第十一章 对状态的不完全观测

11.1 不完全观测问题

之前章节中的 DQN $Q(s,a;\boldsymbol{w})$，策略网络 $\pi(a|s;\boldsymbol{\theta})$、$\boldsymbol{\mu}(s;\boldsymbol{\theta})$，价值网络 $q(s,a;\boldsymbol{w})$、$v(s;\boldsymbol{w})$ 都需要把当前状态 s 作为输入。之前我们一直假设可以完全观测到状态 s；在围棋、象棋、五子棋等简单的游戏中，棋盘上当前的格局就是完整的状态，符合完全观测的假设。但是在很多实际应用中，完全观测假设往往不符合实际。比如在星际争霸、英雄联盟等电子游戏中，屏幕上当前的画面并不能完整反映出游戏的状态，因为观测只是地图的一小部分；甚至最近的 100 帧也无法反映出游戏真实的状态。

把 t 时刻的状态记作 s_t，把观测记作 o_t。观测 o_t 可以是当前游戏屏幕上的画面，也可以是最近 100 帧画面。我们无法用 $\pi(a_t|s_t;\boldsymbol{\theta})$ 做决策，因为我们不知道 s_t。最简单的解决办法就是用当前观测 o_t 代替当前状态 s_t，用 $\pi(a_t|o_t;\boldsymbol{\theta})$ 做决策。同理，对于 DQN 和价值网络，也用 o_t 代替 s_t。虽然这种简单的方法可行，但是效果恐怕不好。

(a) 对状态的完全观测　(b) 对状态的不完全观测　(c) 记忆过去的观测

图 11.1: 在迷宫问题中，智能体可能知道迷宫的整体格局，也可能只知道自己附近的格局。

图 11.1 的例子是让智能体走迷宫。图 11.1(a) 中智能体可以完整观测到迷宫 s；这种问题最容易解决。图 11.1(b) 中智能体只能观测到自身附近一小块区域 o_t，这属于不完全观测问题，这种问题较难解决。如果仅仅靠当前观测 o_t 做决策，智能体做出的决策是非常盲目的，很难走出迷宫。一种更合理的办法是让智能体记住过去的观测，这样就能对状态的观测越来越完整，做出越来越理性的决策；如图 11.1(c) 所示。

对于不完全观测的强化学习问题，应当记忆过去的观测，用所有已知的信息做决策。这正是人类解决不完全观测问题的方式。对于星际争霸、扑克牌、麻将等不完全观测的游戏，人类玩家也需要记忆；人类玩家的决策不止依赖于当前时刻的观测 o_t，而是依赖于过去所有的观测 $o_1,\cdots,o_t$。把从初始到 t 时刻为止的所有观测记作：

$$\boldsymbol{o}_{1:t} \;=\; \left[o_1,\, o_2,\, \cdots,\, o_t\right]$$

可以用 $\boldsymbol{o}_{1:t}$ 代替状态 s，作为策略网络的输入，那么策略网络就记作：

$$\pi\left(a_t \,\middle|\, \boldsymbol{o}_{1:t};\, \boldsymbol{\theta}\right).$$

该如何实现这样一个策略网络呢？请注意，$\boldsymbol{o}_{1:t}$ 的大小是变化的。如果 $o_1, \cdots, o_t$ 都是 $d \times 1$ 的向量，那么 $\boldsymbol{o}_{1:t}$ 是 $d \times t$ 的矩阵或 $dt \times 1$ 的向量，它的大小随 t 增长。卷积层和全连接层都要求输入大小固定，因此不能简单地用卷积层和全连接层实现策略网络。一种可行的办法是将卷积层、全连接层与循环层结合，这样就能处理不固定长度的输入。

11.2 循环神经网络 (RNN)

循环神经网络 (Recurrent Neural Network)，缩写 RNN，是一类神经网络的总称，由循环层 (Recurrent Layers) 和其他种类的层组成。循环层的作用是把一个序列（比如时间序列、文本、语音）映射到一个特征向量。设向量 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ 是一个序列。对于所有的 $t = 1, \cdots, n$，循环层把 $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)$ 映射到特征向量 $\boldsymbol{h}_t$。依次把 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ 输入循环层，会得到：

$$\begin{aligned}
(\boldsymbol{x}_1) \quad &\Longrightarrow \quad \boldsymbol{h}_1, \\
(\boldsymbol{x}_1, \boldsymbol{x}_2) \quad &\Longrightarrow \quad \boldsymbol{h}_2, \\
(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \quad &\Longrightarrow \quad \boldsymbol{h}_3, \\
&\vdots \\
(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_{n-1}) \quad &\Longrightarrow \quad \boldsymbol{h}_{n-1}, \\
(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_{n-1}, \boldsymbol{x}_n) \quad &\Longrightarrow \quad \boldsymbol{h}_n.
\end{aligned}$$

