9 集成学习与规划

在本章中，我们将从学习和规划的角度进一步分析强化学习。我们首先将介绍基于模型和无模型强化学习的概念，并着重介绍模型规划的优势。为了在强化学习中充分利用基于模型和无模型方法，我们将介绍集成学习和规划的架构，并详细阐述应用其架构的 Dyna-Q 算法。最终，将进一步详细分析集成学习和规划的基于模拟的搜索应用。

9.1 简介

在强化学习中，智能体可以和环境进行交互。智能体在每一轮交互中收集到的信息可以称为智能体的经验，这能帮助智能体提升自身的决策策略。一般来说，学习指代智能体决策策略基于实际和环境的交互逐渐提升的过程。直接策略学习是最为基本的学习方式，如图 9.1 所示，其中，智能体首先根据当前的决策策略在环境中制定动作，环境会基于智能体当前的状态和动作反馈给

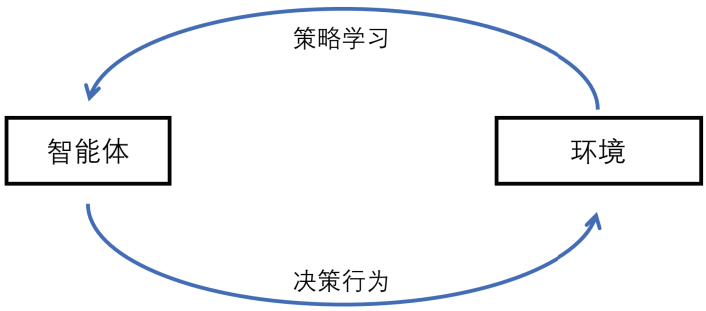

图 9.1 直接策略学习

智能体所得到的收益，使其能够评估当前策略的表现并帮助智能体探索如何进一步提升策略。然而，直接策略学习是基于智能体在环境中每一个单步动作所产生的经验，由于环境的随机性和不确定性，基于单步动作的经验会使学习结果存在很大方差，大大影响了学习的速度和质量。

为了提高学习效率，在策略学习的每一个学习周期中，积累多轮和环境的交互作为智能体的经验是很有帮助的。通过在环境中进行演算（Roll-out）收集多轮交互信息，即在环境中根据当前的状态和决策策略形成一条具体的包含一系列状态、动作和奖励信息的探索轨迹。在一般的无模型学习中，智能体将在真实的环境中在线演算，并将获得的多轮交互信息用于策略学习。

然而，在环境中通过在线演算产生经验的成本很高。例如，在工业界的应用中，一些状态可以指代系统崩溃或者设备爆炸，这些状态在策略学习的探索过程中是十分危险的。另外，在实际环境中只能顺序演算，不能并行计算，这导致其采样效率和学习速度都很低。因而，在一些场景下，我们希望能够使用模拟环境来取代实际环境进行探索和经验积累。在模拟环境中的演算被称为规划（Planning），可通过并行计算高效地为策略学习产生大量模拟经验。为了在规划中使用有效的模拟环境，基于模型的方法得以提出。

9.2　基于模型的方法

为了能够实行规划，模型的概念将在智能体和环境之间产生 (Kaiser et al., 2019)，如图 9.2 所示，当智能体在状态 S_t 采取决策动作 A_t 时，环境会为模型给予反馈奖励 R_{t+1} 并使智能体进入下一状态 S_{t+1}。根据智能体和环境之间收集到的经验信息，我们将 S_{t+1} 和 (S_t, A_t) 之间的映射关系称为转移模型，并将 R_{t+1} 和 (S_t, A_t) 之间的映射关系称为奖励模型。当状态不能完全被观察信息表示时，还将设定观察模型 $\mathcal{M}(O_t|S_t)$ 和表示模型 $\mathcal{M}(S_{t+1}|S_t, A_t, O_{t+1})$ (Hafner et al., 2019)，其中 O_t 表示在状态 S_t 下第 t 步所对应的观察信息。例如，捕捉到的关乎物体运动的图片属于观察信息，可以体现该物体蕴含的所处状态信息。后面，为了集中分析其中的转移模型和奖励模型，我们假设状态是完全可观测的。我们将转移模型和奖励模型分别由方程 $\mathcal{F}_s$ 和 $\mathcal{F}_r$ 表示：

