Lecture 6: 价值函数近似

前瞻实验室跨媒体组
董嘉蓉
dongjiarong@ict.ac.cn
2019年4月22日

提纲

- 引言
- 增量方法(Incremental Methods)
- 批方法(Batch Methods)

引言

简单回顾

强化学习的目标：是找到一个策略 π，使得这个策略下每个状态的价值最大。

- Model-based 方法
  - 动态规划(策略迭代，价值迭代)
- Model-free方法
  - 策略评估：MC, TD
  - 策略控制：Sarsa, Q-learning

引言

- 大型问题

- Backgammon: 10^{20} states
- Computer Go: 10^{170} states
- Helicopter: continuous state space

如何使用model-free的方法解决上述问题？

引言

价值函数近似

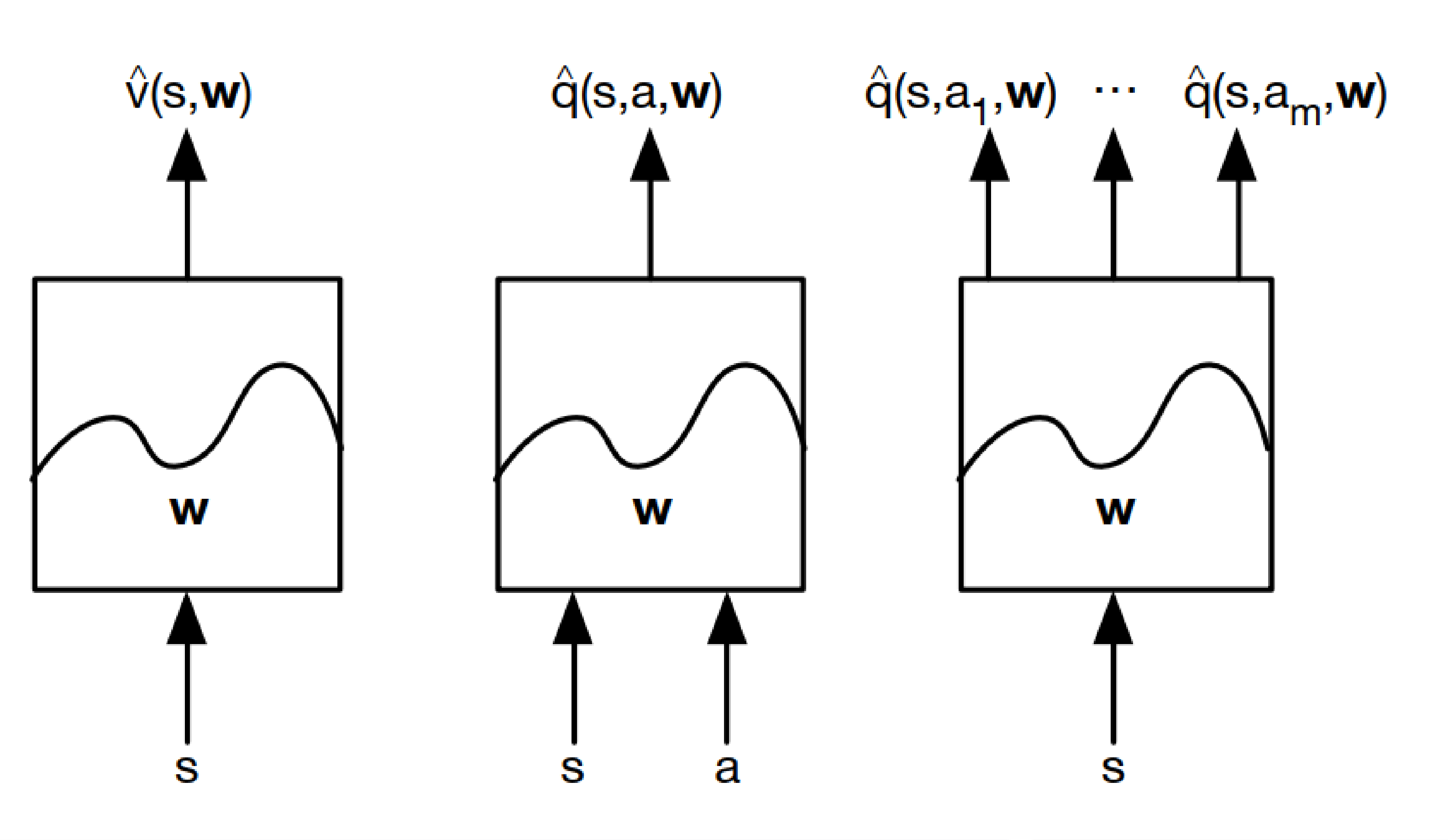
- 之前的内容均是采用查表法表示价值函数
  - 每个状态s都会记录一个V(s)
  - 每个状态-价值(s,a)都会对应一个Q(s,a)
- 大型MDPs存在的问题
  - 需要存储大量的状态和动作
  - 学习每个状态的价值速度太慢
- 解决方法
  - 采用价值函数近似