RNN 的好处在于不论输入序列的长度 n 是多少，从序列中提取出的特征向量 $\boldsymbol{h}_n$ 的大小是固定的。请特别注意，$\boldsymbol{h}_t$ 并非只依赖于 $\boldsymbol{x}_t$ 这一个向量，而是依赖于 $[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t]$；理想情况下，$\boldsymbol{h}_t$ 记住了 $[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t]$ 中的主要信息。比如 $\boldsymbol{h}_3$ 是对 $[\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3]$ 的概要，而非是对 $\boldsymbol{x}_3$ 这一个向量的概要。

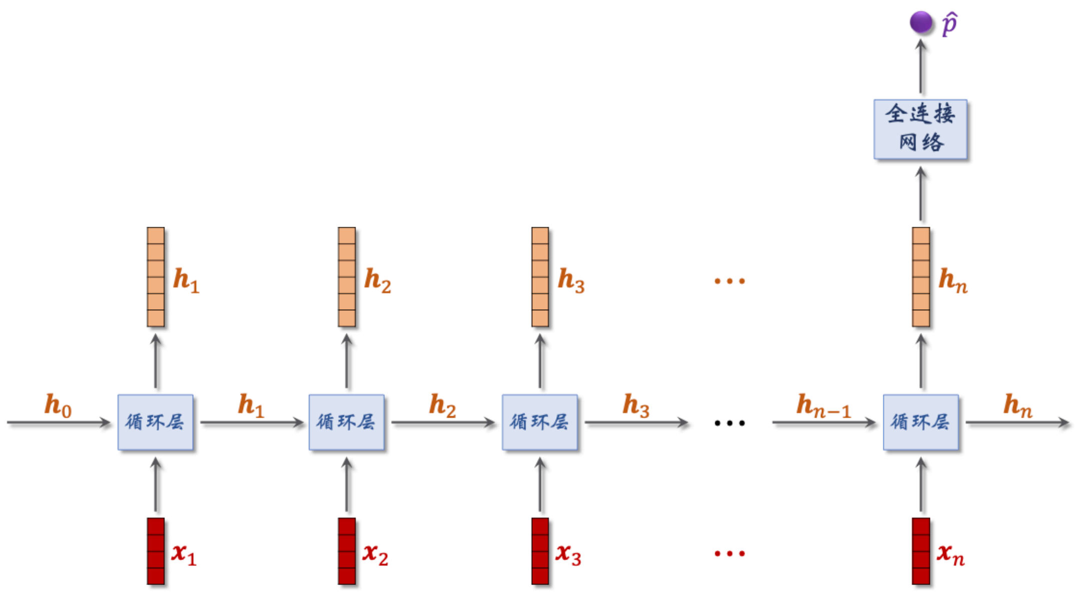

图 11.2: 输入是序列 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$。向量 $\boldsymbol{h}_n$ 是从所有 n 个输入中提取的特征，可以把它看做输入序列的一个概要。把 $\boldsymbol{h}_n$ 输入全连接层（带 Sigmoid 激活函数），得到分类结果 $\hat{p}$。

举个例子，用户给商品写的评论由 n 个字组成（不同的评论有不同的 n），我们想要判断评论是正面的还是负面的，这是个二元分类问题。用词嵌入 (Word Embedding) 把每个字映射到一个向量，得到 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$，把它们依次输入循环层。循环层依次输出 $\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n$。我们只需要用 $\boldsymbol{h}_n$，因为它是从全部输入 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ 中提取的特征；可以忽略掉 $\boldsymbol{h}_1, \cdots, \boldsymbol{h}_{n-1}$。最后，二元分类器把 $\boldsymbol{h}_n$ 作为输入，输出一个介于 0 到 1 之间的数 $\hat{p}$，

