第四部分：完全可观察环境中的概率规划系统

章宗长

2021年5月12日

内容安排

- 规划
- 马尔科夫决策过程
- 精确动态规划
- 近似动态规划
- **在线规划**
- 直接策略搜索

在线规划

- **前向搜索、分支限界搜索、稀疏采样**
- 蒙特卡洛树搜索
- 蒙特卡洛树搜索：应用案例

在线方法

- 离线方法：在按策略执行行动之前，离线计算整个状态空间上的策略

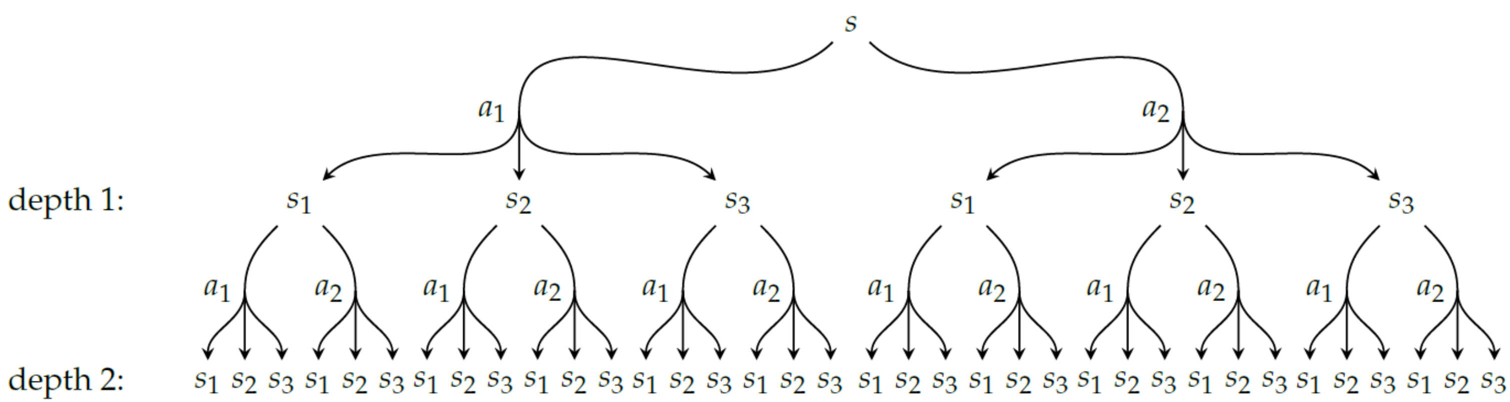
策略构建（C）	策略执行(E)

- 在线方法：把计算限制在从当前状态**可达**的状态上
  - 可达状态空间比整个状态空间小很多
  - 显著减少近似最优行动选择所需的存储空间和计算时间

C	E	C	E	C	E	C	E	C	E

前向搜索

- 简单的在线行动选择：从某一初始状态s开始，向前看直到某一深度d
- 例子：有3个状态、2个行动的MDP问题的深度为2的搜索树



深度为0的结点数：1

深度为1的结点数：$|\mathcal{S}| \times |\mathcal{A}|$

深度为2的结点数：$(|\mathcal{S}| \times |\mathcal{A}|)^2$

$\ldots$

深度为d的结点数：$(|\mathcal{S}| \times |\mathcal{A}|)^d$

- 计算复杂度
  - $O\left((|\mathcal{S}| \times |\mathcal{A}|)^d\right)$

前向搜索（续）

Algorithm 4.6 Forward search

function $\textsc{SelectAction}(s, d)$
  if $d = 0$
    return $(\textsc{nil}, 0)$
  $(a^*, v^*) \leftarrow (\textsc{nil}, -\infty)$
  for $a \in \boxed{A(s)}$
    $v \leftarrow R(s, a)$
    for $s' \in \boxed{S(s, a)}$
      $\boxed{(a', v') \leftarrow \textsc{SelectAction}(s', d-1)}$
      $v \leftarrow v + \gamma T(s' \mid s, a) v'$
    if $v > v^*$
      $(a^*, v^*) \leftarrow (a, v)$
  return $\boxed{(a^*, v^*)}$

在状态s可获得的行动集合

从s执行a可立即到达的状态的集合

深度优先，递归调用自身
直至到达指定深度

返回最优行动a^*和它的值v^*

分支限界搜索

- 前向搜索的一种扩展：使用最优值函数的上界和下界来裁剪搜索树

Algorithm 4.7 Branch-and-bound search

```
function SELECTACTION(s, d)
    if d = 0
        return (NIL, U̲(s))
    (a*, v*) ← (NIL, −∞)
    for a ∈ A(s)
        if Q̄(s, a) < v*
            return (a*, v*)
        v ← R(s, a)
        for s′ ∈ S(s, a)
            (a′, v′) ← SELECTACTION(s′, d − 1)
            v ← v + γ T(s′ | s, a) v′
        if v > v*
            (a*, v*) ← (a, v)
    return (a*, v*)
```

用先验知识快速计算出最优值函数的下界$\underline{U}(s)$和最优行动值函数的上界$\bar{Q}(s,a)$

使用了下界

行动应该按上界降序排列：如果行动a_i在a_j之前被评价，则$\bar{Q}(s,a_i) \geq \bar{Q}(s,a_j)$

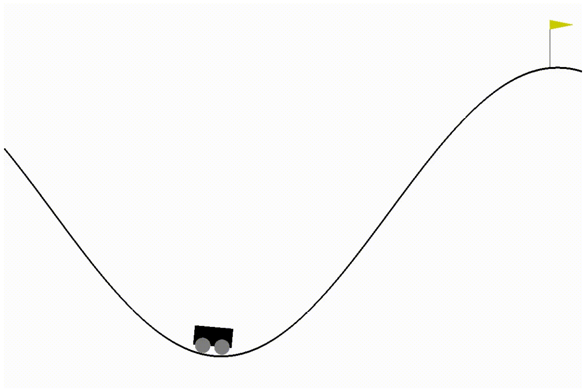
检查是否裁剪分支(s,a)

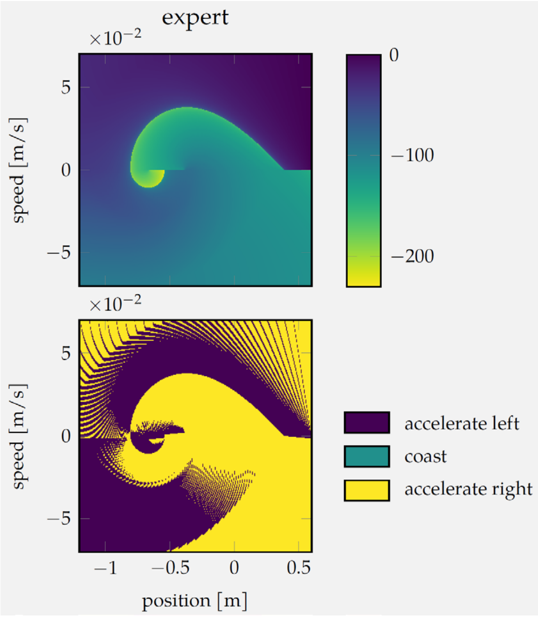
- 上下界之差越小，可裁剪的搜索区域越多，计算时间越少
- 最坏时间复杂度与前向搜索相同

返回最优行动a^*和最优值函数的下界v^*

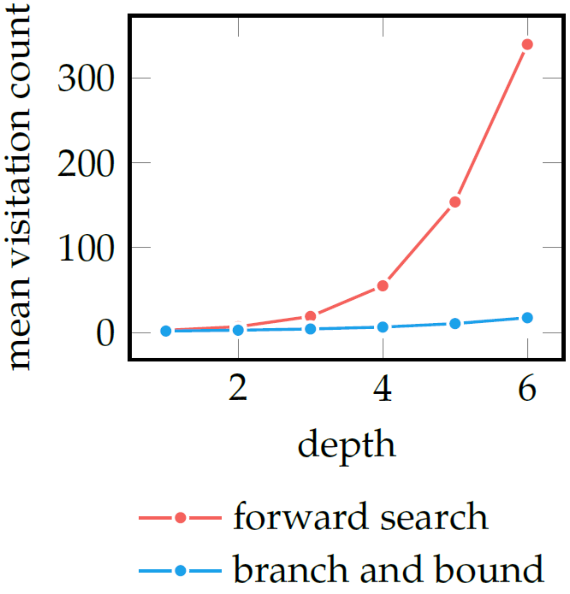
分支限界搜索（续）

- 例子：小车上山
  - 最优值函数的下界：采用总是朝运动方向加速策略的值函数
  - 最优值函数的上界：如果没有右侧的小山，向右加速到达目标的期望回报





最优状态值函数与最优策略



平均访问的结点数：前向搜索 vs. 分支限界

稀疏采样

- 采样方法：稀疏采样
  - 避免前向搜索和分支限界搜索在最坏情况下的指数复杂度
  - 不能保证得到最优行动，但在大多数时候能得到近似最优行动

Algorithm 4.8 Sparse sampling

```
function SelectAction(s, d)
    if d = 0
        return (NIL, 0)
    (a*, v*) ← (NIL, −∞)
    for a ∈ A(s)
        v ← 0
        for i ← 1 to n
            (s′, r) ~ G(s, a)
            (a′, v′) ← SelectAction(s′, d − 1)
            v ← v + (r + γv′)/n
        if v > v*
            (a*, v*) ← (a, v)
    return (a*, v*)
```