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

⇨ 好处
- 减少记忆存储空间
- 可以从已知数据泛化到未知数据

引言

■ 价值函数近似建模方式



引言

- 采用什么函数近似？
  - 线性特征组合
  - 神经网络

  可微函数

  - 决策树
  - 最近邻
  - 傅里叶/小波 基
  - …

与监督学习训练方法的区别：动态变化(non-stationary)，非独立(non-iid)的数据

增量方法

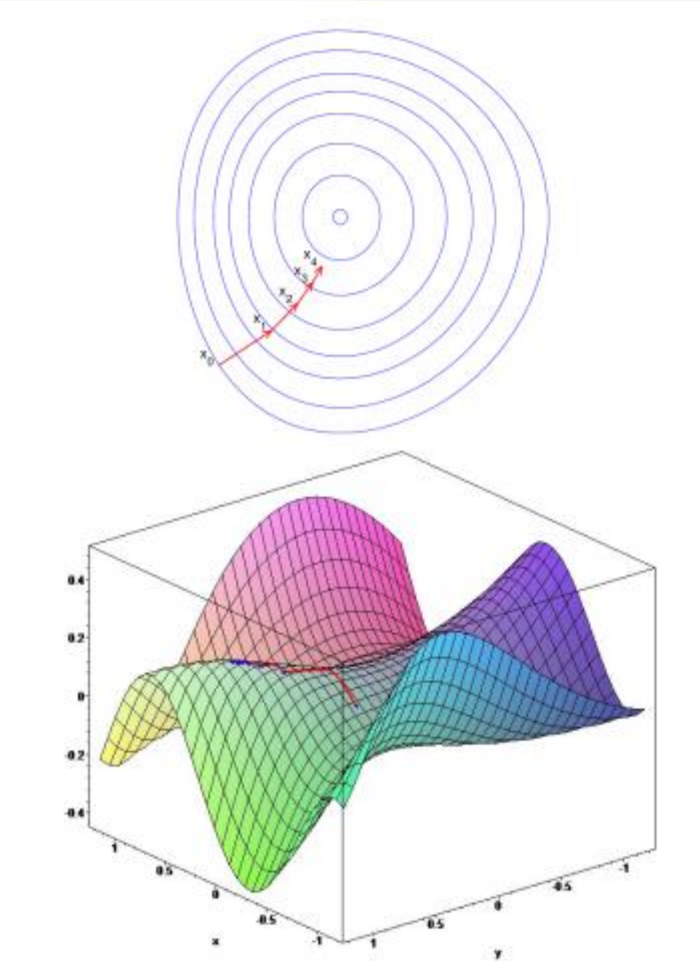
梯度下降

- 关于参数*w*的可微函数*J(w)*
- 梯度

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \begin{pmatrix} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_n} \end{pmatrix}$$

