第十四讲：强化学习与深度强化学习

最优控制的智能方法之四

张杰

人工智能学院
中国科学院大学

复杂系统管理与控制国家重点实验室
中国科学院自动化研究所

2017 年 11 月 2 日

Table of Contents

1. 回顾：Markov 决策过程
2. Policy Evaluation (策略评估)
3. Policy Iteration (策略迭代)
4. Value Iteration (值迭代)
5. Model-free Prediction(无模型预测)
6. Model-free Control(无模型控制)
7. 深度强化学习

Table of Contents

1. 回顾：Markov 决策过程
2. Policy Evaluation (策略评估)
3. Policy Iteration (策略迭代)
4. Value Iteration (值迭代)
5. Model-free Prediction(无模型预测)
6. Model-free Control(无模型控制)
7. 深度强化学习

智能体的基本要素

在强化学习中，智能体可能具有如下要素

- 策略 Policy：建模智能体的行为
- 值函数 Value function：建模智能体对状态和/或控制的估值
- 模型 Model：智能体对环境的表示 representation

Policy (控制策略，控制律)

控制策略policy从状态到控制的映射。本课考察稳态策略

- 确定策略

$$a = \pi(s)$$

- 随机策略

$$\pi(a|s) = P(A_t = a|S_t = s)$$

若求得行动值函数 $q_*(s, a)$，有确定性最优策略

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \text{argmax}_{a \in A}\, q_*(s, a), \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Value Function （值函数）

一个控制策略的值函数value function定义为期望累积收益

$$v_\pi(s) = E_\pi(G_t|S_t = s)$$
$$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$$

其中，

$$G_t := R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$$

- 一个控制策略的值函数用于估计这个策略下特定状态的优劣
- 同时也是对这个策略的评价
- 最优值函数，或简称值函数，是任意策略的值函数的极大值

Model （模型）

智能体用模型model估计下一时刻的环境状态和奖励，对于未知的随机系统常用状态转移矩阵和期望收益表示

$$\mathcal{P}(s,a,s') = P(S_{t+1}=s'|S_t=s, A_t=a)$$
$$\mathcal{R}(s,a) = E(R_{t+1}|S_t=s, A_t=a)$$

求解 Bellman 方程

- Bellman 方程是非线性的，一般情况下没有解析解
- 使用迭代方法
  - 值迭代
  - 策略迭代
  - Q-学习

- Prediction
  - Input: MDP $<\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma>$ 状态空间、控制空间、转移概率、奖励函数、折现。策略 π
  - Output: 策略 π 的状态值函数 v_π
- Control
  - Input: MDP $<\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma>$
  - Output: 最优策略 π^*（和值函数 v^*）

Table of Contents

1 回顾：Markov 决策过程

2 Policy Evaluation (策略评估)

3 Policy Iteration (策略迭代)

4 Value Iteration (值迭代)

5 Model-free Prediction(无模型预测)

6 Model-free Control(无模型控制)

7 深度强化学习

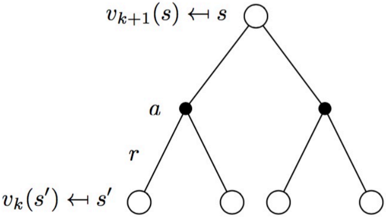
迭代策略评估 1/2

- 任务: 计算策略 π 的总期望收益
- 解: 迭代利用 Bellman 期望方程

$$v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_\pi$$

- 在迭代 $k+1$
- 对任意状态 $s \in S$
- 根据 $v_k(s')$ 更新 $v_{k+1}(s)$

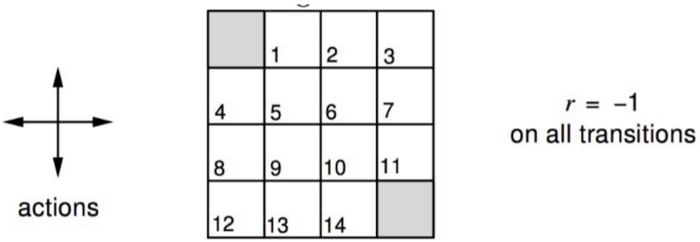
迭代策略评估 2/2



$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(\mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{P}(s,a,s')v_k(s'))$$

$$v_{k+1} = \mathcal{R}_\pi + \gamma \mathcal{P}_\pi v_k$$

例子：Grid-world 评估一个策略 1/3



- $\gamma = 1$
- 两个灰色是终端状态
- 跳出方框将保持状态不变
- 除了终端无收益，Reward 总是 -1
- 策略：

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

例子：Grid-world 评估一个策略 2/3

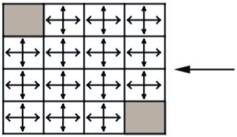
初始化策略的状态值函数

$k = 0$

v_k for the Random Policy

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

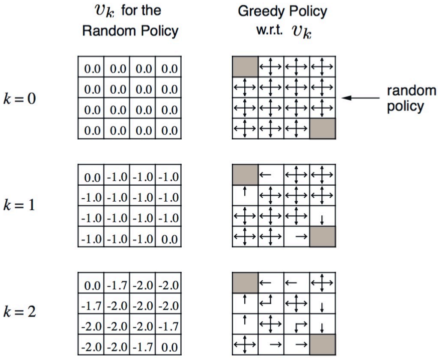
Greedy Policy w.r.t. v_k



random policy

若该值函数为“真”，依此可得右侧的贪婪策略

例子：Grid-world 评估一个策略 3/3



以 $k = 1$， 1 行 2 列为例

$$1/4[-1] + 1/4[-1+0] * 3 = -1.0$$

$k = 2$

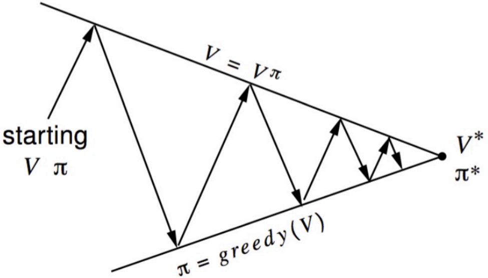
$$1/4 * [-1] + 1/4[-1-1] * 3 = -1.75$$
$$1/4 * [-1-1] * 4 = -2$$

