

Value Function Approximation

授课人：周晓飞
zhouxiaofei@iie.ac.cn
2023-6-14

课件放映 PDF-〉视图-〉全屏模式

第五章 值函数逼近

5.1 值函数逼近

5.2 增量预测与控制

5.3 批量预测与控制

第五章 值函数逼近

5.1 值函数逼近

5.2 增量预测与控制

5.3 批量预测与控制

值函数逼近

问题描述

- **Large-Scale Reinforcement Learning**

  Prediction 和 Control 问题，都需要值函数估计，S --> V(S)， (S, A) --> Q(S, A)

  Large MDPs 状态集大，甚至是连续状态空间。

  问题：

  （1） 存储需求大，每个 S 的值都要估计，计算代价大。

  （2） 观测的 episodes 有限，只能学习部分 S 的值。

- **解决办法：值函数逼近**

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$

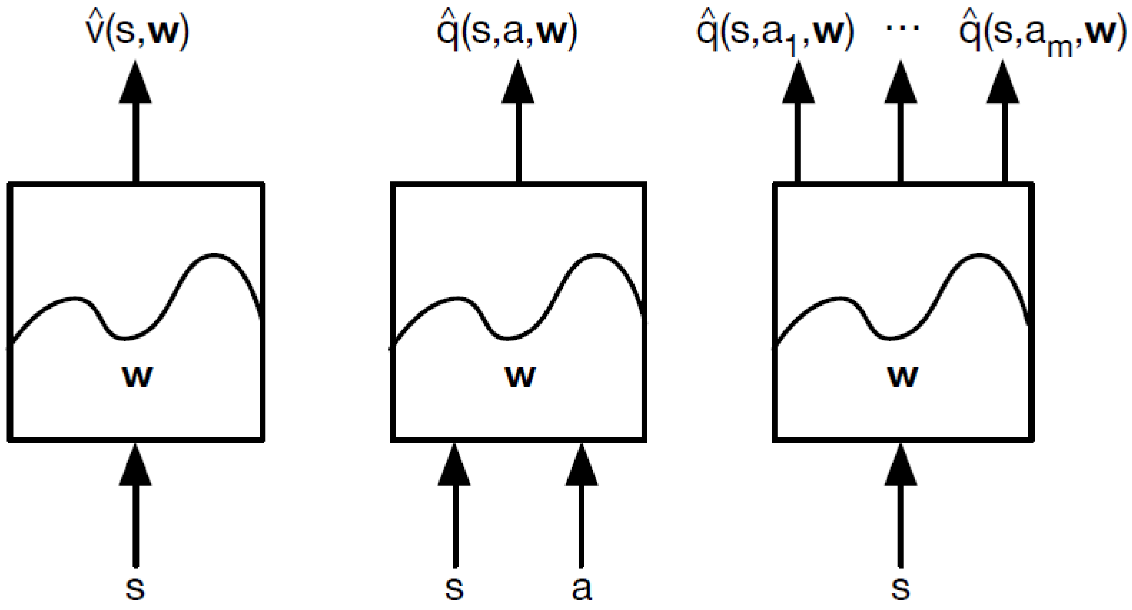
$$\text{or } \hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$

训练学习值函数，对S的值函数可以通过函数逼近来得到。

值函数逼近

问题描述

- **逼近的值函数**



每个状态要由特征向量表示

值函数逼近

问题描述

- **学习函数**
  - Linear combinations of features
  - Neural network
  - Decision tree
  - Nearest neighbour
  - Fourier / wavelet bases
  - ...

第五章 值函数逼近

5.1 值函数逼近

5.2 增量预测与控制

5.3 批量预测与控制

增量预测与控制

增量预测

Value Function Approximation

$$J(\mathbf{w}) = \mathbb{E}_\pi\left[(v_\pi(S) - \hat{v}(S,\mathbf{w}))^2\right]$$

MC:

$$V(S_t) \leftarrow V(S_t) + \alpha\boxed{(G_t - V(S_t))}$$

$$J(\mathbf{w}) = \mathbb{E}_\pi\left[G_t - \hat{v}(S_t,\mathbf{w})^2\right]$$

TD(0):

$$V(S_t) \leftarrow V(S_t) + \alpha\boxed{(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))}$$

$$J(\mathbf{w}) = \mathbb{E}_\pi\left[(R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w}))^2\right]$$

TD(λ):

$$V(s) \leftarrow V(s) + \alpha\boxed{\delta_t E_t(s)}$$

$$J(\mathbf{w}) = \mathbb{E}_\pi\left[\alpha\delta_t E_t(s)^2\right]$$

增量预测与控制

增量预测

$$J(\mathbf{w}) = \mathbb{E}_\pi\left[(v_\pi(S) - \hat{v}(S,\mathbf{w}))^2\right]$$