采样n次，而不是遍历$S(s,a)$的所有状态

对由每个样本得到的$r+\gamma v'$求平均来估计$Q(s,a)$

- 产生式模型G
  - 表示状态转移和奖赏的所有信息
  - 产生下一个状态s'和奖赏r的样本

相比显式地表示概率，使用一个产生式模型往往更容易实现从一个复杂的多维分布中抽取随机样本

- 计算复杂度
  - $O\left((n \times |\mathcal{A}|)^d\right)$

依赖于d
不依赖于$|\mathcal{S}|$

在线规划

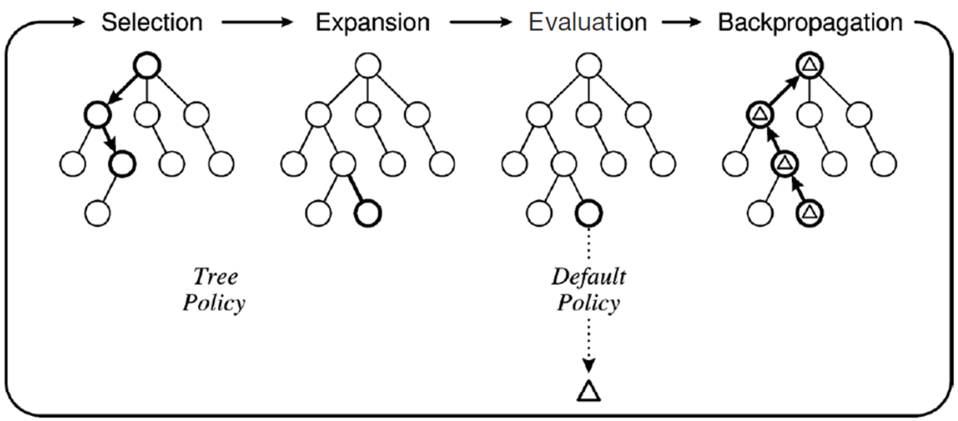
- 前向搜索、分支限界搜索、稀疏采样
- **蒙特卡洛树搜索**
- 蒙特卡洛树搜索：应用案例

蒙特卡洛树搜索

- 蒙特卡洛树搜索（Monte Carlo Tree Search, MCTS）
  - 最成功的基于采样的在线方法之一
  - 使用产生式模型
  - 计算复杂度不随深度指数增长
  - 从当前状态做很多仿真，用仿真的结果更新值函数$Q(s,a)$的估计

以增量式、非对称方式构建一棵搜索树T

- 4个阶段
  - 选择（selection）
  - 扩展（expansion）
  - Rollout评价（evaluation）
  - 反向更新（backpropagation）



- 树策略（tree policy）：指导搜索树T中的行动选择
- 默认策略（default policy）：指导从新扩展结点到指定深度的行动选择

蒙特卡洛树搜索（续）

选择

- 从根结点开始，在搜索树T中前向搜索，直到到达一个不在T中的结点s
- 在搜索过程中，选择能最大化下式的行动分支：

上置信界树（Upper Confidence Bound for Tree, UCT）

$$Q(s,a) + c\boxed{\sqrt{\frac{\log N(s)}{N(s,a)}}} \qquad N(s) = \sum_a N(s,a)$$

参数c用来控制对探索的喜好程度

探索奖金（鼓励选择那些探索次数不多的行动）
若$N(s,a)=0$，则奖金无穷大

扩展

- 遍历在s可使用的行动
- 基于专家先验知识，把$N(s,a)$和$Q(s,a)$初始化为$N_0(s,a)$和$Q_0(s,a)$
- 若先验知识不可获得，则把$N(s,a)$和$Q(s,a)$均初始化为0
- 把结点s加入到搜索树T中

蒙特卡洛树搜索（续）

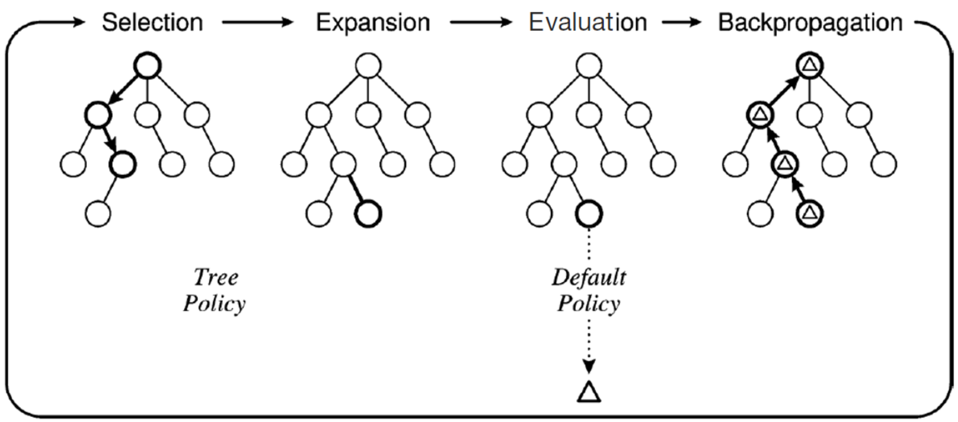
- 评价（算法4.10，也称Rollout评价）
  - 使用某个默认策略来选择行动，直至到达指定深度
  - 默认策略也称滚轮策略（rollout policy）
  - 典型地，默认策略是随机的，即行动通过采样得到：$a \sim \pi_0(\cdot \mid s)$
  - 默认策略为专家提供了一种方式使得搜索偏向有希望的区域

Algorithm 4.10 Rollout evaluation

function $\text{Rollout}(s, d, \pi_0)$
    if $d = 0$
        return 0
    $a \sim \pi_0(\cdot \mid s)$
    $(s', r) \sim G(s, a)$
    return $r + \gamma \text{Rollout}(s', d - 1, \pi_0)$

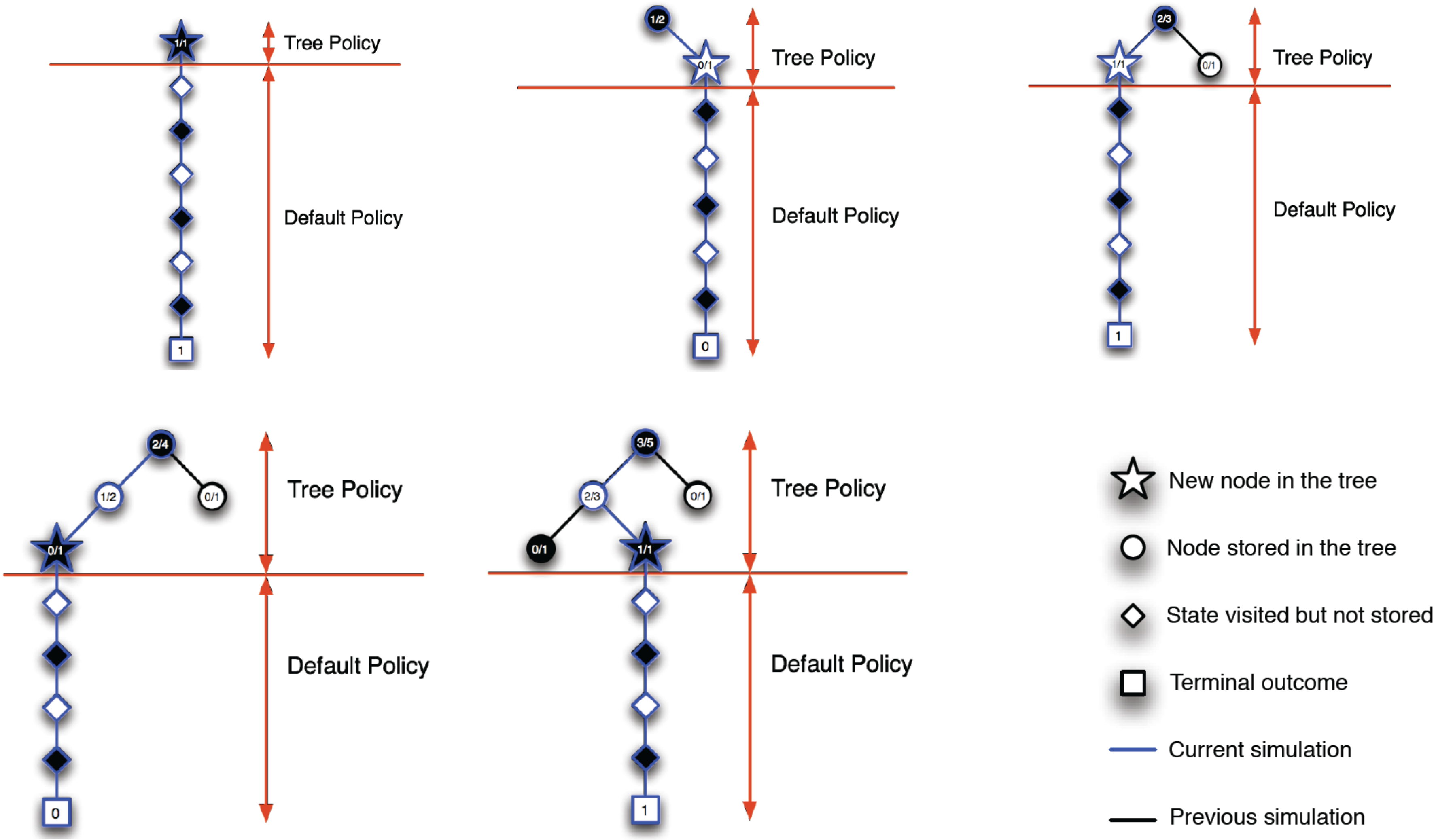
蒙特卡洛树搜索（续）

- 反向更新
  - 用评价的结果更新新扩展结点s的$N(s,a)$和$Q(s,a)$
  - 反向更新根结点到新扩展结点s沿线所有结点的$N(s,a)$和$Q(s,a)$