- 找到*J(w)*的局部最小值

$$\Delta \mathbf{w} = -\frac{1}{2}\alpha \nabla_{\mathbf{w}} J(\mathbf{w})$$



增量方法

采用SGD优化价值函数

- 目标：寻找最小化均方误差的参数w

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

真实价值函数　近似价值函数

- 采用随机梯度下降SGD优化参数

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

⇨ 类似于监督学习的训练方法

增量方法

特征向量

- 用特征向量表示状态s

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

- 举例
  - 机器人与目标的距离
  - 股票的走势

增量方法

线性函数近似

- 采用线性特征组合近似价值函数

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

- 目标方程

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

- 采用随机梯度下降收敛到最优值

$$\begin{aligned} \nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w}) &= \mathbf{x}(S) \\ \Delta\mathbf{w} &= \alpha(v_\pi(S) - \hat{v}(S,\mathbf{w}))\mathbf{x}(S) \end{aligned}$$

增量方法

查表方法与线性函数近似

查表法是线性函数近似的特例

- 查表特征

$$\mathbf{x}^{table}(S) = \begin{pmatrix} \mathbf{1}(S = s_1) \\ \vdots \\ \mathbf{1}(S = s_n) \end{pmatrix}$$

- 参数w即每个状态对应的价值v(s)

$$\hat{v}(S, \mathbf{w}) = \begin{pmatrix} \mathbf{1}(S = s_1) \\ \vdots \\ \mathbf{1}(S = s_n) \end{pmatrix} \cdot \begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{pmatrix}$$

增量方法

增量预测算法

- 不存在真实的价值函数$V\pi$ (s)，只有奖励reward。（强化学习与监督学习的区别：不存在监督信号）

- 实际中，采用价值函数$V\pi$ (s)的估计值

  - For MC, the target is the return G_t

$$\Delta \mathbf{w} = \alpha(G_t - \hat{v}(S_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{v}(S_t, \mathbf{w})$$

  - For TD(0), the target is the TD target $R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w})$

$$\Delta \mathbf{w} = \alpha(R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{v}(S_t, \mathbf{w})$$