- **By Stochastic Gradient Descent**
  - Gradient descent finds a local minimum

$$\begin{aligned}\Delta\mathbf{w} &= -\frac{1}{2}\alpha\nabla_\mathbf{w} J(\mathbf{w}) \\ &= \alpha\mathbb{E}_\pi\left[(v_\pi(S) - \hat{v}(S,\mathbf{w}))\nabla_\mathbf{w}\hat{v}(S,\mathbf{w})\right]\end{aligned}$$

  - Stochastic gradient descent *samples* the gradient

$$\Delta\mathbf{w} = \alpha(v_\pi(S) - \hat{v}(S,\mathbf{w}))\nabla_\mathbf{w}\hat{v}(S,\mathbf{w})$$

增量预测

- **Linear Value Function Approximation**

$$\hat{v}(S, \mathbf{w}) = \mathbf{x}(S)^\top \mathbf{w} = \sum_{j=1}^{n} \mathbf{x}_j(S)\mathbf{w}_j$$

Represent state by a *feature vector*

$$\mathbf{x}(S) = \begin{pmatrix} \mathbf{x}_1(S) \\ \vdots \\ \mathbf{x}_n(S) \end{pmatrix}$$

$$\nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w}) = \mathbf{x}(S)$$

$$\Delta \mathbf{w} = \alpha(v_\pi(S) - \hat{v}(S, \mathbf{w}))\mathbf{x}(S)$$

增量预测与控制

增量预测算法

MC with Value Function Approximation

- Return G_t is an unbiased, noisy sample of true value $v_\pi(S_t)$
- Can therefore apply supervised learning to "training data":

$$\langle S_1, G_1\rangle, \langle S_2, G_2\rangle, ..., \langle S_T, G_T\rangle$$

- For example, using *linear Monte-Carlo policy evaluation*

$$\begin{aligned}\Delta \mathbf{w} &= \alpha({\color{red}G_t} - \hat{v}(S_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{v}(S_t, \mathbf{w}) \\ &= \alpha(G_t - \hat{v}(S_t, \mathbf{w}))\mathbf{x}(S_t)\end{aligned}$$

- Monte-Carlo evaluation converges to a local optimum
- Even when using non-linear value function approximation

增量预测与控制

增量预测算法

TD(0) with Value Function Approximation

- The TD-target $R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w})$ is a *biased* sample of true value $v_\pi(S_t)$
- Can still apply supervised learning to "training data":

$$\langle S_1, R_2 + \gamma \hat{v}(S_2, \mathbf{w})\rangle, \langle S_2, R_3 + \gamma \hat{v}(S_3, \mathbf{w})\rangle, ..., \langle S_{T-1}, R_T\rangle$$

- For example, using *linear TD(0)*

$$\begin{aligned}\Delta \mathbf{w} &= \alpha(R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w}))\nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w}) \\ &= \alpha \delta \mathbf{x}(S)\end{aligned}$$

- Linear TD(0) converges (close) to global optimum

增量预测与控制

增量预测算法

TD(λ) with Value Function Approximation

- The λ-return G_t^λ is also a biased sample of true value $v_\pi(s)$
- Can again apply supervised learning to "training data":

$$\left\langle S_1, G_1^\lambda \right\rangle, \left\langle S_2, G_2^\lambda \right\rangle, \ldots, \left\langle S_{T-1}, G_{T-1}^\lambda \right\rangle$$

- Forward view linear TD(λ)

$$\begin{aligned}\Delta \mathbf{w} &= \alpha(G_t^\lambda - \hat{v}(S_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{v}(S_t, \mathbf{w}) \\ &= \alpha(G_t^\lambda - \hat{v}(S_t, \mathbf{w}))\mathbf{x}(S_t)\end{aligned}$$

- Backward view linear TD(λ)

$$\begin{aligned}\delta_t &= R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w}) \\ E_t &= \gamma\lambda E_{t-1} + \mathbf{x}(S_t) \\ \Delta \mathbf{w} &= \alpha\delta_t E_t\end{aligned}$$

Forward view and backward view linear TD(λ) are equivalent

增量预测与控制

增量控制

Action-Value Function Approximation

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[(q_\pi(S,A) - \hat{q}(S,A,\mathbf{w}))^2 \right]$$