- 循环执行上述4个阶段，直到满足某一终止条件，然后执行最大化当前状态处Q值的行动
- 执行后，重新运行蒙特卡洛树搜索算法来选择下一个行动

蒙特卡洛树搜索（续）



1/1
Tree Policy
Default Policy
1
1/2
0/1
Tree Policy
Default Policy
0
2/3
1/1
0/1
Tree Policy
Default Policy
1
2/4
1/2
0/1
0/1
Tree Policy
Default Policy
0
3/5
2/3
0/1
0/1
1/1
Tree Policy
Default Policy
1
New node in the tree
Node stored in the tree
State visited but not stored
Terminal outcome
Current simulation
Previous simulation

蒙特卡洛树搜索（续）

Algorithm 4.9 Monte Carlo tree search

function SELECTACTION(s, d)
  loop
    SIMULATE(s, d, π_0)
  return $\arg\max_a Q(s, a)$
function SIMULATE(s, d, π_0)
  if $d = 0$
    return 0

扩展

  if $s \notin T$
    for $a \in A(s)$
      $(N(s,a), Q(s,a)) \leftarrow (N_0(s,a), Q_0(s,a))$
    $T = T \cup \{s\}$

评价

    return ROLLOUT(s, d, π_0)

选择

  $a \leftarrow \arg\max_{a \in A(s)} Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$
  $(s', r) \sim G(s, a)$

反向更新

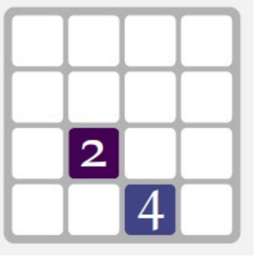
  $q \leftarrow r + \gamma$ SIMULATE$(s', d-1, \pi_0)$
  $N(s,a) \leftarrow N(s,a) + 1$
  $Q(s,a) \leftarrow Q(s,a) + \frac{q - Q(s,a)}{N(s,a)}$
  return q