  - For TD(λ), the target is the λ-return G_t^λ

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

增量方法

蒙特卡洛应用于价值函数近似

- MC的反馈G_t是真实价值$V\pi\ (s)$的无偏，有噪声估计，”训练集”为：

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

- 使用线性MC进行价值评估，参数的修正值为：

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

- MC评估无论线性还是非线性均能收敛到局部最优解

增量方法

时序差分应用于价值函数近似

- TD的目标$R_{t+1}+\gamma\hat{V}(S_{t+1})$是真实价值$V\pi(s)$的有偏估计，”训练集”为：

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

- 使用线性TD(0)进行价值评估，参数的修正值为：

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

- 线性TD(0)收敛(近似)到全局最优解

增量方法

TD(λ)应用于价值函数近似

- λ-return$G_t^{\lambda}{}_{t+1}$是真实价值$V\pi\ (s)$的有偏估计，" 训练集"为：

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

- Forward view 线性TD(λ)，参数的修正值为：

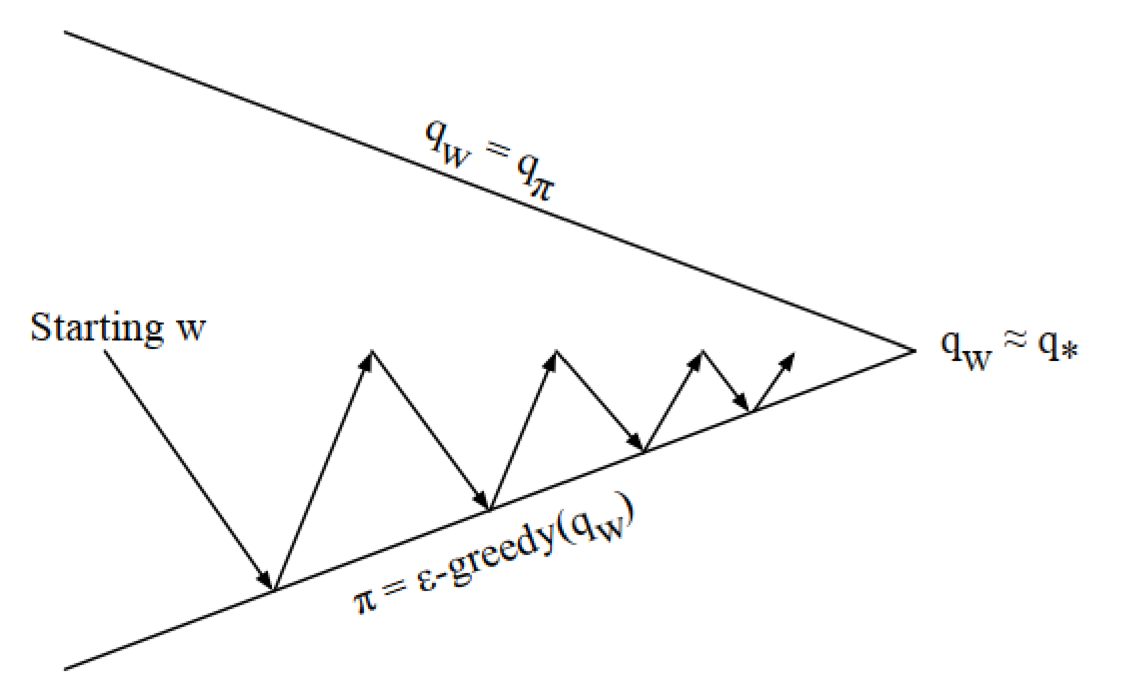
$$\begin{aligned}\Delta \mathbf{w} &= \alpha(G_t^{\lambda} - \hat{v}(S_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{v}(S_t, \mathbf{w}) \\ &= \alpha(G_t^{\lambda} - \hat{v}(S_t, \mathbf{w}))\mathbf{x}(S_t)\end{aligned}$$

- Backward view 线性TD(λ)，参数的修正值为：

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

增量方法

增量控制算法



策略评估(policy evaluation) 近似策略评估 $\hat{q}(\cdot,\cdot,\mathbf{w}) \approx q_\pi$

策略改进(policy improvement) ϵ-greedy

增量方法

动作价值函数近似

- 近似动作价值(action-value)函数

$$\hat{q}(S, A, \mathbf{w}) \approx q_\pi(S, A)$$

- 最小化均方误差

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

- 采用随机梯度下降法优化

$$\begin{aligned} -\frac{1}{2}\nabla_{\mathbf{w}} J(\mathbf{w}) &= (q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}))\nabla_{\mathbf{w}}\hat{q}(S, A, \mathbf{w}) \\ \Delta\mathbf{w} &= \alpha(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}))\nabla_{\mathbf{w}}\hat{q}(S, A, \mathbf{w}) \end{aligned}$$

增量方法

线性动作价值函数近似

- 与预测类似，用特征向量表示(S,A)