- **by SGD**

stochastic gradient descent to find a local minimum

$$-\frac{1}{2}\nabla_{\mathbf{w}} J(\mathbf{w}) = (q_\pi(S,A) - \hat{q}(S,A,\mathbf{w}))\nabla_{\mathbf{w}}\hat{q}(S,A,\mathbf{w})$$

$$\Delta\mathbf{w} = \alpha(q_\pi(S,A) - \hat{q}(S,A,\mathbf{w}))\nabla_{\mathbf{w}}\hat{q}(S,A,\mathbf{w})$$

增量预测与控制

增量控制

- **Linear Action-Value Function Approximation**

$$\hat{q}(S, A, \mathbf{w}) = \mathbf{x}(S, A)^\top \mathbf{w} = \sum_{j=1}^{n} \mathbf{x}_j(S, A)\mathbf{w}_j$$

Represent state *and* action by a *feature vector*

$$\mathbf{x}(S, A) = \begin{pmatrix} \mathbf{x}_1(S, A) \\ \vdots \\ \mathbf{x}_n(S, A) \end{pmatrix}$$

Stochastic gradient descent update

$$\nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w}) = \mathbf{x}(S, A)$$

$$\Delta \mathbf{w} = \alpha(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}))\mathbf{x}(S, A)$$

增量预测与控制

增量控制算法

- For MC, the target is the return G_t

$$\Delta \mathbf{w} = \alpha({\color{red}G_t} - \hat{q}(S_t, A_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{q}(S_t, A_t, \mathbf{w})$$

- For TD(0), the target is the TD target $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$

$$\Delta \mathbf{w} = \alpha({\color{red}R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w})} - \hat{q}(S_t, A_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{q}(S_t, A_t, \mathbf{w})$$

- For forward-view TD(λ), target is the action-value λ-return

$$\Delta \mathbf{w} = \alpha({\color{red}q_t^{\lambda}} - \hat{q}(S_t, A_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{q}(S_t, A_t, \mathbf{w})$$

- For backward-view TD(λ), equivalent update is

$$\begin{aligned}
\delta_t &= R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}) \\
E_t &= \gamma\lambda E_{t-1} + \nabla_{\mathbf{w}}\hat{q}(S_t, A_t, \mathbf{w}) \\
\Delta \mathbf{w} &= \alpha\delta_t E_t
\end{aligned}$$

第五章 值函数逼近

批量预测与控制

批量预测问题

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[(v_\pi(S) - \hat{v}(S, \mathbf{w}))^2 \right]$$

- **Least Squares Prediction (LSP)**

$$\begin{aligned} LS(\mathbf{w}) &= \sum_{t=1}^{T} (v_t^\pi - \hat{v}(s_t, \mathbf{w}))^2 \\ &= \mathbb{E}_\mathcal{D} \left[(v^\pi - \hat{v}(s, \mathbf{w}))^2 \right] \end{aligned}$$

批量预测与控制

SGD with Experience Replay (批量回放)

Given experience consisting of $\langle state, value \rangle$ pairs

$$\mathcal{D} = \{\langle s_1, v_1^\pi \rangle, \langle s_2, v_2^\pi \rangle, ..., \langle s_T, v_T^\pi \rangle\}$$

Repeat:

1. Sample state, value from experience

$$\langle s, v^\pi \rangle \sim \mathcal{D}$$

2. Apply stochastic gradient descent update

$$\Delta \mathbf{w} = \alpha (v^\pi - \hat{v}(s, \mathbf{w})) \nabla_{\mathbf{w}} \hat{v}(s, \mathbf{w})$$

Converges to least squares solution

$$\mathbf{w}^\pi = \underset{\mathbf{w}}{\operatorname{argmin}}\ LS(\mathbf{w})$$

批量预测与控制

MSE 预测

LSP 最优解

We can solve the least squares solution directly

$$\mathbb{E}_{\mathcal{D}}\left[\Delta \mathbf{w}\right] = 0$$

Linear LSP 最优解

Using *linear* value function approximation $\hat{v}(s, \mathbf{w}) = \mathbf{x}(s)^\top \mathbf{w}$

$$\alpha \sum_{t=1}^{T} \mathbf{x}(s_t)(v_t^\pi - \mathbf{x}(s_t)^\top \mathbf{w}) = 0$$

MSE 预测

Linear LSP 算法

LSMC

$$0 = \sum_{t=1}^{T} \alpha(G_t - \hat{v}(S_t, \mathbf{w}))\mathbf{x}(S_t)$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} \mathbf{x}(S_t)\mathbf{x}(S_t)^\top\right)^{-1} \sum_{t=1}^{T} \mathbf{x}(S_t)G_t$$

