第四课 价值函数近似 上

1. $large\ MDP$问题：
   - 行为价值函数较多，浪费存储空间
   - 一个个计算值函数很费时
2. 解决方法：近似
   - $$\hat{v}(s,\vec{w}) \approx v^{\pi}(s)$$ $$\hat{q}(s,a,\vec{w}) \approx q^{\pi}(s,a)$$ $$\hat{\pi}(s,a,\vec{w}) \approx \pi(a|s)$$
   - 计算的种类

     $$1.\ in:s \rightarrow out:\hat{v}(s,\vec{w})$$
     $$2.\ in:s,a \rightarrow out:\hat{q}(s,a,\vec{w})$$
     $$3.\ in:s \rightarrow out:\hat{q}(s,a_1,\vec{w})\cdots,\hat{q}(s,a_n,\vec{w})$$
   - 近似常用方法：
     - 线性模型
     - 神经网络
     - 决策树
     - 最邻近
3. $value\ funtion\ approximation\ with\ an\ oracle$
   - 已知每个状态下的真实值函数值，采用最小二乘，每个状态用一个特征向量$\vec{x}(s)=(x_1(1),\cdots,x_n(s))^T$

     $$J(\vec{w})=E_{\pi}[(v^{\pi}(s)-\hat{v}(s,\vec{w}))^2]$$
     $$\Delta\vec{w}=-\frac{1}{2}\alpha\nabla_{\vec{w}}J(\vec{w})$$
     $$\vec{w}_{t+1}=\vec{w}_t+\Delta\vec{w}$$
4. $linear\ value\ function\ approximation$

   $$\hat{v}(s,\vec{w})=\vec{x}(s)^T\vec{w}$$
   $$\Delta\vec{w}=\alpha(v^{\pi}(s)-\hat{v}(s,\vec{w}))\vec{x}(s)$$

   SGD能达到全局最优
5. $linear\ VFA\ with\ table\ lookup\ feature$

   类似于将状态进行$one-hot$编码

   $$\vec{x}^{table}(s)=(I(s=s_1),\cdots,I(s=s_n))^T$$
   $$\hat{v}(s,\vec{w})=(I(s=s_1),\cdots,I(s=s_n))(w_1,\cdots,w_n)^T$$
   $$thus\ \hat{v}(s_k,\vec{w})=w_k$$

   这样表示的$\vec{w}$中每个分量即为对应状态的值函数值
6. $model\ free\ prediction$的VFA
   - 现实中一般无法得到每个状态对应值函数的真实值，于是在更新$\vec{w}$时，用$target\ value$代替值函数真实值
   - 对于蒙特卡洛方法，用$return$代替值函数真实值，$return$是值函数的无偏估计，但有噪声

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

- 对于时序差分算法（仅针对$TD(0)$）,用$TD\ target$代替值函数真实值，$TD\ target$是值函数的有偏估计

$$\Delta\vec{w}=\alpha(R_{t+1}+\gamma\hat{v}(s_{t+1},\vec{w})-\hat{v}(s_t,\vec{w}))\nabla_{\vec{w}}\hat{v}(s_t,\vec{w})$$

  这里这个梯度称为$semi-gradient$，由于$TD\ target$中包含了$\vec{w}$

7. $control\ with\ VFA$
   - 策略评估：$\hat{q}(\cdots,\vec{w})\approx q^{\pi}$
   - 策略改进：$\epsilon-greedy$
   - $AVFA(action\ value\ function\ approximation)$:

$$\hat{q}(s,a,\vec{w})\approx q^{\pi}(s,a)$$
$$J(\vec{w})=E_{\pi}[(q^{\pi}(s,a)-\hat{q}(s,a,\vec{w}))^2]$$
$$\Delta\vec{w}=\alpha(q^{\pi}(s,a)-\hat{q}(s,a,\vec{w})\nabla_{\vec{w}}\hat{q}(s.a.\vec{w})$$

8. $incremental\ control\ algorithm$
   - MC: $\Delta\vec{w}=\alpha(G_t-\hat{q}(s_t,a_t,\vec{w}))\nabla_{\vec{w}}\hat{q}(s_t,a_t,\vec{w})$
   - $sarsa$: $\Delta\vec{w}=\alpha(R_{t+1}+\gamma\hat{q}(s_{t+1},a_{t+1},\vec{w})-\hat{q}(s_t,a_t,\vec{w}))\nabla_{\vec{w}}\hat{q}(s_t,a_t,\vec{w})$
   - $Q-learning$:
     $\Delta\vec{w}=\alpha(R_{t+1}+\gamma\max\limits_{a}\hat{q}(s_{t+1},a_{t+1},\vec{w})-\hat{q}(s_t,a_t,\vec{w}))\nabla_{\vec{w}}\hat{q}(s_t,a_t,\vec{w})$
   - $TD\ with\ VFA$的梯度不是很正确，且在更新中用到了两次估计（$bellman\ backup$和VFA）于是在$off\ policy$或非线性时可能发散
9. $the\ deadly\ triad\ of\ the\ danger\ of\ instability\ and\ divergence$
   - $funtion\ approximation$
   - $bootstrapping$：用估计量更新目标（如动态规划或者TD）而非用真实$reward$或完整$return$
   - $offpolicy\ training$：目标策略与行为策略不同
10. 这些近似方法在不同算法下的收敛性

	table lookup	linear	non-linear
MC	$\surd$	($\surd$)	$\times$
Sarsa	$\surd$	($\surd$)	$\times$
Q-learning	$\surd$	$\times$	$\times$

11. $batch\ reinforcement\ learning$
    - 增量更新（$incremental\ gradient\ descend$）不是$sample\ efficient$的。基于批量的方法企图在$a\ batch\ of\ agent's\ experience$中找到最佳拟合值函数
    - 其余类似，区别是训练数据序对$<s_t,v_t^{\pi}>$可以来自于一个或多个$episode$