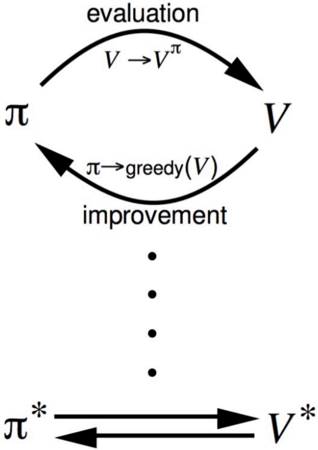
$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(\mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{P}(s,a,s') v_k(s'))$$

Table of Contents

1. 回顾：Markov 决策过程
2. Policy Evaluation (策略评估)
3. Policy Iteration (策略迭代)
4. Value Iteration (值迭代)
5. Model-free Prediction(无模型预测)
6. Model-free Control(无模型控制)
7. 深度强化学习

Policy Iteration，策略迭代





Policy evaluation Estimate v_π

Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

Greedy policy improvement

策略改进 1/2

- 考察一个确定性策略 $a=\pi(s)$，其值函数为 $q_\pi(s,a), v_\pi(s)$
- 定义贪婪策略

$$\pi'(s) = \operatorname*{argmax}_{a\in A} q_\pi(s,a)$$

$$q_\pi(s,\pi'(s)) = \max_{a\in A} q_\pi(s,a) \geq q_\pi(s,\pi(s)) = v_\pi(s)$$

于是

$$\begin{aligned}
v_\pi(s) \leq q_\pi(s,\pi'(s)) &= E_{\pi'}(R_{k+1} + \gamma v_\pi(S_{k+1})|S_k = s) \\
&\leq E_{\pi'}(R_{k+1} + \gamma q_\pi(S_{k+1},\pi'(S_{k+1}))|S_k = s) \\
&\leq E_{\pi'}(R_{k+1} + \gamma R_{k+2} + \gamma^2 q_\pi(S_{k+2},\pi'(S_{k+2}))|S_k = s) \\
&\leq E_{\pi'}(R_{k+1} + \gamma R_{k+2} + \ldots|S_k = s) = v_{\pi'}(s)
\end{aligned}$$

改进停止 2/2

若策略改进停止，即对于任意状态，

$$q_\pi(s, \pi'(s)) = \max_{a \in A} q_\pi(s, a) = q_\pi(s, \pi(s)) = v_\pi(s)$$

则 Bellman 方程已经满足

$$v_\pi(s) = \max_{a \in A} q_\pi(s, a)$$

即，$v_\pi(s) = v^*(s), \forall s \in S$. π 最优

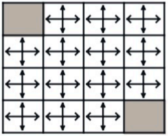
例子：Grid-world 策略迭代

v_k for the Random Policy

Greedy Policy w.r.t. v_k

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



random policy

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

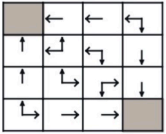


Table of Contents

1 回顾：Markov 决策过程

2 Policy Evaluation (策略评估)

3 Policy Iteration (策略迭代)

4 Value Iteration (值迭代)

5 Model-free Prediction(无模型预测)

6 Model-free Control(无模型控制)

7 深度强化学习

值迭代 1/2

- 任务: 求解最优策略 π (或值函数 v)
- 解: 迭代利用 Bellman 方程

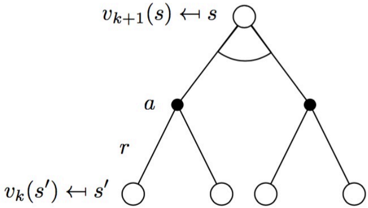
$$v_1 \to v_2 \to \ldots \to v^*$$

- 在迭代 $k+1$
- 对任意状态 $s \in S$
- 根据 $v_k(s')$ 更新 $v_{k+1}(s)$

Remark 1

解得过程中并无显式策略, v_k 也不是某个策略的值

值迭代 2/2



$$v_{k+1}(s) = \max_{a \in \mathcal{A}} [\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') v_k(s')]$$

$$v_{k+1} = \max_{a \in \mathcal{A}} [\mathcal{R}_a + \gamma \mathcal{P}_a v_k]$$

例子：最短路

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} [\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') v_k(s')]$$

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

小结

- 策略评估：需已知下时刻状态（的分布）
- 策略迭代：需策略评估
- 值迭代：需下时刻状态（的分布）
- 模型未知?

Table of Contents

1 回顾：Markov 决策过程

2 Policy Evaluation (策略评估)

3 Policy Iteration (策略迭代)

4 Value Iteration (值迭代)

5 Model-free Prediction(无模型预测)

6 Model-free Control(无模型控制)

7 深度强化学习

Monte-Carlo 策略评估

- 任务: 实施策略 π，利用样本评估值函数

$$S_1, A_1, R_2, S_2, A_2, \ldots \sim \pi$$

- 想法：用实际的总收益近似期望收益

$$v_\pi(s) = E_\pi(G_t | S_t = s)$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$$

- 实施策略 π 直到终止，n 次
- 对每条“路径”首次出现 s 的时刻 t，令 $N(s) \leftarrow N(s) + 1, S(s) \leftarrow S(s) + G_t$
- $V(s) = S(s)/N(s)$，根据大数定律，$n \to$ 时收敛至 $v_\pi(s)$