LSTD

$$0 = \sum_{t=1}^{T} \alpha(R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w}))\mathbf{x}(S_t)$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} \mathbf{x}(S_t)(\mathbf{x}(S_t) - \gamma\mathbf{x}(S_{t+1}))^\top\right)^{-1} \sum_{t=1}^{T} \mathbf{x}(S_t)R_{t+1}$$

LSTD(λ)

$$0 = \sum_{t=1}^{T} \alpha\delta_t E_t$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} E_t(\mathbf{x}(S_t) - \gamma\mathbf{x}(S_{t+1}))^\top\right)^{-1} \sum_{t=1}^{T} E_t R_{t+1}$$

批量预测与控制

批量控制问题

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}))^2 \right]$$

- **Least Squares Control**

$$\begin{aligned} LS(\mathbf{w}) &= \mathbb{E}_\mathcal{D} \left[(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}))^2 \right] \\ &= \sum_{t=1}^{T} (q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}))^2 \end{aligned}$$

Experience Replay in Deep Q-Networks (DQN)

Take action a_t according to ϵ-greedy policy

Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory $\mathcal{D}$

Sample random mini-batch of transitions (s, a, r, s') from $\mathcal{D}$

Compute Q-learning targets w.r.t. old, fixed parameters w^-

Optimise MSE between Q-network and Q-learning targets

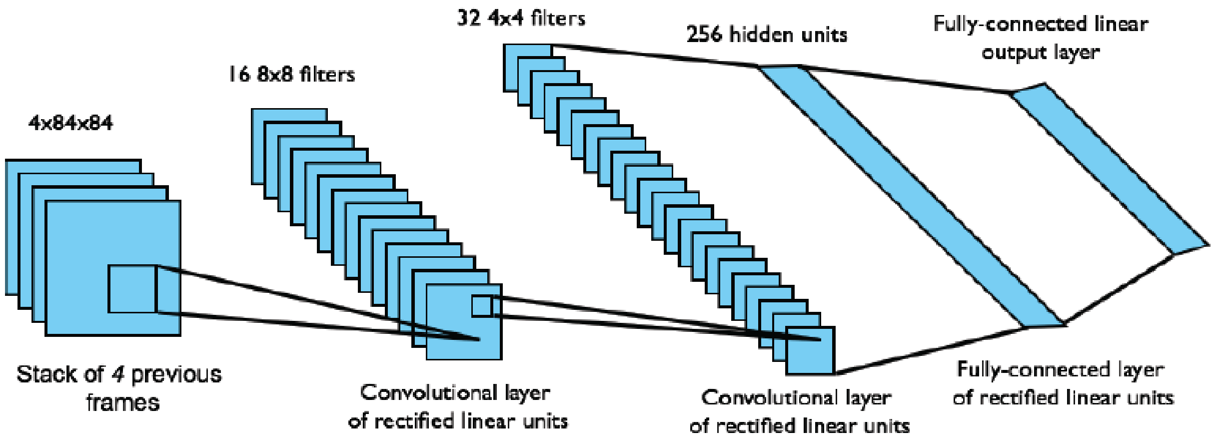
$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

Using variant of stochastic gradient descent

Experience Replay in Deep Q-Networks (DQN)

$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step



批量预测与控制

Least Squares Q-Learning

- **Using Linear Action-State Function**

LSTDQ algorithm: solve for total update = zero

$$0 = \sum_{t=1}^{T} \alpha(R_{t+1} + \gamma \hat{q}(S_{t+1}, \pi(S_{t+1}), \mathbf{w}) - \hat{q}(S_t, A_t, \mathbf{w}))\mathbf{x}(S_t, A_t)$$

$$\mathbf{w} = \left(\sum_{t=1}^{T} \mathbf{x}(S_t, A_t)(\mathbf{x}(S_t, A_t) - \gamma \mathbf{x}(S_{t+1}, \pi(S_{t+1})))^\top \right)^{-1} \sum_{t=1}^{T} \mathbf{x}(S_t, A_t) R_{t+1}$$

Least Squares Q-Learning

- The following pseudocode uses LSTDQ for policy evaluation
- It repeatedly re-evaluates experience $\mathcal{D}$ with different policies

function **LSPI-TD**$(\mathcal{D}, \pi_0)$
    $\pi' \leftarrow \pi_0$
    repeat
        $\pi \leftarrow \pi'$
        $Q \leftarrow$ **LSTDQ**$(\pi, \mathcal{D})$
        for all $s \in \mathcal{S}$ **do**
            $\pi'(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q(s, a)$
        end for
    until $(\pi \approx \pi')$
    return π
end function

本讲参考文献

1. Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. (Second edition, in progress，draft).
2. David Silver，Slides@《Reinforcement Learning:An Introduction》,2016.