在线规划

- 前向搜索、分支限界搜索、稀疏采样
- 蒙特卡洛树搜索
- **蒙特卡洛树搜索：应用案例**

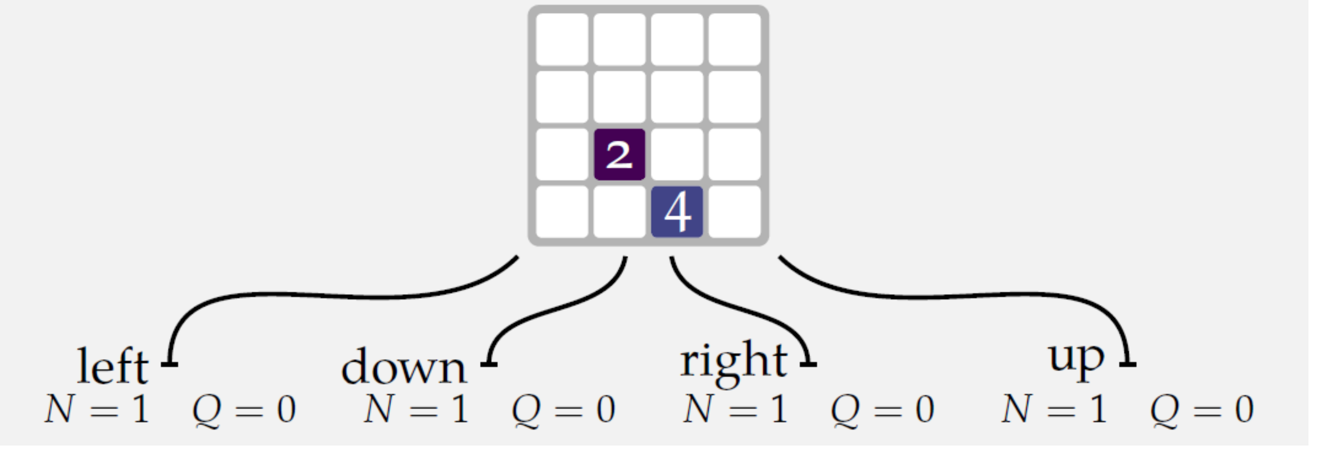
2048

- 用蒙特卡洛树搜索来玩2048游戏
  - 最大搜索深度$d = 10$
  - 探索参数$c = 100$
  - Rollout策略：均匀随机



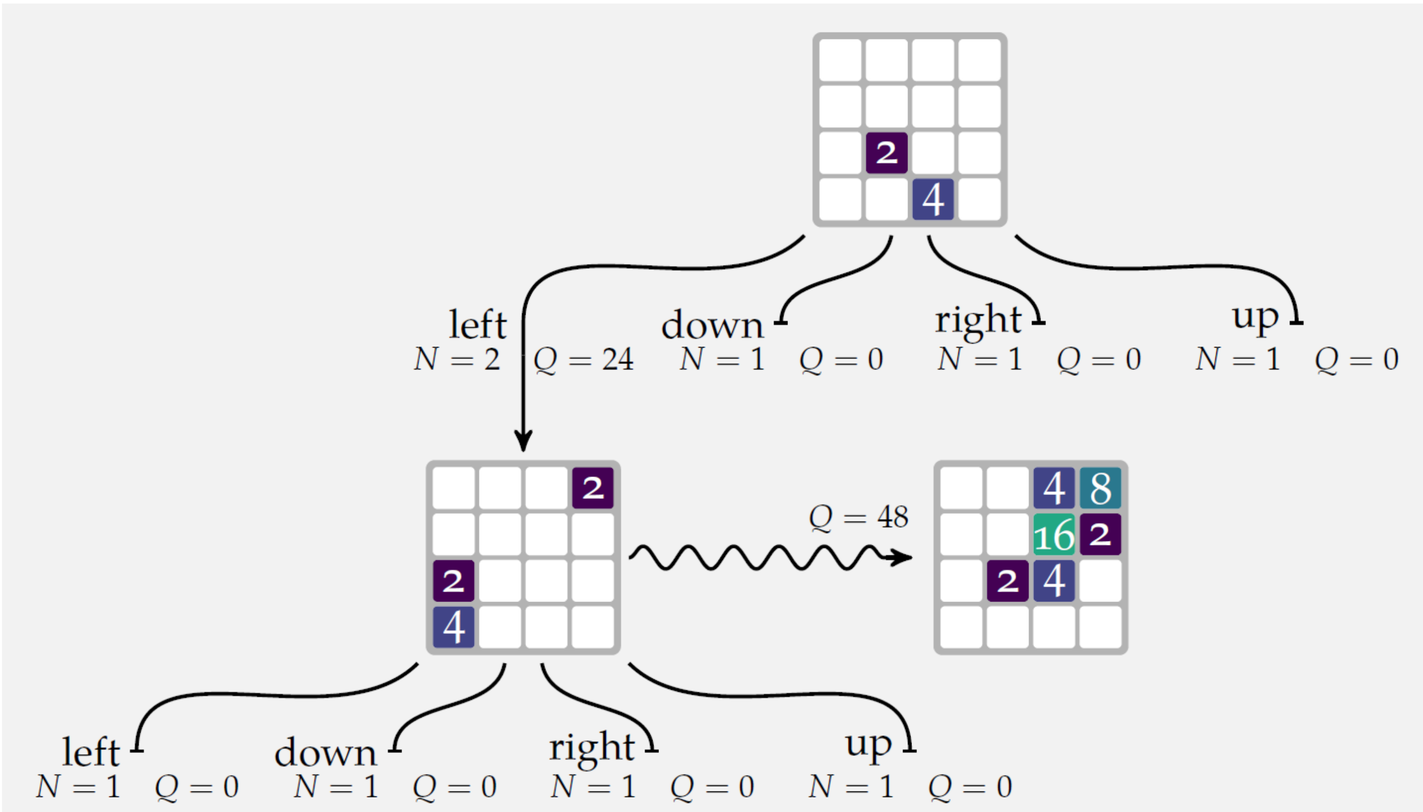
初始状态

- 扩展初始状态，初始化$N(s,a)$和$Q(s,a)$



2048（续）

- 选择行动left，采样一个新的后继状态，扩展该状态，从该状态开始模拟，用仿真的结果反向更新



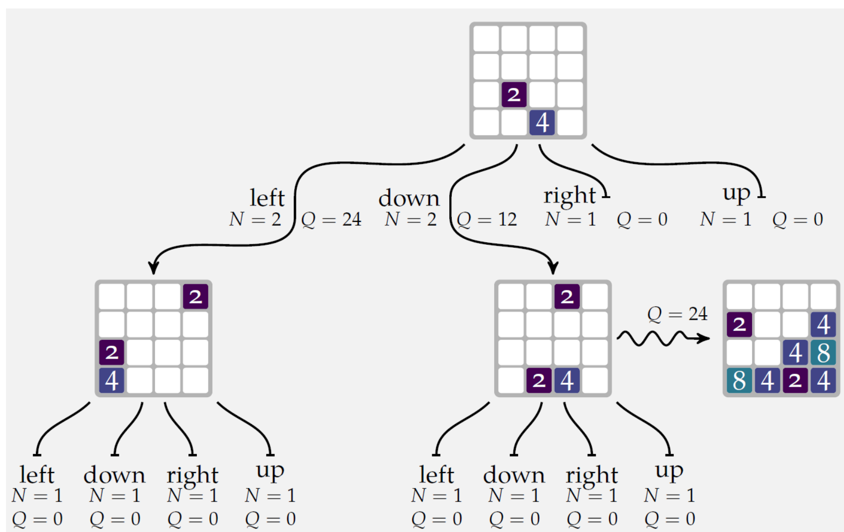
2048（续）

- 选择行动down

$$Q(s_0,\texttt{down}) + c\sqrt{\frac{\log N(s_0)}{N(s_0,\texttt{down})}} = 0 + 100\sqrt{\frac{\log 5}{1}} = 126.864$$

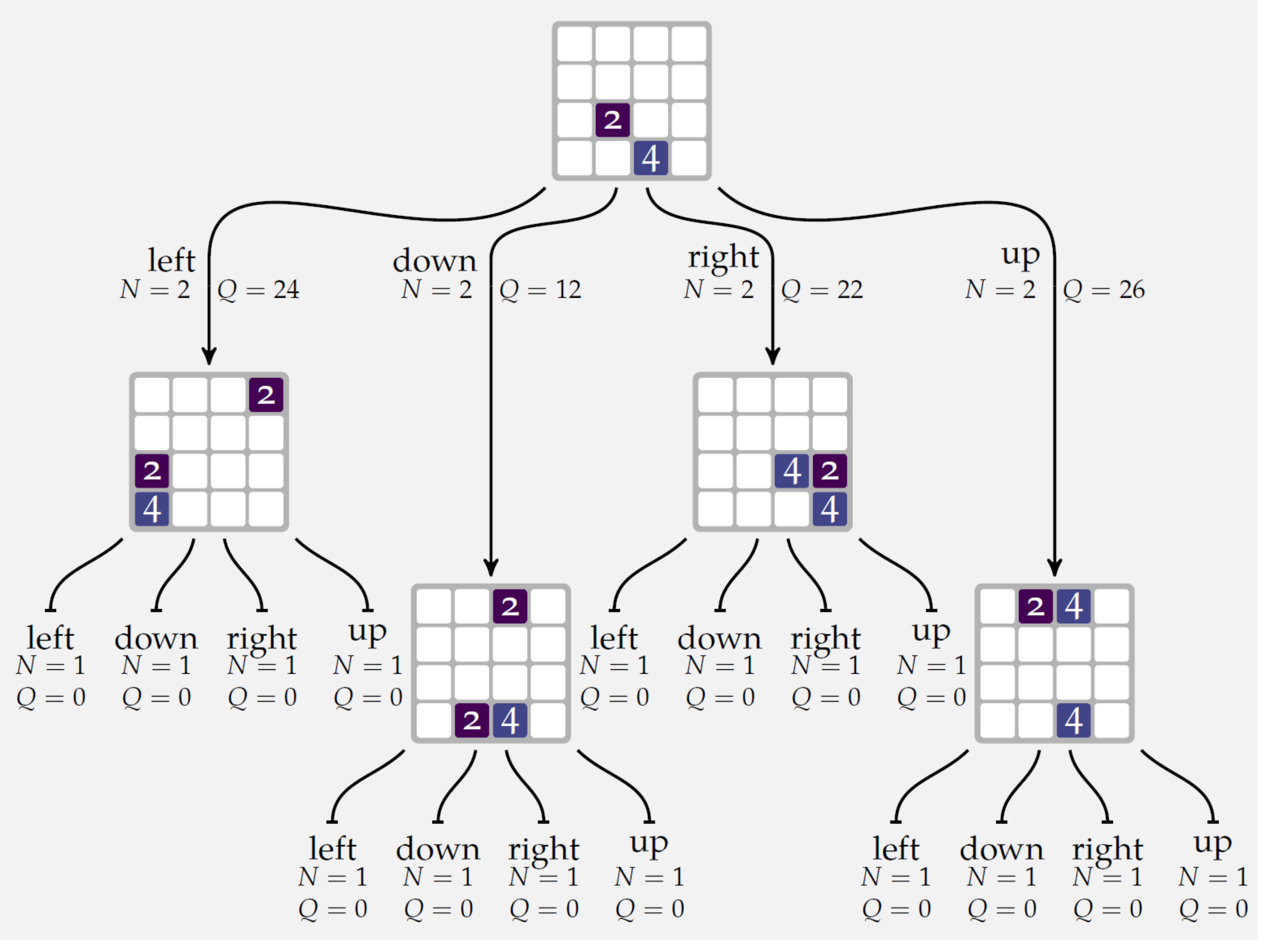
$$Q(s_0,\texttt{left}) + c\sqrt{\frac{\log N(s_0)}{N(s_0,\texttt{left})}} = 24 + 100\sqrt{\frac{\log 5}{2}} = 113.706$$