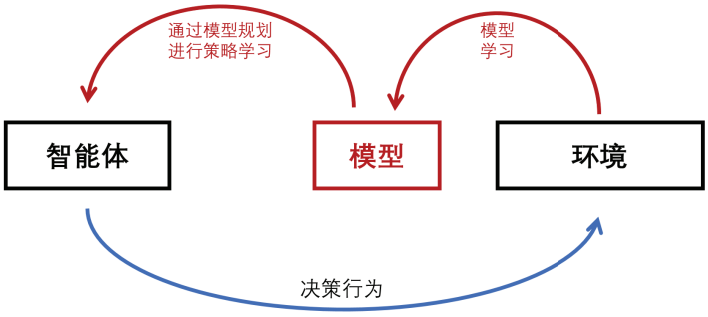

图 9.2　基于模型的强化学习方法

$$S_{t+1} \sim \mathcal{F}_s\left(S_t, A_t\right), \tag{9.1}$$

$$R_{t+1} = \mathcal{F}_r\left(S_t, A_t\right). \tag{9.2}$$

模型学习是一个监督式的拟合学习过程，目标是建立一个虚拟的环境，其中的转移关系和奖励关系和真实环境保持一致。因而，基于对真实环境的了解，我们可以使用一个环境模型使智能体在其中进行规划，然后将收集到的经验信息用于帮助其策略学习。

在不同的应用场景中，模型学习和策略学习的关系是多样的，具体如下所述。

- **直接学习**：如果智能体已经基于规则或专家信息和环境交互过多次，那么之前收集到的经验信息可以直接用来进行模型学习。当模型学习完成时，智能体可以将训练后的模型当作模拟的环境，并与其交互帮助其进行策略学习。
- **迭代学习**：如果模型在初始时并没有足够的数据进行学习，那么模型学习和策略学习可以迭代交替进行。基于当前智能体和环境交互产生的有限信息，模型可以学习真实环境中部分且有限的信息。智能体在基于有限学习产生的模拟环境进行规划并以此训练参数，且其策略表现得到了少许提升后，将用更新的策略在真实环境中交互，并将收集到的经验信息进一步用于对模型的学习。随着迭代次数的增加，模型学习和策略学习将逐步收敛到最优结果。因此，模型学习和策略学习可以相互辅助而进行有效的学习。

因此，基于模型的强化学习将通过对真实环境的学习建立一个模拟环境的模型，并在其中进行规划，使智能体更好地进行策略学习。模型学习的优势可列举如下：

- 由于规划可以在智能体和模型之间完成，智能体不需要在真实环境中采取大量的决策动作进行探索和策略学习。因而，和成本高并且需要在线采取动作的真实环境相比，基于模型的方法能够有效地降低训练时间并且保障在策略学习过程中的安全性。例如，在真实环境中，机器人完成任务需要实际操作，在 QT-Opt (Kalashnikov et al., 2018) 方法中，为了完成抓取的任务，7 个机器人需要昼夜不停地在实际环境中收集采样数据。然而一个模拟的环境（通过学习或人工建立）可以用来节约大量的时间并且降低机器人的磨损。
- 当策略学习在智能体和模拟模型之间进行时，学习过程可以采用并行计算。在分布式系统中可以存在多个学习者合作同时进行策略学习，其中每个学习者可以和一个根据真实环境模拟的模型进行交互，从而所有学习者都可以在其对应的模型进行规划。模型之间是相互独立的，并且不会影响到真实环境中所处的状态信息。因此，具有并行性的策略学习大大提高了学习效率，且增大了可学习问题的规模。

然而，基于模型的强化学习的结构同样也存在缺点和不足：

- 在基于模型的强化学习中，模型学习的表现将会影响策略学习的结果。对于复杂且动态的环境场景，如果学习到的模型不能很好地模拟出真实环境，智能体在规划中会和一个错误且不准确的模型进行交互，从而将增大策略学习的误差。
- 如果真实环境有更新或者调整，模型需要通过多次迭代之后才会学到环境的变化，然后还需要耗费大量训练时间使智能体学习并调整其策略。因此，对于在线学习中真实环境的变化，

智能体对其策略的相应调整有着很高的延迟，这并不适用于那些对实时性有要求的应用。

9.3　集成模式架构

综合无模型和基于模型的强化学习方法的优劣，集成学习和规划的过程可以很好地将无模型和基于模型的方法结合在一起。对于不同的应用场景，集成学习和规划的方法和架构是不同的。

一般来说，在无模型的方法中，智能体仅在与真实环境的交互中得到真实的经验，没有采用规划辅助其策略的学习和提升。在基本的基于模型的方法中，首先将通过智能体和真实环境的交互进行模型学习，然后基于学到的模型，智能体将迭代式采取规划并用收集到的经验进行策略学习。