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

- 采用线性特征组合算法近似action-value函数

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

- 采用随机梯度下降法优化

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

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

增量方法

增量控制算法

- For MC, the target is the return G_t

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

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

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

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

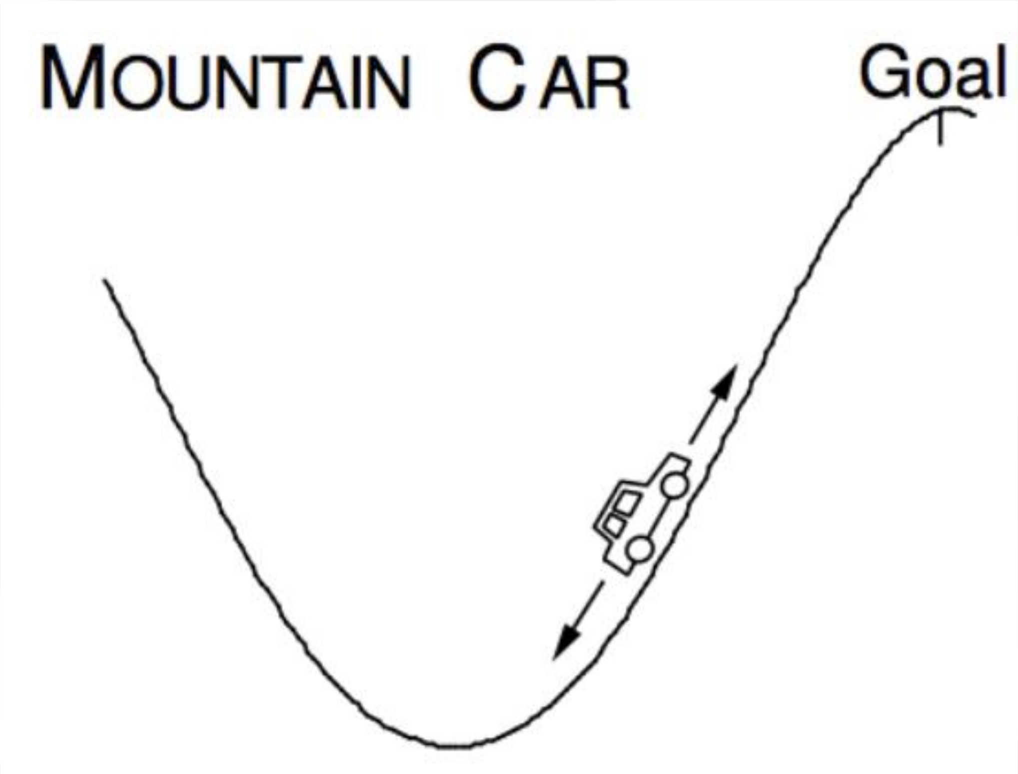
$$\Delta \mathbf{w} = \alpha(q_t^{\lambda} - \hat{q}(S_t, A_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{q}(S_t, A_t, \mathbf{w})$$

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

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

增量方法

例子：Mountain Car

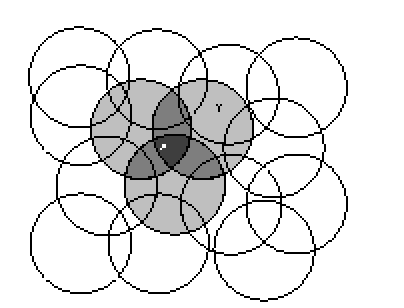


- 问题说明：车困在深坑中，需要开足马力到达goal。
  - 状态（二维连续空间）：
    当前的位置p~(-1.2, 0.6)
    当前的速度V~(-0.07, 0.07)
  - 动作(一维离散空间)：motor~（left, neutral, right）
  - 奖励：-1/每时间步
  - 终止条件：到达目标(位置>=0.6)