- 采样一个新的后继状态，扩展该状态，从该状态开始仿真，用仿真的结果反向更新



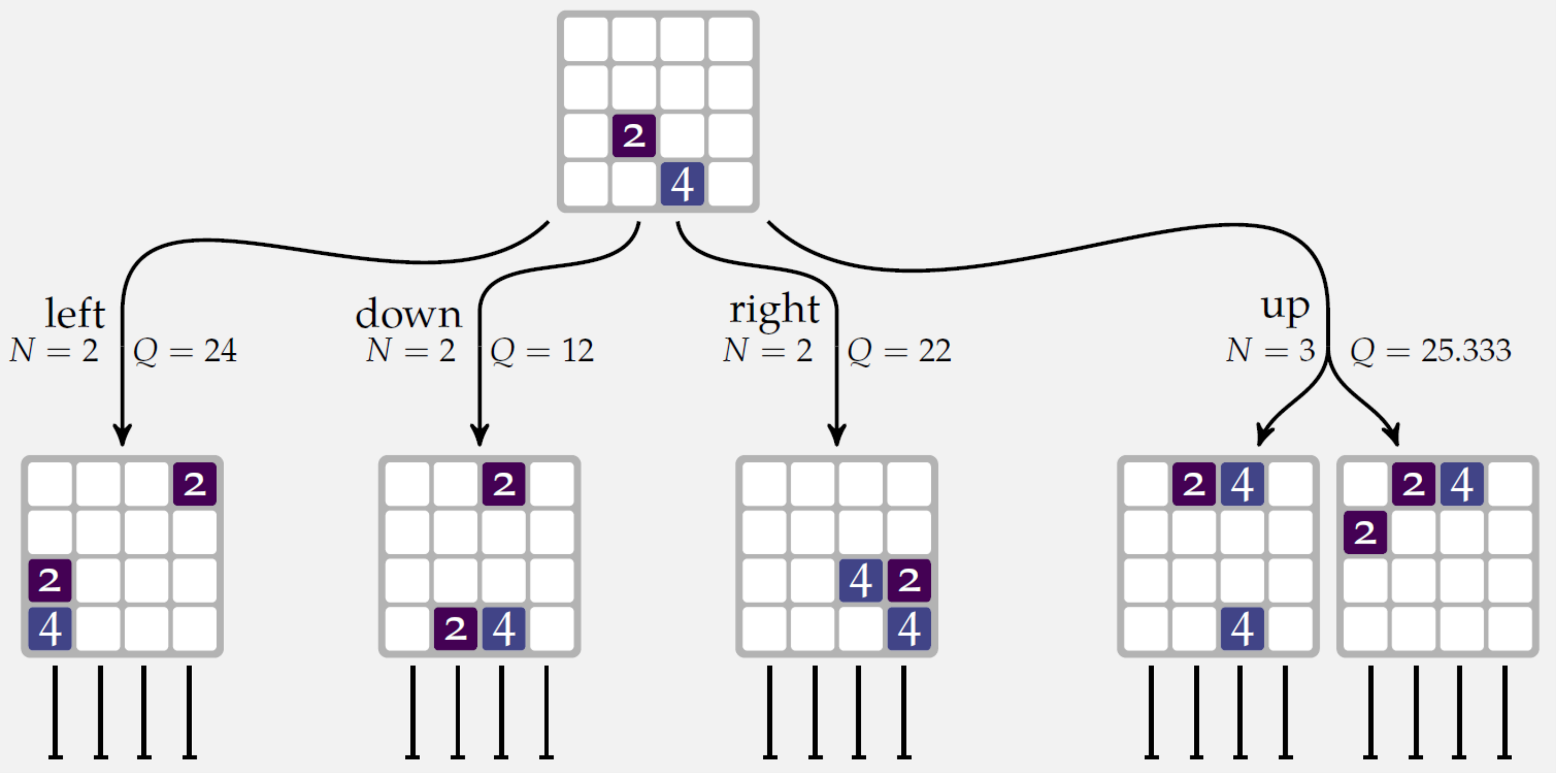
2048（续）

- 接下来两轮，选择行动right和up



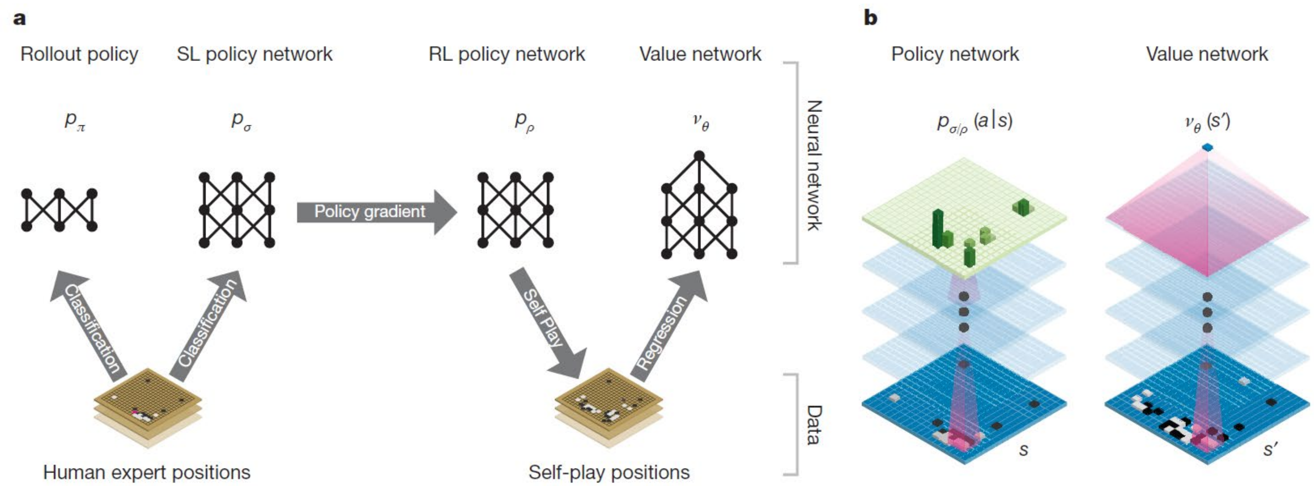
2048（续）

- 在第5轮，行动up有最高值。选择该行动，产生一个新的后继状态

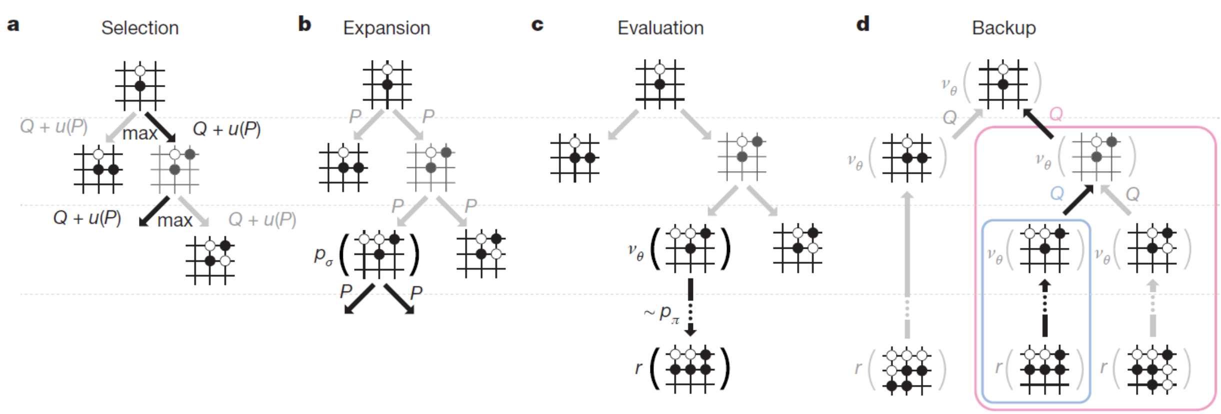


AlphaGo

AlphaGo的训练过程及使用到的几个不同的网络



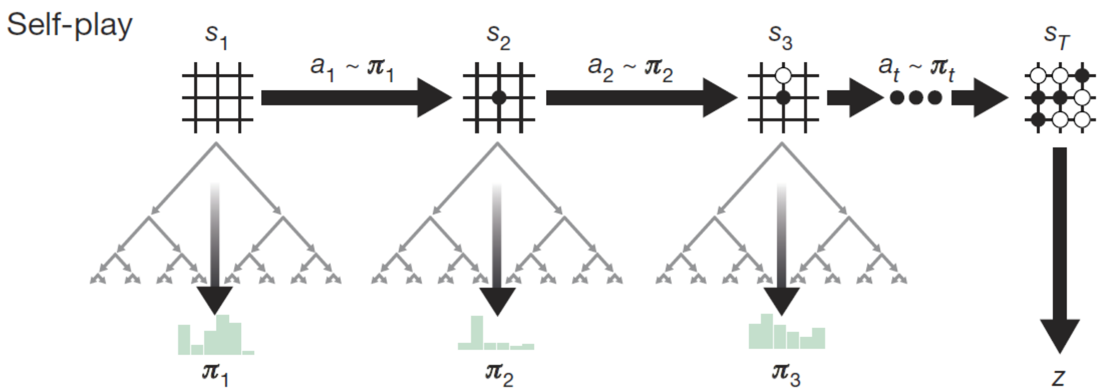
AlphaGo中的蒙特卡洛树搜索

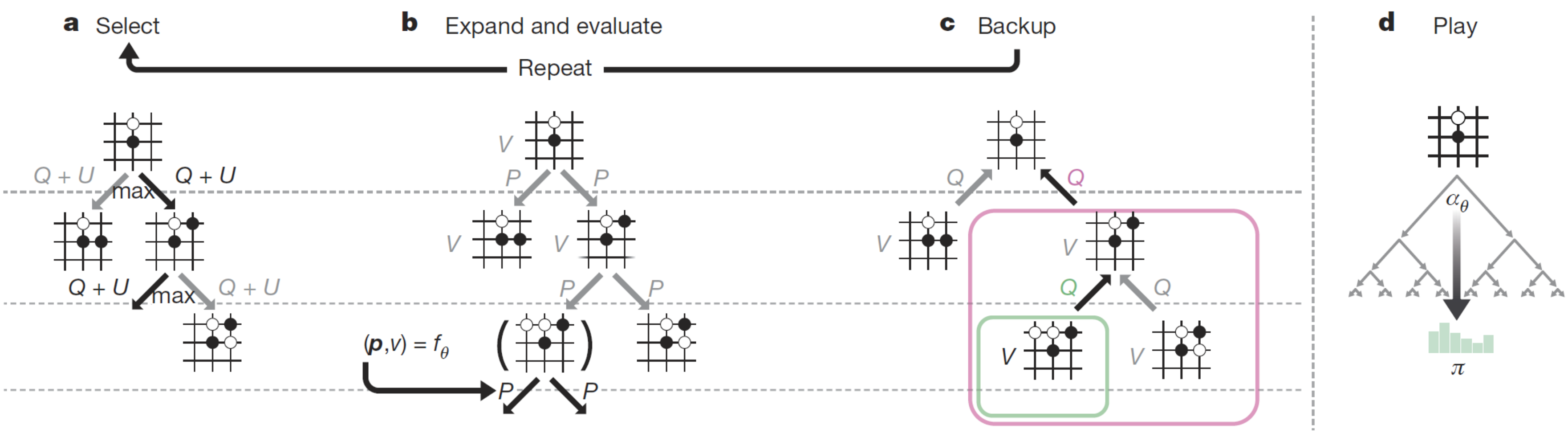


Silver D, Huang A, Maddison C J, et al. **Mastering the game of Go with deep neural networks and tree search**. Nature, 2016, 529(7587): 484-489