0 代表负面，1 代表正面。图 11.2 描述了神经网络的结构。

循环层的种类有很多，常见的包括简单循环层、LSTM、GRU。本书只介绍简单循环层。LSTM、GRU 是对简单循环层的改进，结构更复杂，效果更好；但是它们的原理与简单循环层基本相同。读者只需要理解简单循环层就足够了。用 TensorFlow、PyTorch、Keras 编程实现的话，几种循环层的使用方法完全相同（唯一区别是函数名）。

简单循环层的输入记作 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^{d_{\text{in}}}$，输出记作 $\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n \in \mathbb{R}^{d_{\text{out}}}$。循环层的参数是矩阵 $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ 和向量 $\boldsymbol{b} \in \mathbb{R}^{d_{\text{in}}}$。循环层的输出是这样计算出来的：从 $t = 1, \cdots, n$，依次计算

$$\boldsymbol{h}_t \;=\; \tanh\Big(\boldsymbol{W}\left[\boldsymbol{h}_{t-1};\, \boldsymbol{x}_t\right] + \boldsymbol{b}\Big).$$

图 11.3 解释上面的公式。注意，不论输入序列长度 n 是多少，简单循环层的参数只有唯一的 $\boldsymbol{W}$ 和 $\boldsymbol{b}$。公式中的 tanh 是双曲正切函数，见图 11.4。tanh 是标量函数；如果输入是向量，那么 tanh 应用到向量的每一个元素上。对于 $d \times 1$ 的向量 $\boldsymbol{z}$，有

$$\tanh\left(\boldsymbol{z}\right) \;=\; \Big[\tanh\left(z_1\right), \tanh\left(z_2\right), \cdots, \tanh\left(z_d\right)\Big]^T.$$

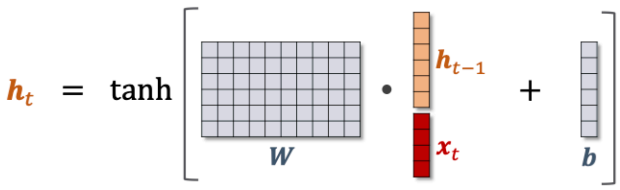

图 11.3: 简单循环层。

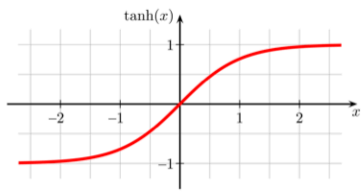

图 11.4: 双曲正切函数。

Attention 与 RNN 相结合可以让 RNN 更好地记住过去的观测，而且能关注到最相关的历史记录。Attention 是改进 RNN 最有效的技巧，RNN+Attention 通常比只用 RNN 表现更好。读者甚至可以只用 Attention，不用 RNN。Attention 层也允许输入的大小动态变化，就像 RNN 一样，因此能用 RNN 的任务都可以用 Attention 层。Attention 层的原理比较复杂，此处就不介绍了，有兴趣的读者可以参考 Attention、Transformer、BERT 的论文。

11.3 RNN 作为策略网络

在不完全观测的设定下，我们希望策略网络能利用所有已经收集的观测 $\boldsymbol{o}_{1:t} = [o_1, \cdots, o_t]$ 做决策。定义策略网络为 $\boldsymbol{f}_t = \pi(a_t|\boldsymbol{o}_{1:t};\boldsymbol{\theta})$，结构如图 11.5 所示。在第 t 时刻，观测到 o_t，用卷积网络提取特征，得到向量 $\boldsymbol{x}_t$。循环层把 $\boldsymbol{x}_t$ 作为输入，然后输出 $\boldsymbol{h}_t$。$\boldsymbol{h}_t$ 是从 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 中提取出的特征，是对所有观测 $\boldsymbol{o}_{1:t} = [o_1, \cdots, o_t]$ 的一个概要。全连接网络（输出层激活函数是 Softmax）把 $\boldsymbol{h}_t$ 作为输入，然后输出向量 $\boldsymbol{f}_t$，作为 t 时刻决策的依据。$\boldsymbol{f}_t$ 的维度是动作空间的大小 $|\mathcal{A}|$，它的每个元素对应一个动作，表示选择该动作的概率。