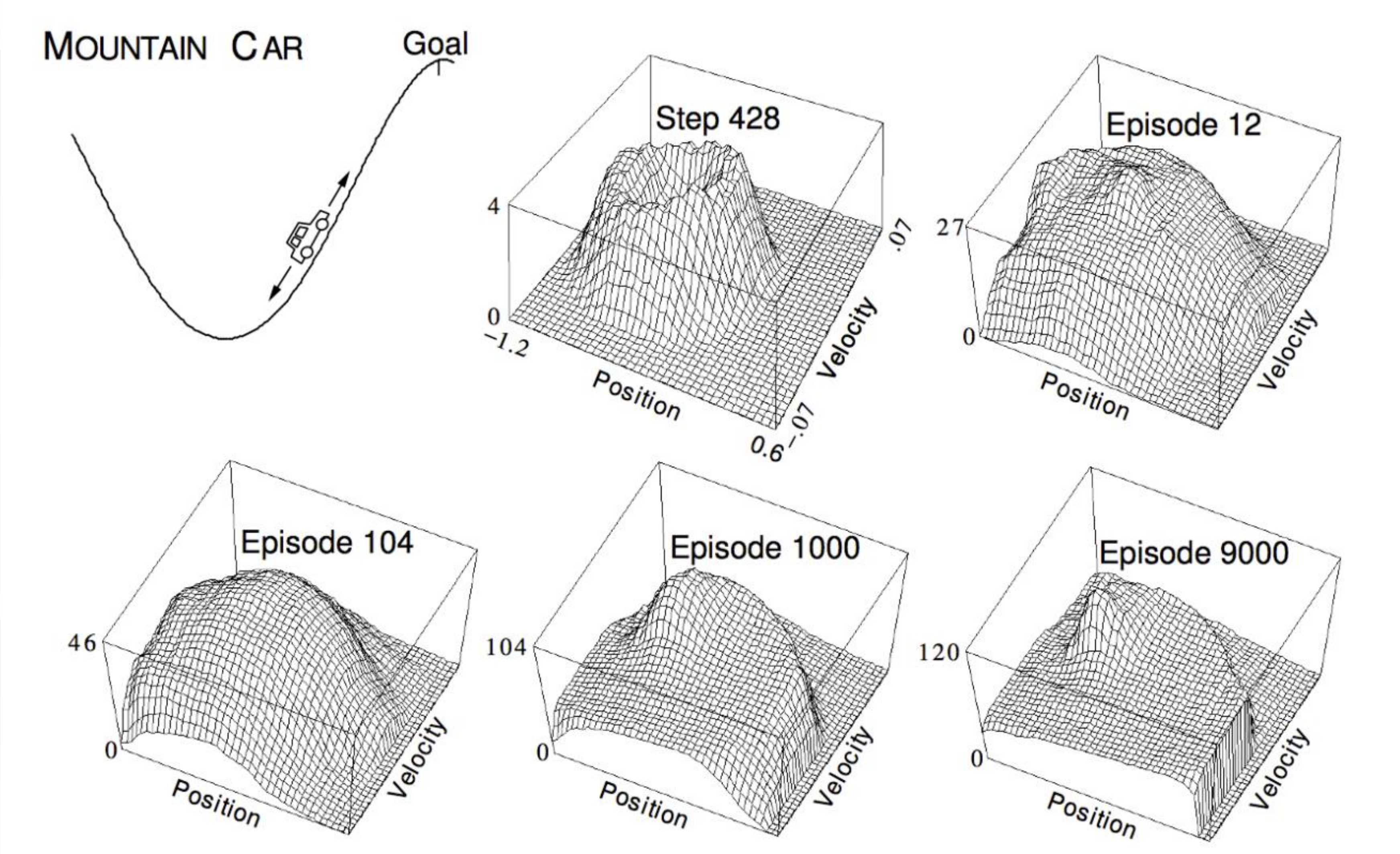
增量方法

■ 例子：Linear Sarsa with Coarse Coding in Mountain Car

采用coarse coding表示当前状态：



线性Sarsa进行优化



增量方法

预测算法的收敛性

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	TD(0)	✓	✓	✗
	TD(λ)	✓	✓	✗
Off-Policy	MC	✓	✓	✓
	TD(0)	✓	✗	✗
	TD(λ)	✓	✗	✗

增量方法

控制算法的收敛性

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
Gradient Q-learning	✓	✓	✗

(✓) = chatters around near-optimal value function

批方法

- 为什么采用批方法？

  - 增量方法样本利用率低

  ⇨ 使用批方法高效率利用样本拟合价值函数

  Hint: 保留智能体的经验

批方法

经验回放

- 给定经验数据

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

- Repeat:

1. Sample state, value from experience

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

2. Apply stochastic gradient descent update

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

- 收敛到最小二乘解

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

批方法

DQN

- DQN采用经验回放和固定Q-target

  - 依据ϵ-$greedy$采样动作a_t;
  - 将经验数据$(s_t, a_t, r_{t+1}, s_{t+1})$存储在回放记忆体D中；
  - 从D中随机采样mini-batch的样本(s, a, r, s');
  - 根据过去的固定参数w^-计算Q-target；
  - 优化Q-network与Q-target之间的MSE；