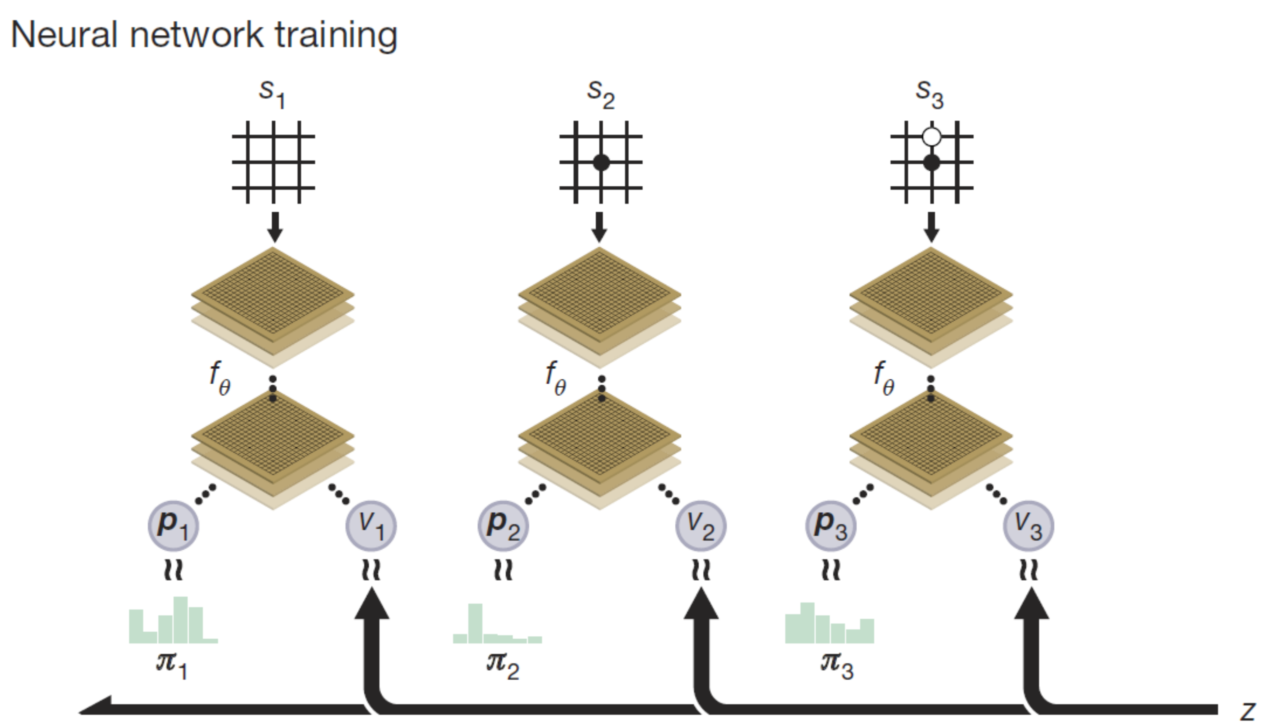
AlphaGo Zero

- 仅需知道围棋的基本规则就能无师自通



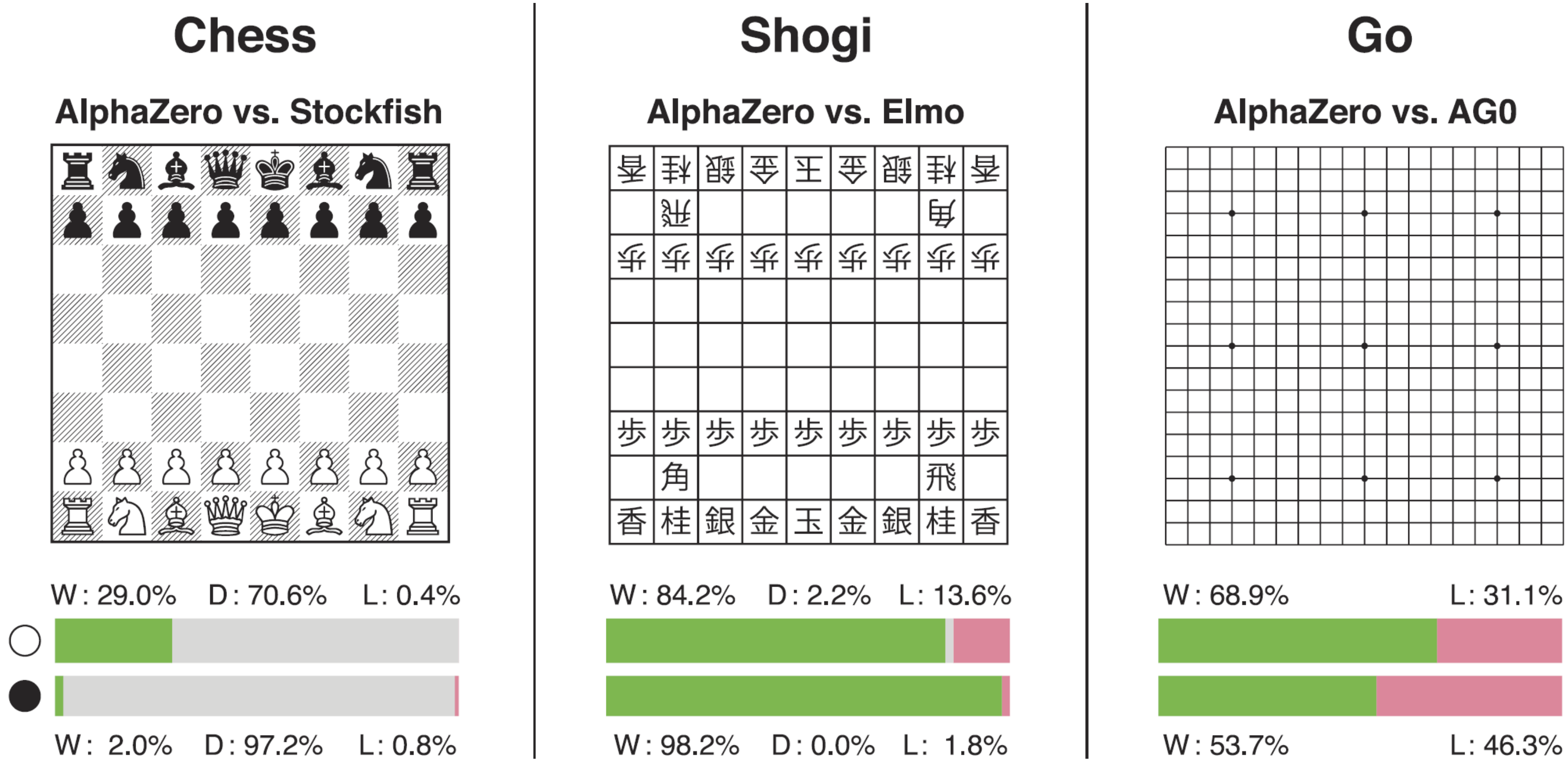


AlphaGo Zero（续）



Silver D, Schrittwieser J, Simonyan K, et al. **Mastering the game of Go without human knowledge**. Nature, 2017, 550: 354-359

AlphaZero



Silver D, Hubert T, Schrittwieser J, et al. **A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play**. Science, 2018, 362(6419): 1140-1144

小结：在线规划

- 离线规划 vs. 在线规划
- 前向搜索
- 分支限界搜索
  - 前向搜索的一种扩展
  - 使用最优值函数的上界和下界来裁剪搜索树
- 稀疏采样
  - 计算复杂度只依赖于n，不依赖于状态数
- 蒙特卡洛树搜索
  - 循环执行4个阶段：选择、扩展、评价、反向更新
  - 直到满足某一终止条件，然后执行最大化当前状态处Q值的行动

内容安排

- 规划
- 马尔科夫决策过程
- 精确动态规划
- 近似动态规划
- 在线规划
- **直接策略搜索**

直接策略搜索

- 局部搜索方法、进化方法
- 交叉熵方法

直接策略搜索

- 特点
  - 直接搜索策略空间
  - 有些问题，状态空间是高维的，但策略空间是相对低维的
  - 近似值函数困难，但直接搜索策略更容易
- 方法
  - 局部搜索方法（也称爬山法、梯度上升法）
  - 进化方法
  - 交叉熵方法
- $\pi_\lambda(a \mid s)$
  - 被参数化为λ的策略π在状态s选择行动a的概率

目标函数

- 给定一个初始状态s，估计

$$U^{\pi_\lambda}(s) \approx \frac{1}{n}\sum_{i=1}^{n} u_i$$

u_i：使用π_λ进行第i次Rollout评价得到的值

- 直接策略搜索的目标：找到λ，最大化：

V(λ)：随机函数

$$V(\lambda) = \sum_s b(s) U^{\pi_\lambda}(s)$$

b：初始状态的分布

Algorithm 4.11 Monte Carlo policy evaluation

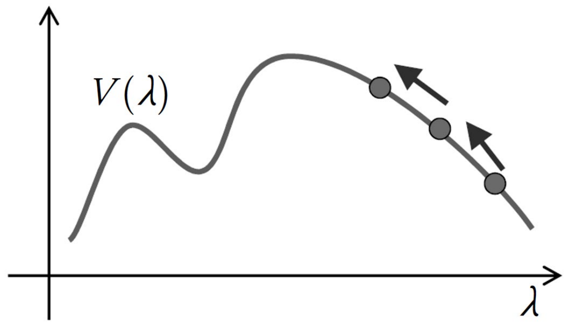
1: **function** MonteCarloPolicyEvaluation(λ, d)
2: **for** $i \leftarrow 1$ **to** n
3: $s \sim b$
4: $u_i \leftarrow$ Rollout(s, d, π_λ)
5: **return** $\frac{1}{n}\sum_{i=1}^{n} u_i$

使用蒙特卡洛策略评价算法来估计$V(\lambda)$

- 如何在策略参数空间中搜索使得V(λ)最大的λ?