由于模型处于智能体和环境之间，在智能体策略学习中，经验来源可以分为如下两类：

- **真实经验**：真实经验是从智能体和真实环境中直接采样获得的。一般来说，真实经验体现了环境正确的特征和属性，但获得成本较高，并且在真实环境中的探索不可逆且难以人工干预。
- **模拟经验**：模拟经验是从模型规划过程中获得的，可能不能准确地表现真实环境的真实特征，但模型很容易人工操纵，并且可以通过模型学习减小模型和真实环境的误差。

对于策略学习，如果我们能够同时考虑真实经验和模拟经验，那么就能结合无模型和基于模型的方法的优势，提高学习的效率和准确性。Dyna 架构在 (Sutton, 1991) 中提出。如图 9.3 所示，根据基础的基于模型的方法，在策略学习中，智能体不仅从已经学到的模型所提供的模拟经验中更新策略，并且考虑了与真实环境交互所收集到的真实经验。因此，在策略学习中，模拟经验能够保证学习过程中有足够多的训练数据来降低学习方差，另外，真实经验能够更准确地体现环境的动态变化和正确特征，从而降低由于环境而产生的学习偏差。

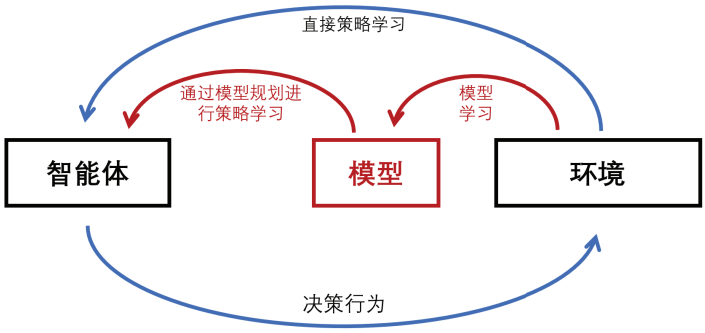

图 9.3　Dyna 架构

基于此架构，Dyna-Q 算法得以进一步提出，如算法 9.30 所述。Dyna-Q 算法将建立并维护一个 Q 表格，据此指导智能体做出动作决策。在每个学习周期中，Q 表格通过智能体和真实环境的

交互中学习更新，模拟的模型同时也会从真实经验中学习，并且通过规划获得 n 组模拟经验用于进一步的 Q 表格学习。因此，随着学习周期的增加，Q 表格能够学习并收敛到最佳的结果。

算法 9.30 Dyna-Q

初始化 $Q(s,a)$ 和 Model(s,a)，其中 $s \in \mathcal{S}$，$a \in \mathcal{A}$。
while(true):
(a) $s \leftarrow$ 当前（非终止）状态
(b) $a \leftarrow \epsilon$-greedy(s,Q)
(c) 执行决策动作 a; 观测奖励 r, 获得下一个状态 s'
(d) $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
(e) Model$(s,a) \leftarrow r, s'$
(f) 重复 n 次:
$s \leftarrow$ 随机历史观测状态
$a \leftarrow$ 在状态 s 下历史随机决策动作
$r, s' \leftarrow$ Model(s,a)
$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$

9.4 基于模拟的搜索

在本节中，我们侧重于规划部分，并介绍一些基于模拟的搜索算法，使其在当前的状态通过演算形成探索轨迹。因此，基于模拟的搜索算法一般是使用基于样本规划的前向搜索范式。前向搜索和采样具体的阐述如下。

- **前向搜索**：在规划过程中，智能体当前所处马尔可夫过程中的状态比其他的状态更值得关注。因而从另一角度，我们将具有有限选择的 MDP 看作一个树形的结构，其中树的根部代表当前状态，如图 9.4 所示，前向搜索算法从当前的状态选择最佳的决策动作，并且通过树形结构的枝干来考虑未来的选择。
- **采样**：当基于 MDP 采用规划过程时，从当前的状态到下一个状态可能有多种选择，因而在规划中需要采样的操作，即智能体随机选定下一个状态并继续前向搜索的演算过程。因而下一个状态的选取具有随机性，并且有可能服从某种概率或分布，具体是由模拟中智能体采取的决策策略决定的。