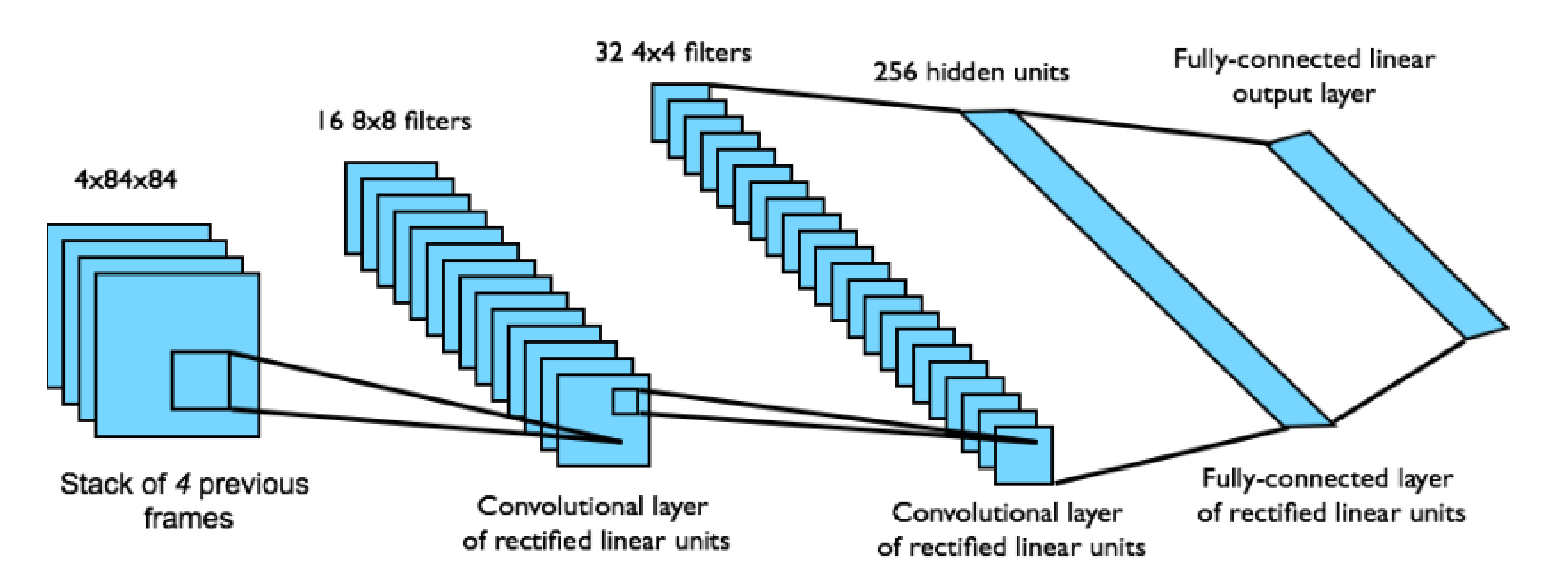
$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