局部搜索方法

- 假设： 值V(λ)越大，λ离最优值越接近
  - 易受局部最优解的影响
- 搜索策略
  - 选择沿着最大值的邻居方向搜索，直至收敛
- 一些方法：直接估计某个策略的梯度$\nabla_\lambda V$，然后朝最陡峭上升的方向移动某一数量
  - 解析地推导出梯度



假设策略π是高斯策略模型

$$\pi(a|\boldsymbol{s},\boldsymbol{\mu},\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(a-\boldsymbol{\mu}^\top\boldsymbol{\phi}(\boldsymbol{s}))^2}{2\sigma^2}\right)$$

$$\nabla_{\boldsymbol{\mu}} \log \pi(a|\boldsymbol{s},\boldsymbol{\mu},\sigma) = \frac{a-\boldsymbol{\mu}^\top\boldsymbol{\phi}(\boldsymbol{s})}{\sigma^2}\boldsymbol{\phi}(\boldsymbol{s}), \quad \nabla_{\sigma} \log \pi(a|\boldsymbol{s},\boldsymbol{\mu},\sigma) = \frac{(a-\boldsymbol{\mu}^\top\boldsymbol{\phi}(\boldsymbol{s}))^2-\sigma^2}{\sigma^3}.$$

  - 评估当前搜索点邻域的有限采样点，向有最大值的邻居移动

进化方法

- 进化搜索方法：从生物的进化中获得灵感
- 遗传算法
  - 用（二进制的）字符串表示策略
  - 基于适应性函数，通过杂叉、变异来产生新的一代，重复这一过程直至得到一个可满足的策略
- 遗传编程
  - 用树结构表示策略，比固定长度的字符串更灵活
  - 杂交：交换子树
  - 变异：随机地修改子树
- 与其他方法结合
  - 遗传局部搜索：遗传算法得到一个可满足的策略，再用局部搜索改进策略

直接策略搜索

- 局部搜索方法、进化方法
- 交叉熵方法

信息量

- 衡量一个事件的不确定性
  - 事件发生的概率越大，不确定性越小，信息量越小
- n值随机变量X：$\{1,2,\dots,n\}$
  - $P(x^1),P(x^2),\dots,P(x^n)$
  - $P(x^n) = 1 - (P(x^1) + P(x^2) + \dots + P(x^{n-1}))$
- 定义事件$X=i$的信息量为：

当表示自然对数时，信息量的单位为奈特（nat）

$$I(x^i) = -\log P\left(x^i\right)$$

- 当$P\left(x^i\right)=1$时，该事件必然发生，其信息量为0

熵

- 也称为香农熵，衡量一个系统的混乱程度，代表系统中信息量的总和
  - 熵越大，表明这个系统的不确定性就越大

- 熵是信息量的期望值：

$$H(x)=\sum_{i=1}^{n}\underbrace{P(x^{i})}_{\text{事件}X=i\text{的概率}}\underbrace{I(x^{i})}_{\text{事件}X=i\text{的信息量}}=-\sum_{i=1}^{n}P(x^{i})\log P(x^{i})$$

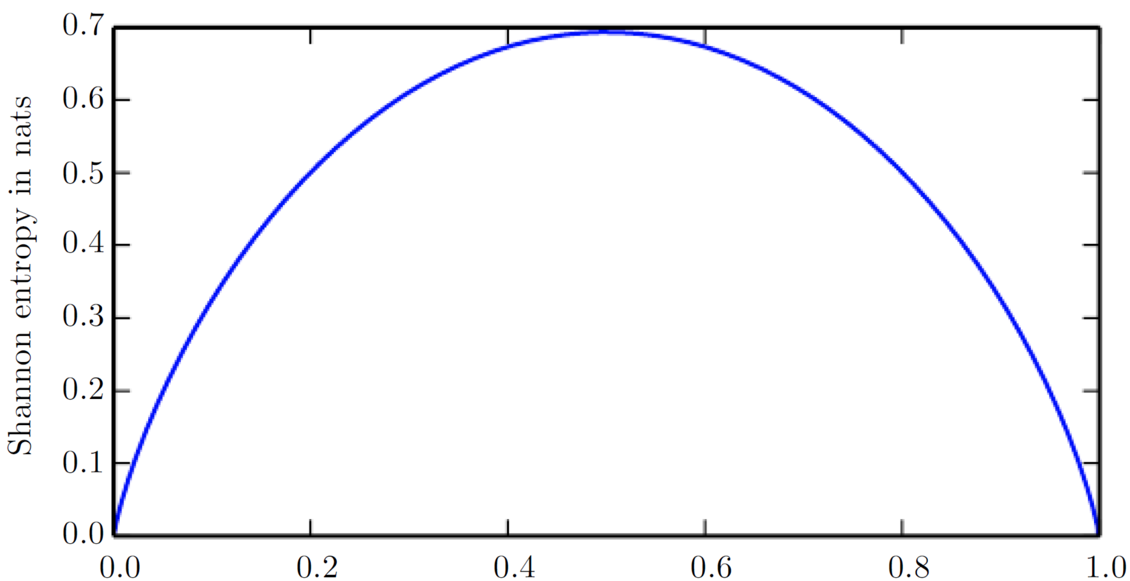
熵（续）

$$H(x)=\sum_{i=1}^{n}P(x^i)\,I(x^i)=-\sum_{i=1}^{n}P(x^i)\log P(x^i)$$

用期望符号$\mathbb{E}$

记为$H(P)$

$$H(\mathrm{x})=\mathbb{E}_{\mathrm{x}\sim P}[I(x)]=-\mathbb{E}_{\mathrm{x}\sim P}[\log P(x)]$$



二值随机变量的熵

相对熵

- 也称为Kullback-Leibler（KL）散度，表示同一个随机变量的两个不同分布间的距离

- $P(x)$、$Q(x)$：随机变量X的两个概率分布

- P对Q的相对熵：

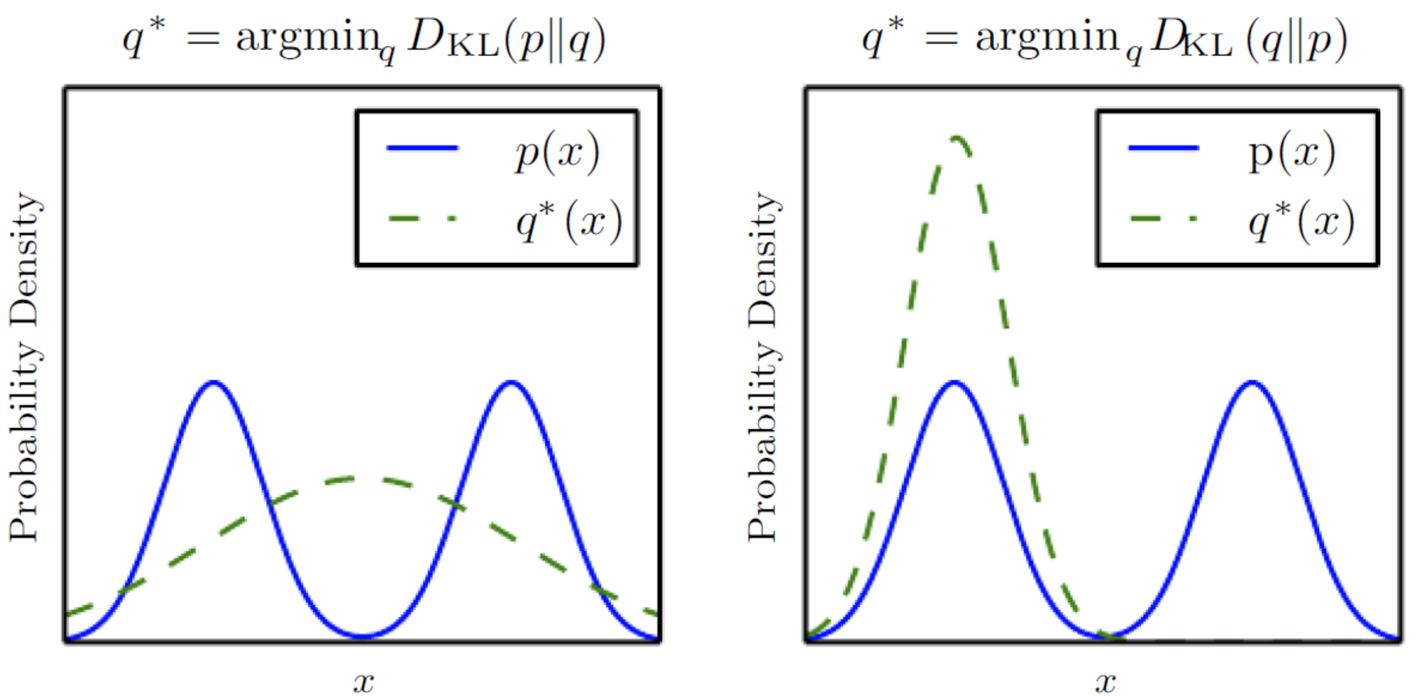
$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{\mathrm{x}\sim P}\left[\log P(x) - \log Q(x)\right]$$