在基于模拟的搜索中，模拟策略被用来指导规划过程中探索的方向。模拟策略与智能体学习的策略相结合，有助于规划过程能够准确地反映智能体当前的决策策略。

下面将进一步介绍几种不同的基于模拟的搜索方法并结合策略学习来解决问题。

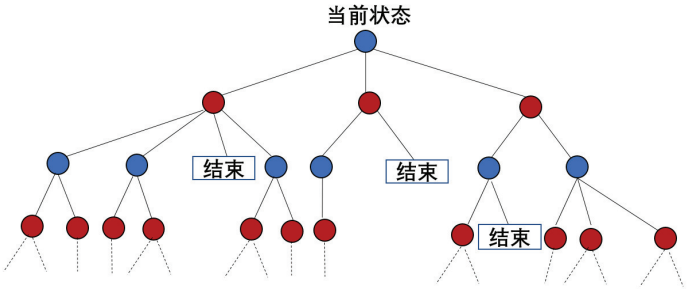

图 9.4　前向搜索

9.4.1　朴素蒙特卡罗搜索

如果一开始提供了固定的模型 $\mathcal{M}$ 和固定的模拟策略 π，朴素蒙特卡罗搜索可以依据模拟得到的经验来评估对应动作的性能好坏并更新学习到的策略。如算法 9.31 所示，对每一个作用于当前状态 S_t 的动作 $a, a \in \mathcal{A}$，执行模拟策略 π 并用 G_t^k 表示第 k 个轨迹的全部奖励。根据保存的轨迹，我们利用 $Q(S_t, A_t)$ 来评估选择动作 A_t 的性能，最后根据当前状态下所有动作各自的 Q 值选择最优的动作。

算法 9.31 朴素蒙特卡罗搜索

固定模型 $\mathcal{M}$ 和模拟策略 π
for 每个动作 $a \in \mathcal{A}$ **do**
　for 每个片段 $k \in \{1, 2, \cdots, K\}$ **do**
　　根据模型 $\mathcal{M}$ 和模拟策略 π, 从当前状态 S_t 开始在环境中展开
　　记录轨迹 $\{S_t, a, R_{t+1}^k, S_{t+1}^k, A_{t+1}^k, R_{t+2}^k, \cdots, S_T^k\}$
　　计算从每个 S_t 开始的累积奖励 $G_t^k = \sum_{j=t+1}^{\mathrm{T}} R_j^k$
　end for
　$Q(S_t, a) = \frac{1}{K} \sum_{k=1}^{K} G_t^k$
end for
返回当前最大 Q 值的动作 $A_t = \arg\max_{a \in \mathcal{A}} Q(S_t, a)$。

9.4.2　蒙特卡罗树搜索

朴素蒙特卡罗搜索的一个明显不足是，它的模拟策略是固定的，从而没有办法利用在规划过程中学习到的信息。蒙特卡罗树搜索（Monte Carlo Tree Search，MCTS）(Browne et al., 2012) 正是针对这个不足所设计的。具体地说，MCTS 维护了一棵搜索树来保存收集到的信息并逐步优化模拟策略。

如算法 9.32 所示，在从当前的状态 S_t 开始采样到一个轨迹之后，对于轨迹中所有访问过的

(s,a)，MCTS 类似地使用平均回报更新了 Q 值，进而根据树中新的 Q 值更新每个节点处的模拟策略 π。一个更新模拟策略 π 的方法是根据当前 Q 值的 ϵ 贪心策略。当模拟策略到达一个新的当前并不在搜索树中的状态的时候，π 转换成默认的策略，比如均匀探索策略。第一个被探索的新状态会接着被加入搜索树中。[1] MCTS 重复这个节点评估和策略提升的过程直到到达模拟的预算。最后，智能体选择在当前状态 S_t 上有最大 Q 值的动作。

算法 9.32 蒙特卡罗树搜索

固定模型 $\mathcal{M}$
初始化模拟策略 π
for 每个动作 $a \in \mathcal{A}$ **do**
  for 每个片段 $k \in \{1, 2, \cdots, K\}$ **do**
    根据模型 $\mathcal{M}$ 和模拟策略 π 从当前状态 S_t 在环境中展开
    记录轨迹 $\{S_t, a, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, \cdots S_T\}$
    用从 (S_t, A_t)，$A_t = a$ 开始的平均回报更新每个 $(S_i, A_i), i = t, \cdots, T$ 的 Q 值
    由当前的 Q 值更新模拟策略 π
  end for
end for
返回当前最大 Q 值的动作 $A_t = \arg\max_{a \in \mathcal{A}} Q(S_t, a)$

9.4.3 时间差分搜索

除了 MCTS 的方法，时间差分（Temporal Difference，TD）搜索同样受到关注 (Silver et al., 2012)。和 MCTS 的方法相比，TD 搜索不需要演算一个扩展轨迹并用其来评估和更新当前策略。在模拟的每一步中，策略都将被更新并用更新的策略指导智能体在下一个状态中做出决策动作。