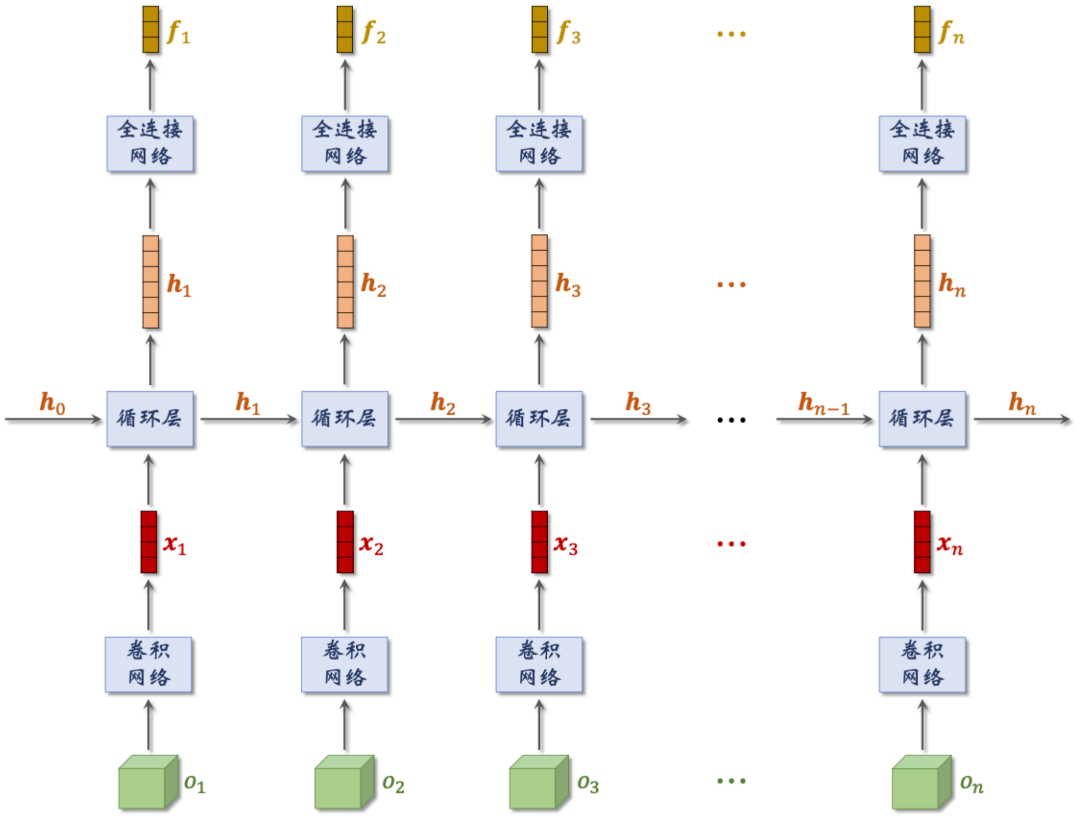

图 11.5: 基于 RNN 的策略网络。图中所有的全连接网络都有相同的参数；所有的循环层都有相同的参数；所有的卷积层都有相同的参数。

对于不完全观测问题，我们可以类似地搭建 DQN 和价值网络。DQN 可以定义为：

$$Q(\boldsymbol{o}_{1:t}, a_t;\ \boldsymbol{w}).$$

价值网络可以定义为：

$$q(\boldsymbol{o}_{1:t}, a_t;\ \boldsymbol{w}) \qquad \text{或} \qquad v(\boldsymbol{o}_{1:t};\ \boldsymbol{w}).$$

这些神经网络与图 11.5 中策略网络的区别只是在于全连接网络的结构而已；它们使用的卷积网络、循环层与图 11.5 相同。

相关文献

RNN 是一类很重要的神经网络。学术界认为最早的 RNN 是 Hopfield network [36]，尽管它跟我们今天用的 RNN 很不一样。现在最常用的 RNN 包括 LSTM [35] 和 GRU [20]。

注意力机制 (Attention) 由 2015 年的论文 [4] 提出，将 Attention 与 RNN 结合，可以大幅提升 RNN 在机器翻译任务上的表现。2017 年的论文 [73] 提出 Transformer 模型，去掉 RNN，只保留 Attention 层，在机器翻译任务上取得了远优于 RNN+Attention 的表现。2018 年的论文 [26] 提出了一种叫做 BERT 的方法，它可以在海量数据上预训练 Transformer，得到更大更强的 Transformer 模型。

2015 年的论文 [33] 首先将 RNN 应用于深度强化学习，把 RNN 与 DQN 相结合，把得到的方法叫做 DRQN。之后 RNN 常被用于解决部分观测问题，比如论文 [44, 28, 52]。

参考文献

[1] M. S. Abdulla and S. Bhatnagar. Reinforcement learning based algorithms for average cost markov decision processes. *Discrete Event Dynamic Systems*, 17(1):23–52, 2007.