均值的增量计算

序列 $x_1, x_2, \ldots$ 的均值序列 $\mu_1, \mu_2, \ldots$ 满足

$$\begin{aligned}\mu_k &= \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{1}{k}(x_k + \sum_{i=1}^{k-1} x_i) = \frac{1}{k}(x_k + (k-1)\mu_{m-1}) \\ &= \mu_{k-1} + \frac{1}{k}(x_k - x_{k-1})\end{aligned}$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

常使用

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

时间差分 Temporal-Difference(TD)

回顾有模型动态规划的策略评估根据 $v_k(s')$ 更新 $v_{k+1}(s)$

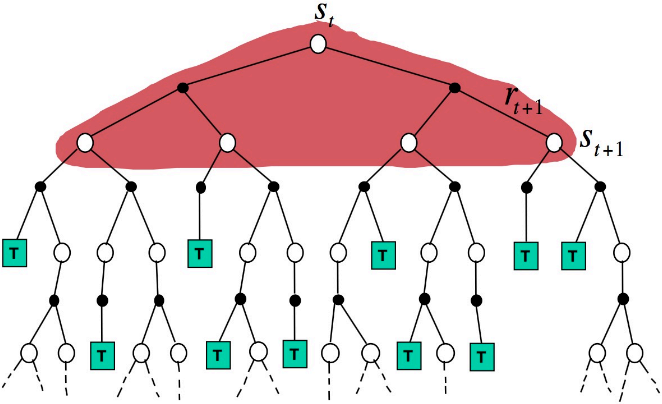
$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(\mathcal{R}(s,a) + \gamma \sum_{s'} \mathcal{P}(s,a,s')v_k(s'))$$

可使用 $R_{t+1} + \gamma V(S_{t+1})$ 近似替代 G_t，得 TD 方法

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

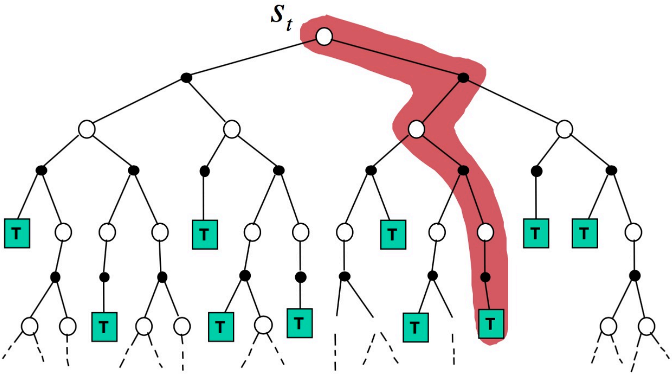
Dynamic Programming

$$V(S_t) \leftarrow E_\pi[R_{t+1} + \gamma V(S_{t+1})]$$



Monte-Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$



Temporal-Difference

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

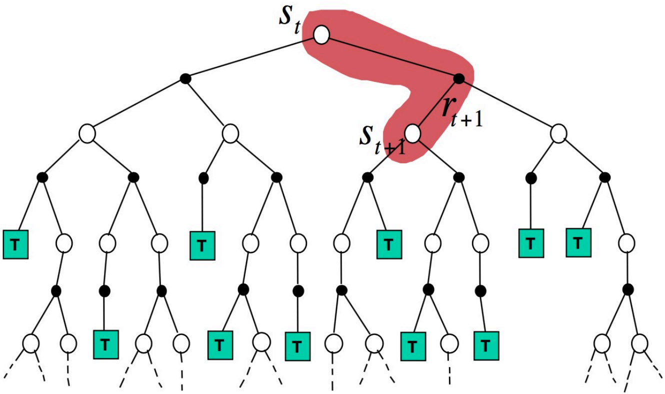


Table of Contents

1 回顾：Markov 决策过程

2 Policy Evaluation (策略评估)

3 Policy Iteration (策略迭代)

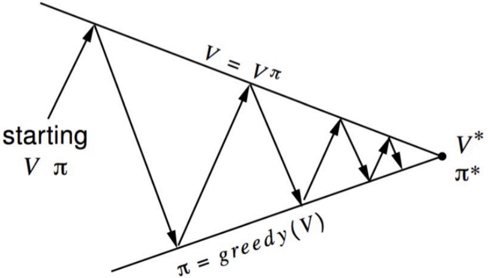
4 Value Iteration (值迭代)

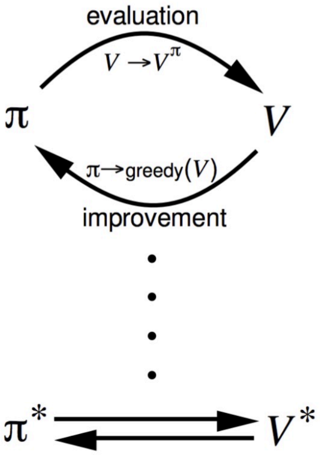
5 Model-free Prediction(无模型预测)

6 Model-free Control(无模型控制)

7 深度强化学习

Policy Iteration，策略迭代





Policy evaluation Estimate v_π

Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

Greedy policy improvement

策略迭代

- 有模型的策略迭代

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\, E[R_{t+1} + \gamma V(s')]$$

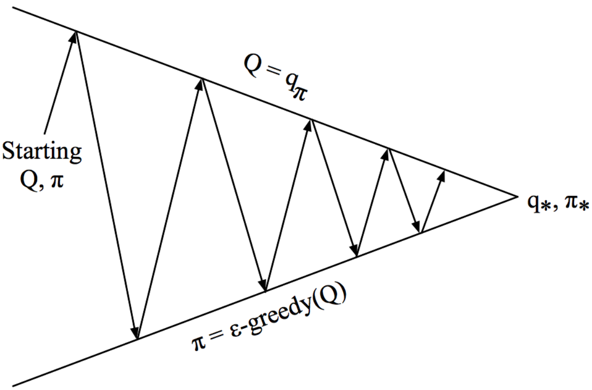
- 无模型的策略迭代

$$\pi(s) = \underset{a \in A}{\operatorname{argmax}}\, Q(s,a)$$

或 $\epsilon-$ 贪婪策略，ϵ 概率随机行动，$1-\epsilon$ 贪婪行动

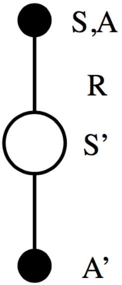
$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a = \operatorname{argmax}_{a \in A} Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