  - 使用SGD的变体更新参数。

批方法

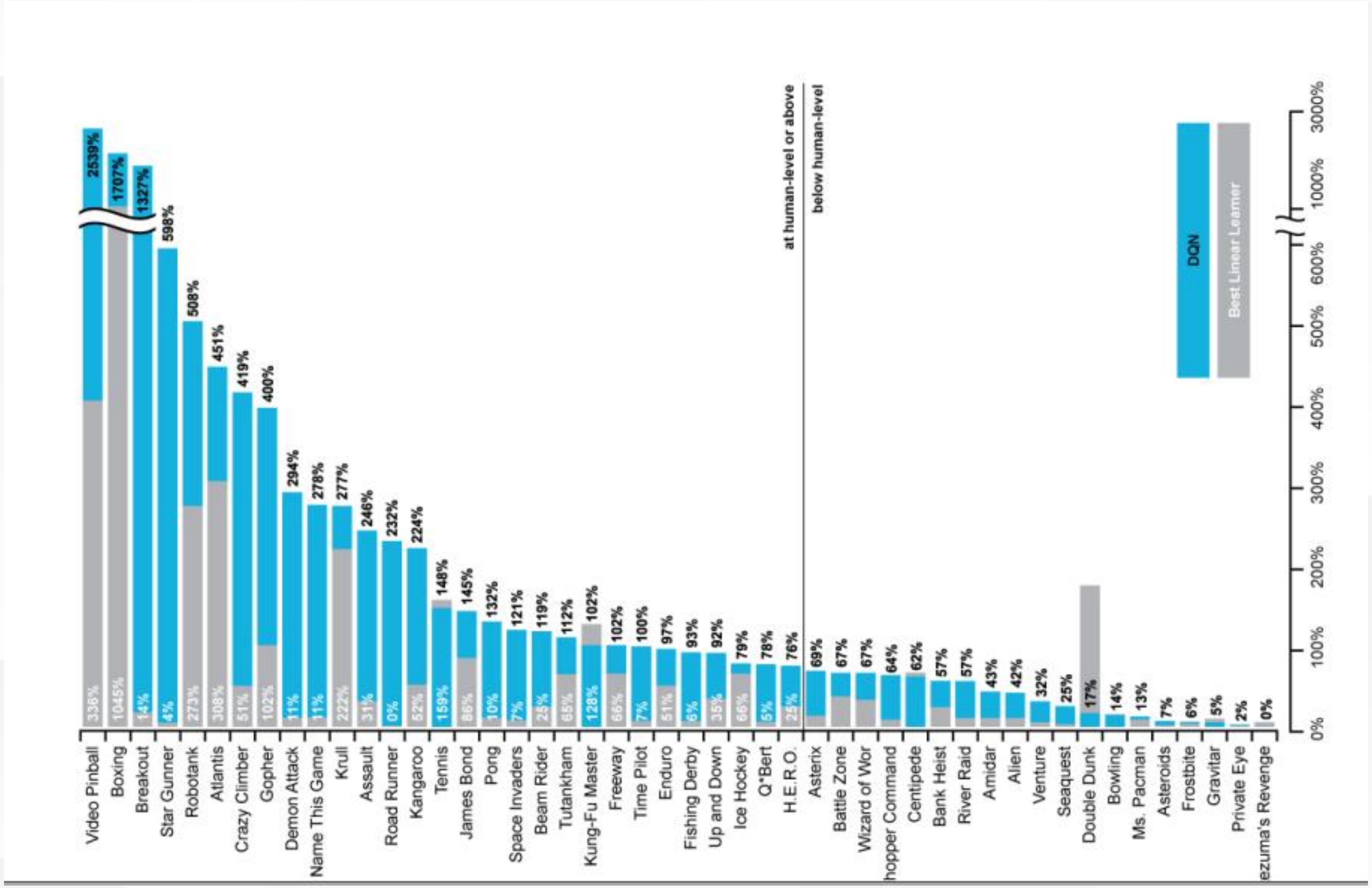
DQN in Atari

- 从像素s端到端的训练价值函数Q(s,a);
- 输入的状态s是游戏最近4帧的堆叠；
- 网络Q(s,a)的输出为18个按钮的位置;
- 每步的得分变化作为reward。



批方法

- DQN in Atari



批方法

DQN in Atari

	Replay Fixed-Q	Replay Q-learning	No replay Fixed-Q	No replay Q-learning
Breakout	316.81	240.73	10.16	3.17
Enduro	1006.3	831.25	141.89	29.1
River Raid	7446.62	4102.81	2867.66	1453.02
Seaquest	2894.4	822.55	1003	275.81
Space Invaders	1088.94	826.33	373.22	301.99

The End