Dyna-2 算法就是采用 TD 搜索的方式 (Silver et al., 2008)，如算法 9.33 所述，智能体将存储两组网络参数，分别存储于长期存储空间和短期存储空间。在下层中通过采用 TD 学习的方法，短期存储空间中的网络参数将会根据收集到的模拟经验进行更新，并在策略 $\overline{Q}$ 的指导下将学到的网络参数 $\overline{\theta}$ 用于帮助智能体在真实环境中做出决策动作，而在长期存储空间的网络参数将在真实环境的探索中通过在上层的 TD 学习得到更新。在上层中学习到的基于网络参数 θ 的策略 Q 将是最终智能体学习到的最佳策略。

和 MCTS 的方法相比，由于每一步策略都会更新，TD 搜索会更有效率。然而，由于频繁的更新，TD 搜索倾向于降低结果的方差但是有可能增大偏差。

[1] 另一个方法是将轨迹上所有新的节点都加入搜索树中。

算法 9.33 Dyna-2

function LEARNING
　初始化 $\mathcal{F}_s$ 和 $\mathcal{F}_r$
　$\theta \leftarrow 0$　　# 初始化长期存储空间中网络参数
　loop
　　$s \leftarrow S_0$
　　$\overline{\theta} \leftarrow 0$　　# 初始化短期存储空间中网络参数
　　$z \leftarrow 0$　　# 初始化资格迹
　　SEARCH(s)
　　$a \leftarrow \pi(s; \overline{Q})$　　# 基于和 $\overline{Q}$ 相关的策略选择决策动作
　　while s 不是终结状态 **do**
　　　执行 a, 观测奖励 r 和下一个状态 s'
　　　$(\mathcal{F}_s, \mathcal{F}_r) \leftarrow$ UpdateModel(s, a, r, s')
　　　SEARCH(s')
　　　$a' \leftarrow \pi(s'; \overline{Q})$　　# 选择决策动作使其用于下一个状态 s'
　　　$\delta \leftarrow r + Q(s', a') - Q(s, a)$　　# 计算 TD-error
　　　$\theta \leftarrow \theta + \alpha(s, a)\delta z$　　# 更新长期存储空间中网络参数
　　　$z \leftarrow \lambda z + \phi$　　# 更新资格迹
　　　$s \leftarrow s', a \leftarrow a'$
　　end while
　end loop
end function

function SEARCH(s)
　while 时间周期内 **do**
　　$\overline{z} \leftarrow 0$　　# 清除短期存储的资格迹
　　$a \leftarrow \overline{\pi}(s; \overline{Q})$　　# 基于和 $\overline{Q}$ 相关的策略决定决策动作
　　while s 不是终结状态 **do**
　　　$s' \leftarrow \mathcal{F}_s(s, a)$　　# 获得下一个状态
　　　$r \leftarrow \mathcal{F}_r(s, a)$　　# 获得奖励
　　　$a' \leftarrow \overline{\pi}(s'; \overline{Q})$
　　　$\overline{\delta} \leftarrow R + \overline{Q}(s', a') - \overline{Q}(s, a)$　　# 计算 TD-error
　　　$\overline{\theta} \leftarrow \overline{\theta} + \overline{\alpha}(s, a)\overline{\delta}\overline{z}$　　# 更新短期存储空间中网络参数
　　　$\overline{z} \leftarrow \overline{\lambda}\overline{z} + \overline{\phi}$　　# 更新短期存储的资格迹
　　　$s \leftarrow s', a \leftarrow a'$
　　end while
　end while
end function

参考文献

BROWNE C B, POWLEY E, WHITEHOUSE D, et al., 2012. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in games, 4(1): 1-43.

HAFNER D, LILLICRAP T, BA J, et al., 2019. Dream to control: Learning behaviors by latent imagination[J]. arXiv preprint arXiv:1912.01603.

KAISER L, BABAEIZADEH M, MILOS P, et al., 2019. Model-based reinforcement learning for atari[Z].

KALASHNIKOV D, IRPAN A, PASTOR P, et al., 2018. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation[J]. arXiv preprint arXiv:1806.10293.

SILVER D, SUTTON R S, MÜLLER M, 2008. Sample-based learning and search with permanent and transient memories[C]//Proceedings of the 25th international conference on Machine learning. ACM: 968-975.

SILVER D, SUTTON R S, MÜLLER M, 2012. Temporal-difference search in computer go[J]. Machine learning, 87(2): 183-219.

SUTTON R S, 1991. Dyna, an integrated architecture for learning, planning, and reacting[J]. ACM Sigart Bulletin, 2(4): 160-163.