Monte-Carlo 策略迭代



- 策略评估: Monte-Carlo 策略评估
- 策略改进: $\epsilon-$ 贪婪策略

TD 控制: SARSA

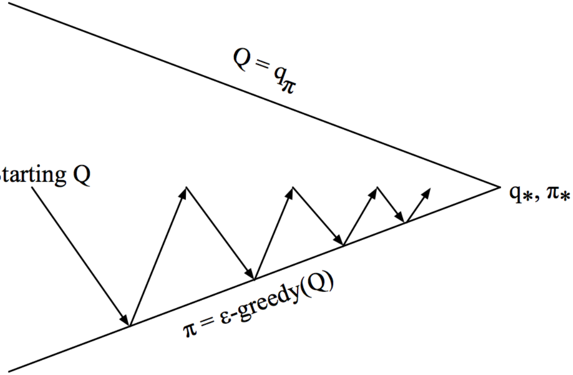


TD 预测:$V(S) \leftarrow V(S) + \alpha(R + \gamma V(S) - V(S))$

SARSA:$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$

其中 A' 是 $\epsilon-$ 贪婪策略，也是下时刻将实施的控制

On-Policy Control with SARSA



任意时刻

- 策略评估: SARSA, $Q \approx q_\pi$
- 策略改进: $\epsilon-$ 贪婪策略

SARSA: On-Policy Control

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

A' 的选择依据与 A 相同，都是 Agent 实施的控制策略（$\epsilon-$ 贪婪），On-Policy

Q-Learning: Off-Policy Control

A' 并不直接由 Agent 的 $\epsilon-$ 贪婪策略生成，采用贪婪策略

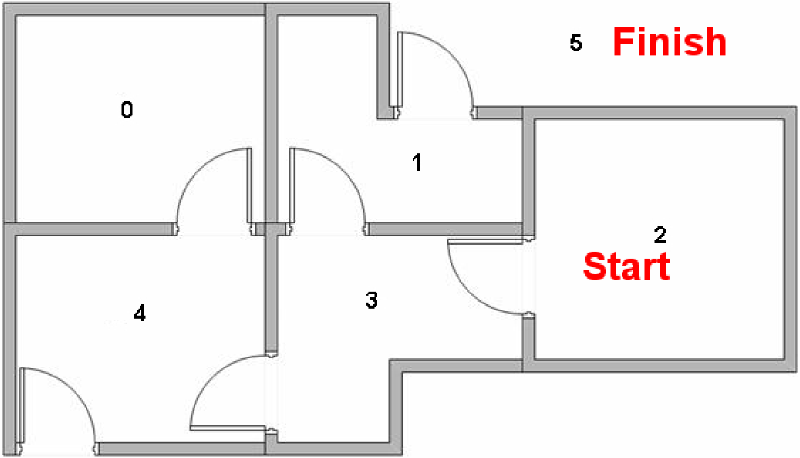
$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma \max_{a'} Q(S',a') - Q(S,A))$$

Q-Learning: Off-Policy Control

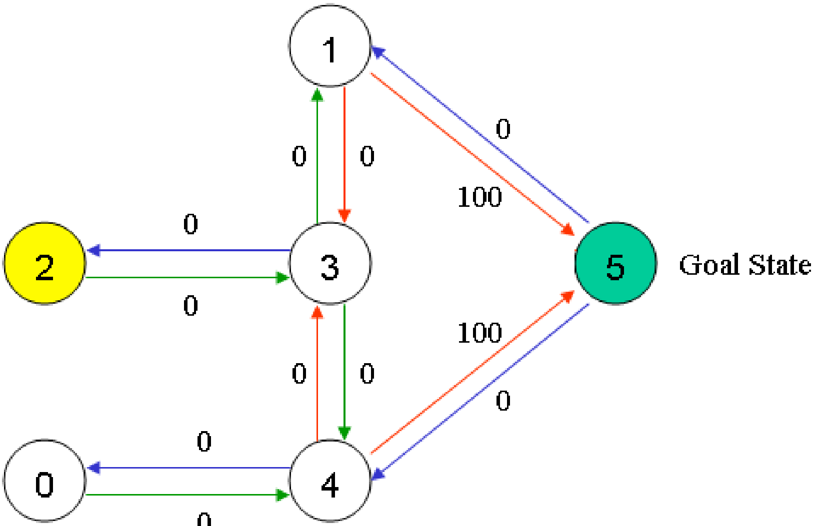
Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
        $S \leftarrow S'$;
    until S is terminal

A' 的选择依据与 A 不同，换言之，不从当前策略学习 Off-Policy

Example: Maze 1/3



Example: Maze 2/3



Example: Maze 3/3

-1 表示没有通路（不是数值）

$$
\begin{array}{cc}
 & \textbf{Action} \\
\textbf{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}
\end{array}
$$

$$
R = \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & -1
\end{bmatrix}
$$

Example: Maze Q-Learning

令 $\alpha = 1.,\ \gamma = 0.8.$

$$
\begin{aligned}
Q(S_t, A_t) \leftarrow & Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)) \\
= & Q(S_t, A_t) + R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t) \\
= & R_{t+1} + 0.8 \max_{a'} Q(S_{t+1}, a')
\end{aligned}
$$

Example: Maze Q-Learning, Episode 1

$$
R = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & -1 \end{bmatrix} \end{array}
\quad
Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}
$$

- 随机初始化 $S_1 = 1$, $A_1 \in \{3, 5\}$
- $\epsilon-$ 贪婪, $Q(1,3) = Q(1,5) = 0$, 随机选择 $A_1 = 5$
- 实施 A_1, 获得 $R(1,5) = 100$, 观测 $S_2 = 5$ 终止; $Q(5,\cdot) = 0$
- 更新 $Q(1,5) = R(1,5) + 0.8\max_{a'}[Q(5,a')] = 100$.