- 性质1：$D_{\mathrm{KL}}(P \,||\, Q) \geq 0$
  - $D_{\mathrm{KL}}(P \,||\, Q) = 0$，如果$P = Q$

相对熵（续）

- 性质2：相对熵不具有对称性，即

$$D_{\mathrm{KL}}(P \,||\, Q) \neq D_{\mathrm{KL}}(Q \,||\, P)$$



交叉熵

- $H(P,Q)$：使用分布$Q(x)$表示真实分布$P(x)$的差异程度

$$H(P,Q) = -\mathbb{E}_{\mathrm{x}\sim P} \log Q(x)$$

- 结合

$$H(P) = -\mathbb{E}_{\mathrm{x}\sim P}[\log P(x)]$$

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log P(x) - \log Q(x)\right]$$

$$\Downarrow$$

$$H(P,Q) = H(P) + D_{\mathrm{KL}}(P\|Q)$$

- 最小化$H(P,Q)$等价于最小化$D_{\mathrm{KL}}(P \,||\, Q)$

交叉熵方法

直接策略搜索的目标是找到λ，最大化$V(\lambda)$

- $P(\lambda \mid \theta)$：λ的分布（反映λ^*的估计），用θ来参数化该分布
- 交叉熵方法：使用交叉熵最小化，基于表现好的策略来更新分布$P(\lambda \mid \theta)$
- 采样
  - 从$P(\lambda \mid \theta)$中采样n个样本，用算法4.11评价它们的性能
  - 对样本按性能降序排列，使得$i < j$意味着$V(\lambda_i) \geq V(\lambda_j)$
- 更新
  - 使用性能最好的m个样本（精英样本）来更新θ，使用交叉熵最小化：

$$-H(P,Q)$$

$$\theta \leftarrow \arg\max_{\theta} \underline{\sum_{j=1}^{m} \log P(\lambda_j \mid \theta)}$$

$$\boxed{H(P,Q) = -\mathbb{E}_{\mathrm{x}\sim P} \log Q(x)}$$

交叉熵方法（续）

输入：λ的初始分布参数θ，样本数n，精英样本数m

Algorithm 4.12 Cross entropy policy search

function CrossEntropyPolicySearch(θ, n, m)
    repeat
        for $i \leftarrow 1$ **to** n
            $\lambda_i \sim P(\cdot \mid \theta)$
            $v_i \leftarrow$ MonteCarloPolicyEvaluation(λ_i)
        Sort $(\lambda_1, \ldots, \lambda_n)$ in decreasing order of v_i
        $\theta \leftarrow \arg\max_\theta \sum_{j=1}^{m} \log P(\lambda_j \mid \theta)$
    until convergence
    return $\lambda \leftarrow \arg\max P(\lambda \mid \theta)$

采样

更新

新的θ对应为性能最好的m个样本的极大似然估计

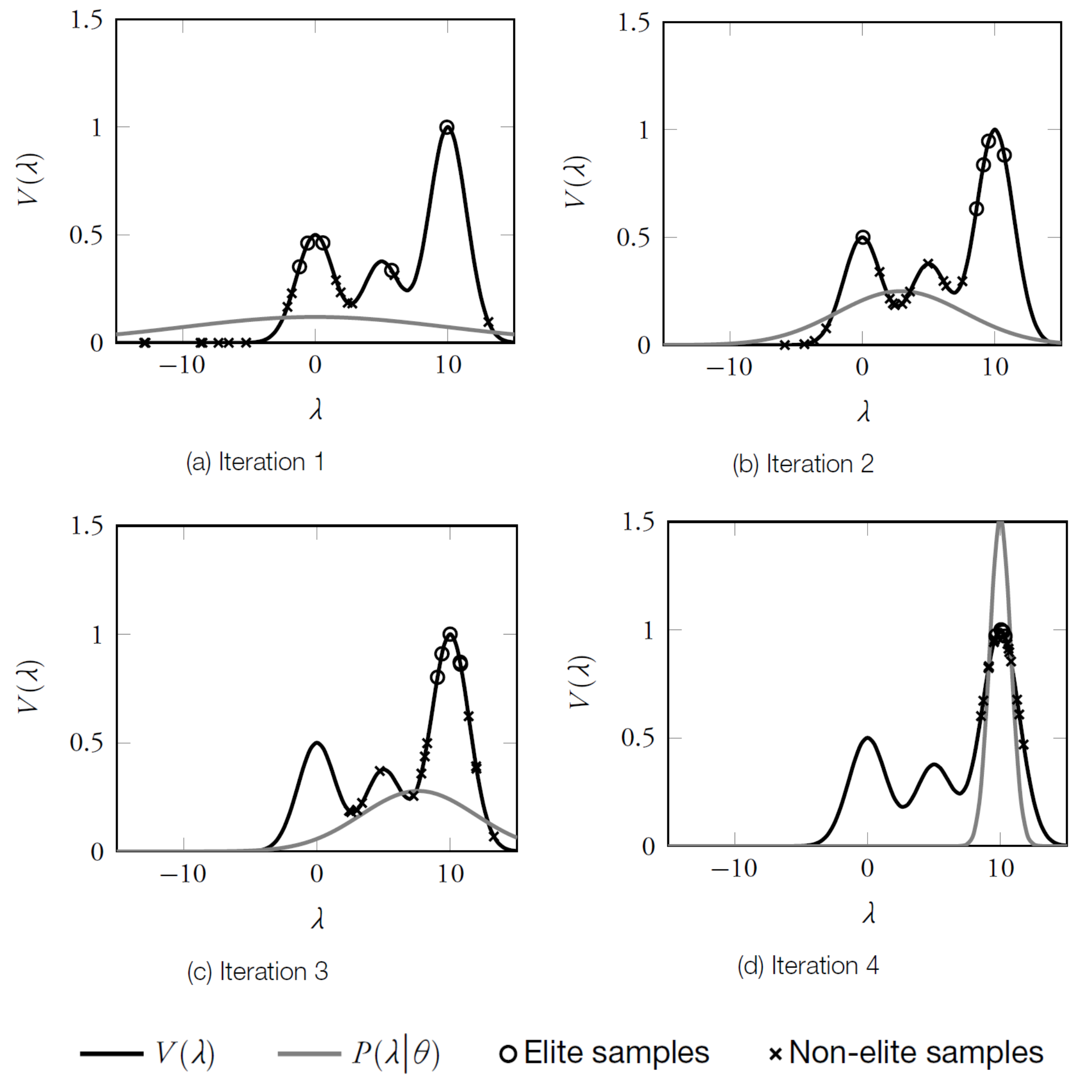
交叉熵方法（续）

- 样本数：20
- 精英样本数：5

高斯分布

- $\theta = (\mu, \sigma)$
  - 初始化为$(0, 10)$
- $P(\lambda \mid \theta) \sim \mathcal{N}(\lambda \mid \mu, \sigma)$

把精英样本的平均值、标准差设为更新后的$P(\lambda \mid \theta)$的均值、标准差



小结：直接策略搜索

- 目标：在被λ参数化的策略空间中，直接搜索最大化下式的λ：

$$V(\lambda) = \sum_s b(s) U^{\pi_\lambda}(s)$$

- 局部搜索方法（也称爬山法、梯度上升法）
- 进化方法
  - 遗传算法、遗传编程、与其他方法结合（如局部搜索）
- 交叉熵方法
  - 采样多个样本，评价样本的性能，对样本按性能降序排列
  - 使用交叉熵最小化，基于表现好的策略来更新分布$P(\lambda \mid \theta)$

课后练习4.9

- 试比较动态规划、近似动态规划和在线规划，说明每类规划方法在何种情况下更有优势。

课后练习4.10

- 在稀疏采样方法中，如果令$n = |\mathcal{S}|$，那么它与前向搜索方法是等价的吗？为什么？

课后练习4.11

- 给定一个MDP问题，其中，$|\mathcal{S}| = 10$，$|\mathcal{A}| = 3$，$T(s' \mid s, a) = \frac{1}{|\mathcal{S}|}$是对所有$s$和$a$都成立的均匀转移分布。问：用样本数$n = |\mathcal{S}|$和深度$d = 1$的稀疏采样方法产生与深度$d = 1$的前向搜索方法完全相同的搜索树的概率是多少？

编程作业2

用蒙特卡洛树搜索方法设计和实现会玩2048游戏的智能程序。要求程序每1秒执行一个行动，并可视化智能程序玩2048游戏的过程。

提交代码（用Python或者C++实现 ）和实验报告。

截止时间为：**2021年5月26日**

本科生班的同学把作业发给尹皓： yinh@lamda.nju.edu.cn

研究生班的同学把作业发给刘旭辉： liuxh@lamda.nju.edu.cn