Example: Maze Q-Learning, Episode 2 - 1/2

$$R = \begin{array}{c|c} & \text{Action} \\ \text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & -1 \end{bmatrix} \end{array} \quad Q = \begin{array}{cc} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

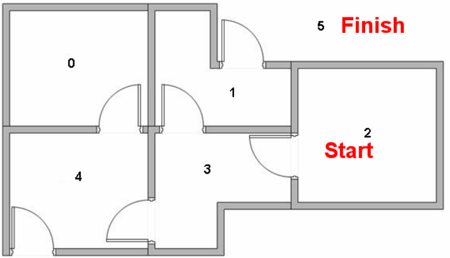
- 随机初始化 $S_1 = 3$, $A_1 \in \{1, 2, 4\}$
- $\epsilon-$ 贪婪, $Q(3,1) = Q(3,2) = Q(3,4) = 0$, 随机选择 $A_1 = 1$
- 实施 A_1, 获得 $R(3,1) = 0$, 观测 $S_2 = 1$;
- 更新 $Q(3,1) = R(3,1) + 0.8 \max_{a'}[Q(1,3), Q(1,5)] = 80$

Example: Maze Q-Learning, Episode 2 - 2/2

$$
R = \begin{array}{c|c}
 & \text{Action} \\
\text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & -1 \end{bmatrix}
\end{array}
\quad
Q = \begin{array}{cc}
 & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

- 继续 $S_2 = 1, A_2 \in \{3, 5\}$
- $\epsilon-$ 贪婪, $Q(1,3) = 0, Q(1,5) = 100$, 贪婪选择 $A_2 = 5$
- 实施 A_2, 获得 $R(1,5) = 100$, 观测 $S_3 = 5$, 终止; $Q(5,\cdot) = 0$
- 更新 $Q(1,5) = R(1,5) + 0.8\max_{a'}[Q(5,a')] = 100$

Example: Maze Q-Learning Result 1/2



$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 0 \end{array}\right] \end{array}$$

Example: Maze Q-Learning Result 2/2

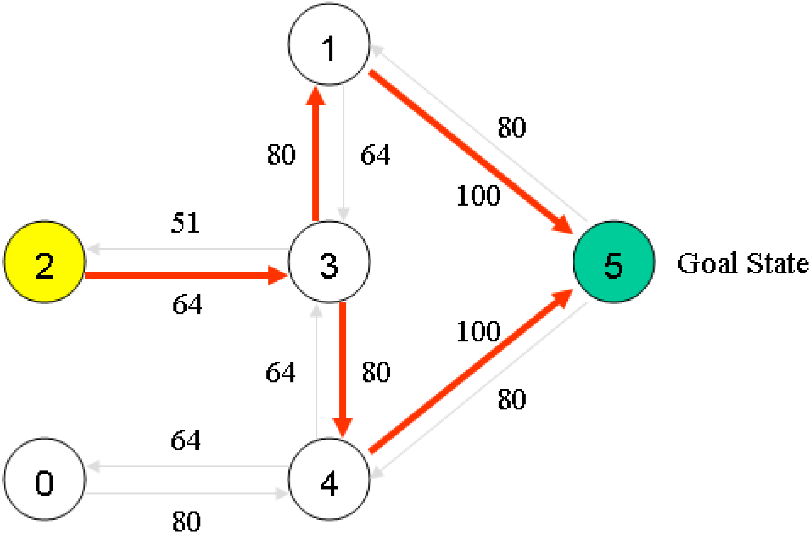
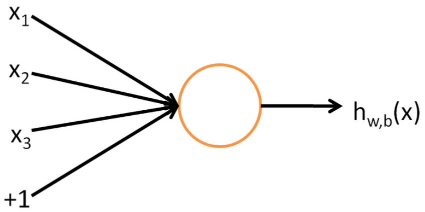


Table of Contents

1. 回顾：Markov 决策过程
2. Policy Evaluation (策略评估)
3. Policy Iteration (策略迭代)
4. Value Iteration (值迭代)
5. Model-free Prediction(无模型预测)
6. Model-free Control(无模型控制)
7. 深度强化学习

最简单的神经网络：神经元



- 神经元 “Neuron” 以输入 x 和截距 $+1$ 为输入，输出

$$h(x;W,b)=f(W^Tx+b)=f(\sum_{i=1}^{3}W_ix_i+b)$$

其中 f 被称为激活函数。

- 先线性变换，再经过激活函数

线性激活函数 Linear 1/4

线性激活函数

$$f(z) = z, \quad h(x; W, b) = W^T x + b$$

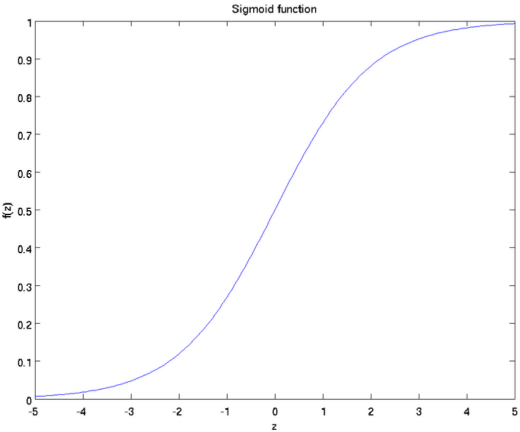
- 给定训练集 $X = \{x^{(1)}, \ldots, x^{(m)}\}, Y = \{y^{(1)}, \ldots, y^{(m)}\}$
- 寻找参数 $\theta = (W, b)$，最小化误差的平均值

$$J(X, Y; \theta) = \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}; \theta) - y^{(i)})^2$$

- 梯度下降法 $\theta_j \leftarrow \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$ (向梯度相反方向移动)

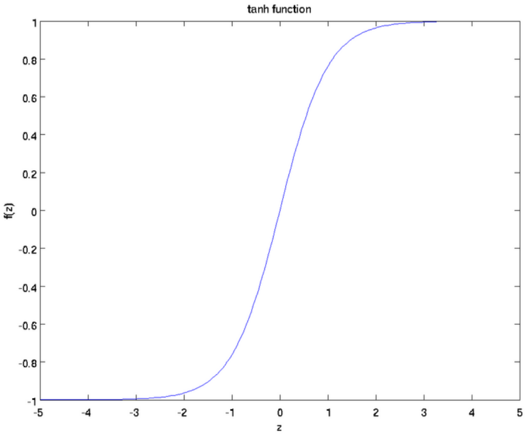
$$\begin{aligned} \frac{\partial J}{\partial W_j} &= \frac{1}{m} \sum_{i=1}^{m} 2(h(x^{(i)}; \theta) - y^{(i)}) \frac{\partial}{\partial W_j} (h(x^{(i)}; \theta) - y^{(i)}) \\ &= \frac{2}{m} \sum_{i=1}^{m} (h(x^{(i)}; \theta) - y^{(i)}) x_j^{(i)} \end{aligned}$$

非线性激活函数 Sigmoid (S 型生长函数) 2/4



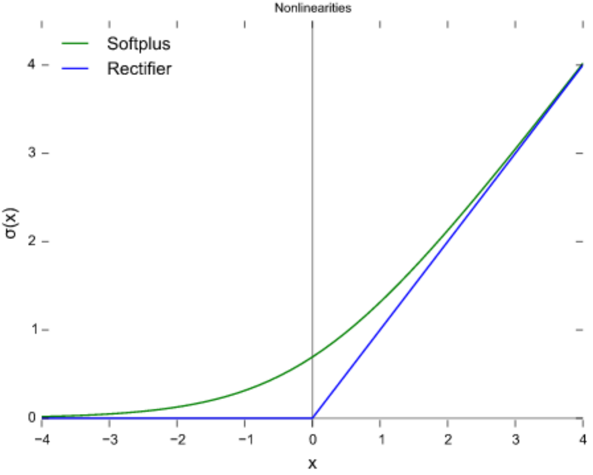
$$f(z) = \frac{1}{1+e^{-z}}, \quad h(x; W, b) = f(W^T x + b)$$

非线性激活函数 Tanh (双曲正切函数)3/4



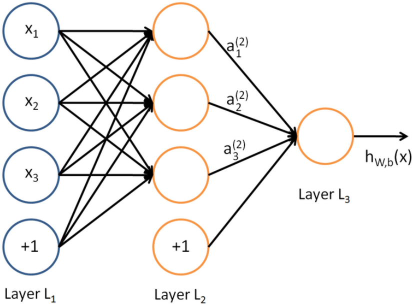
$$f(z) = \frac{e^z - e^{-1}}{e^z + e^{-z}}, \quad h(x; W, b) = f(W^T x + b)$$

非线性激活函数 Rectifier 4/4



$$f(z) = \max(0, z),\ f(z) = \ln(1 + e^x), \quad h(x; W, b) = f(W^T x + b)$$

多层神经网络



L_1 为输入，L_2 为隐层，L_3 为输出

$$a_1^{(2)} = f_1(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)})$$

$$a_2^{(2)} = f_1(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)})$$

$$a_3^{(2)} = f_1(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)})$$

$$h(x; W, b) = a_1^{(3)} = f_2(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)})$$

前向传播和反向传播

上述神经网络的“前向传播”可记为向量值形式:

$$a^{(2)} = f_1(W^{(1)}x + b^{(1)})$$
$$h(x; W, b) = a^{(3)} = f_2(W^{(2)}a^{(2)} + b^{(2)})$$

可通过“反向传播”拟合参数

$$W_{ij}^{(l)} \leftarrow W_{ij}^{l} - \alpha \frac{\partial J}{\partial W_{ij}^{l}}$$
$$b_i^{(l)} \leftarrow b_i^{l} - \alpha \frac{\partial J}{\partial b_i^{l}}$$

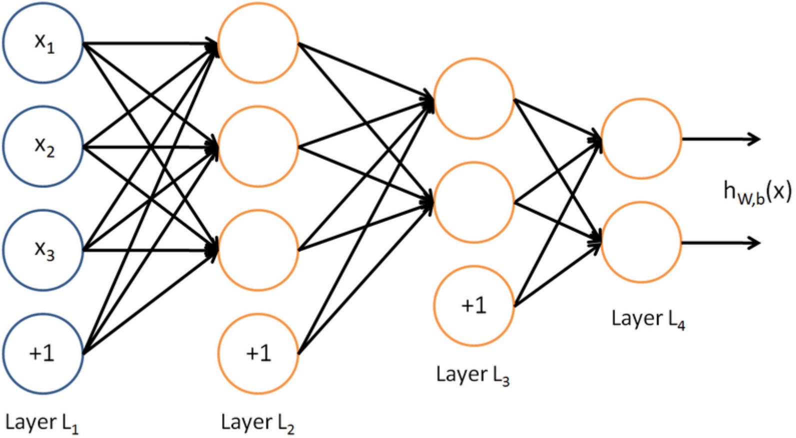
多层神经网络的特性

定理 1 (Cybenko1989)

给定 $I_m = [0,1]^m$ *上任意连续函数* g *和* $\epsilon > 0$，*存在整数* N *和参数* W, b，*使得，有* N *个隐层神经元，以 sigmoid 为激活函数的多层神经网络* h，

$$\|h(x; W, b) - g(x)\| < \epsilon, \ \forall x \in I_m.$$

高维输出，多层网络

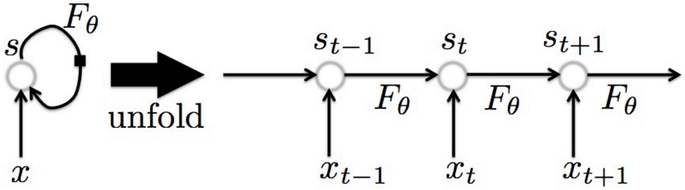


Recurrent Neural Network (循环神经网络)

状态 s_t 和输入 x_t

$$s_t = F(s_{t-1}, x_t; \theta)$$

得到更“深”的神经网络 $s_t = G_t(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$



$$s_t = G_t(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$$

- 传统神经网络：层与层之间是全连接的，每层之间的节点无连接
- 循环神经网络：对前面的信息进行记忆并应用于当前输出的计算中

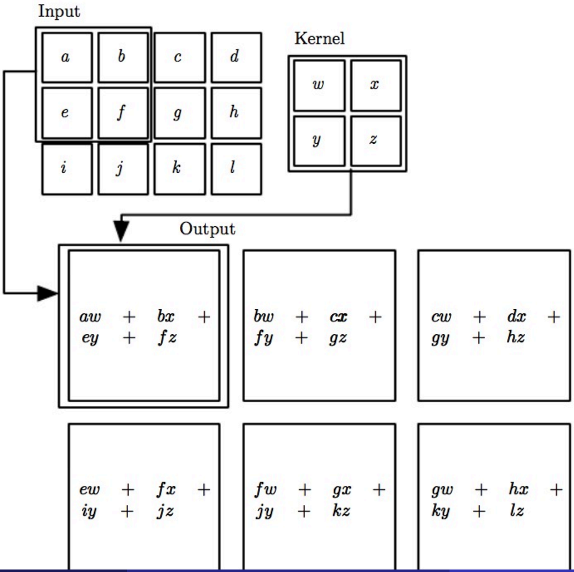
Convolution (卷积) 1/2

卷积 “Convolution” 是对状态的一种平滑估计

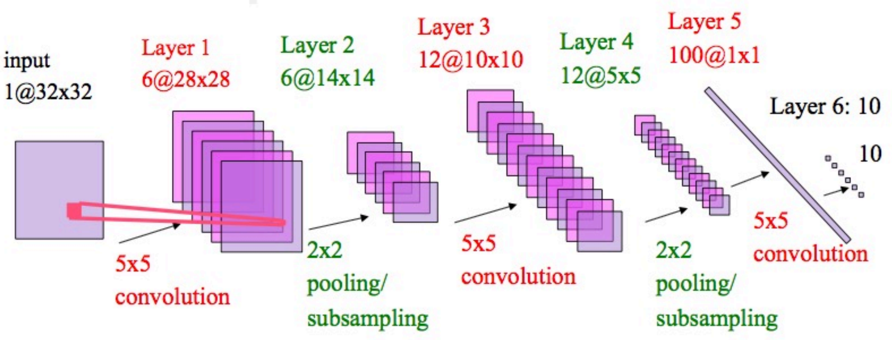
$$s(t) = \int x(\tau)\omega(t-\tau)d\tau$$

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

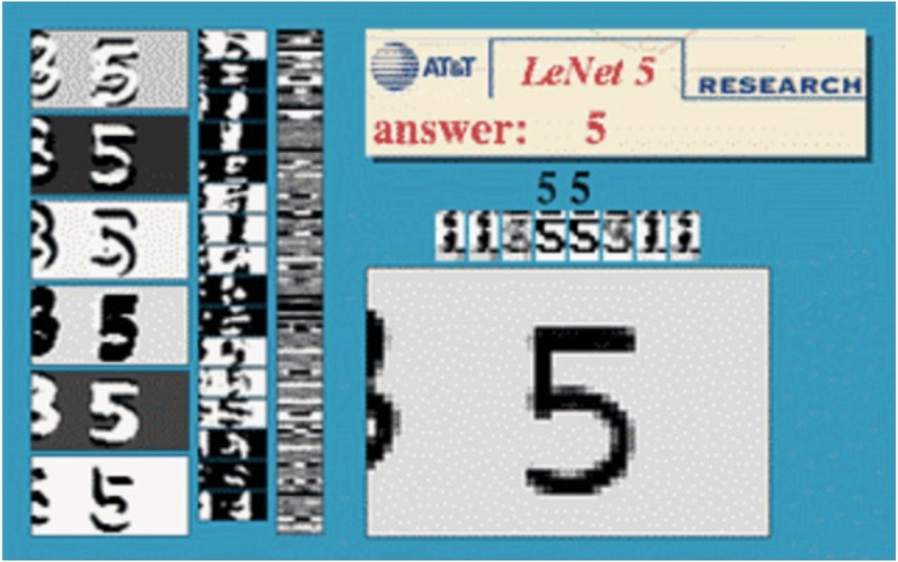
Convolution (卷积) 2/2



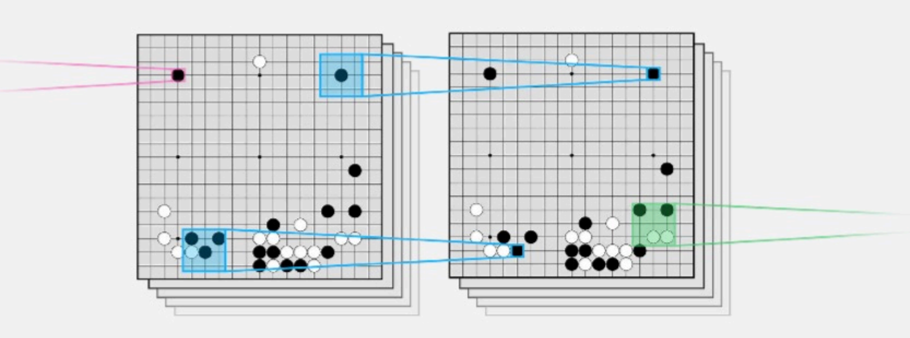
Convolutional Neural Network (卷积神经网络) LeNet



LeNet 手写识别

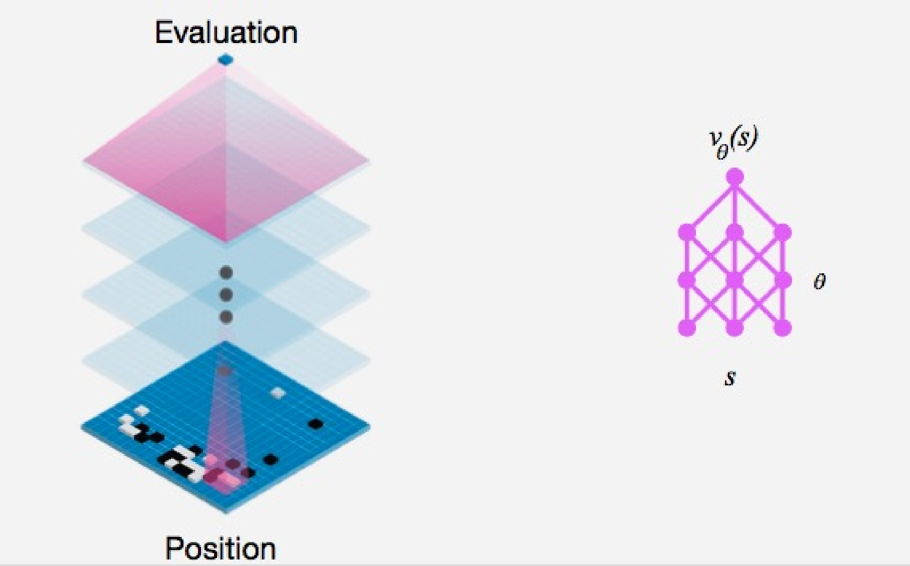


围棋的卷积神经网络

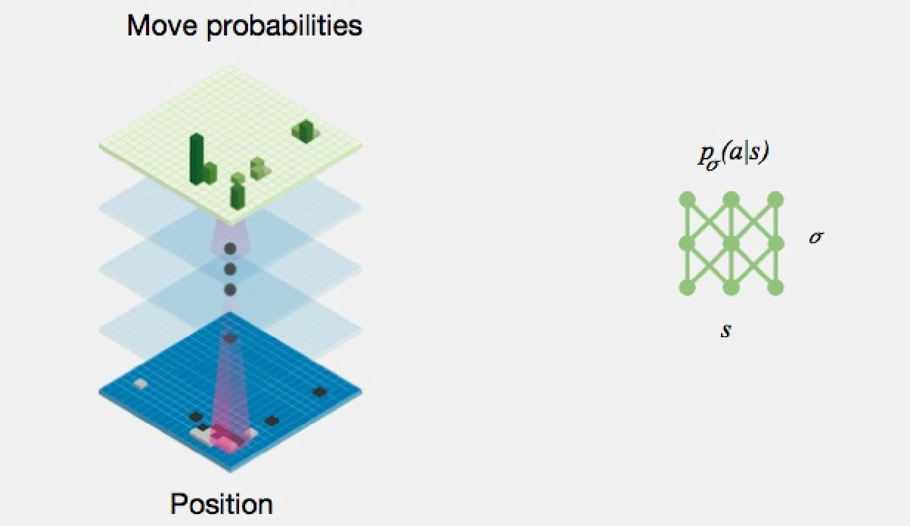


- Input: 30M positions from human expert games
- Output: Optimal control/equlibrium of game

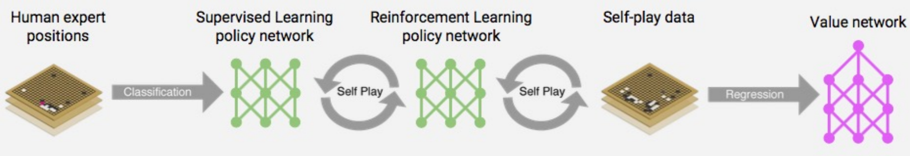
Value Network (价值网络)



Policy Network (策略网络)



AlphaGo Training Pipeline



策略网络的监督学习

- Policy network: 12 layer convolutional neural network
- Training data: 30M positions from human expert games
- Training algorithm: Maximize likelihood by stochastic gradient descent (随机梯度下降法)

$$\Delta\sigma \propto \frac{\partial \log p(a|s;\sigma)}{\partial \sigma}$$

- Training time: 4 weeks on 50 GPUs using Google Cloud
- Results: 57% accuracy on test data

Reinforcement Learning of Policy networks (策略网络的强化学习)

- Policy network: 12 layer convolutional neural network
- Training data: Games of self-play between policy network
- Training algorithm: Maximize wins z by policy gradient reinforcement learning

$$\Delta\sigma \propto \frac{\partial \log p(a|s;\sigma)}{\partial\sigma} z$$

- Training time: 1 week on 50 GPUs using Google Cloud
- Results: 80% vs supervised learning. About network $\tilde{3}$ amateur dan.

价值网络的强化学习

- Value network: 12 layer convolutional neural network
- Training data: 30 million games of self-play
- Training algorithm: minimize MSE by stochastic gradient descent

$$\Delta\theta \propto \frac{\partial v(s;\theta)}{\partial \theta}$$

- Training time: 1 week on 50 GPUs using Google Cloud
- Results: 4:1 围棋时间冠军-李世石