[2] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning (ICML)*, 2019.

[3] L. V. Allis et al. *Searching for solutions in games and artificial intelligence*. Ponsen & Looijen Wageningen, 1994.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015.

[5] L. Baird. Residual algorithms: reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

[6] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[7] P. Baudiš and J.-l. Gailly. Pachi: State of the art open source go program. In *Advances in computer games*, pages 24–38. Springer, 2011.

[8] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

[9] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[10] S. Bhatnagar and S. Kumar. A simultaneous perturbation stochastic approximation-based actor-critic algorithm for markov decision processes. *IEEE Transactions on Automatic Control*, 49(4):592–598, 2004.

[11] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

[12] B. Bouzy and B. Helmstetter. Monte-Carlo go developments. In *Advances in computer games*, pages 159–174. Springer, 2004.

[13] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[15] M. Buro. From simple features to sophisticated evaluation functions. In *International Conference on Computers and Games*, pages 126–145. Springer, 1998.

[16] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

[17] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-Carlo tree search: A new framework for game AI. In *AIIDE*, 2008.

[18] G. Chaslot, J.-T. Saito, B. Bouzy, J. Uiterwijk, and H. J. Van Den Herik. Monte-Carlo strategies for computer Go. In *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*, 2006.

[19] G. M. J.-B. C. Chaslot. *Monte-Carlo tree search*. Maastricht University, 2010.

[20] K. Cho, B. v. M. C. Gulcehre, D. Bahdanau, F. B. H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. 2014.

[21] Y. Chow, O. Nachum, and M. Ghavamzadeh. Path consistency learning in Tsallis entropy regularized mdps. In *International Conference on Machine Learning (ICML)*, pages 979–988, 2018.

[22] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods*. SIAM, 2000.

[23] R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[24] R. Coulom. Computing "elo ratings" of move patterns in the game of Go. *ICGA journal*, 30(4):198–208, 2007.

[25] T. Degris, P. M. Pilarski, and R. S. Sutton. Model-free reinforcement learning with continuous action in practice.

In *American Control Conference (ACC)*, 2012.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional Transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[27] M. Enzenberger, M. Müller, B. Arneson, and R. Segal. Fuego: an open-source framework for board games and go engine based on monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):259–270, 2010.

[28] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, 2018.

[29] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, et al. Noisy networks for exploration. In *International Conference on Learning Representations (ICLR)*, 2018.

[30] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

[31] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, 2017.

[32] R. Hafner and M. Riedmiller. Reinforcement learning in feedback control. *Machine learning*, 84(1-2):137–169, 2011.

[33] M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.

[34] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[35] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[36] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[37] T. Jaakkola, M. I. Jordan, and S. P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural computation*, 6(6):1185–1201, 1994.

[38] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[39] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[40] K. Lee, S. Choi, and S. Oh. Sparse Markov decision processes with causal sparse Tsallis entropy regularization for reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):1466–1473, 2018.

[41] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[42] L.-J. Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[43] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of Markov reward processes: Implementation issues. In *IEEE Conference on Decision and Control*, 1999.

[44] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[45] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[47] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[48] M. Müller. Computer go. *Artificial Intelligence*, 134(1-2):145–179, 2002.

[49] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[50] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih. Combining policy gradient and Q-learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[51] D. V. Prokhorov and D. C. Wunsch. Adaptive critic designs. *IEEE transactions on Neural Networks*, 8(5):997–1007, 1997.

[52] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

[53] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[54] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. *science*, 317(5844):1518–1522, 2007.

[55] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53(2-3):273–289, 1992.

[56] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2015.

[57] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

[58] W. Shi, S. Song, and C. Wu. Soft policy gradient method for maximum entropy deep reinforcement learning. *arXiv preprint arXiv:1909.03198*, 2019.

[59] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[60] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

[61] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[62] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems (NIPS)*, 1996.

[63] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[64] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

[65] G. Tesauro and G. R. Galperin. On-line policy improvement using monte-carlo search. In *Advances in Neural Information Processing Systems*, pages 1068–1074, 1997.

[66] C. Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of statistical physics*, 52(1-2):479–487, 1988.

[67] J. N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine learning*, 16(3):185–202, 1994.

[68] J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE transactions on automatic control*, 42(5):674–690, 1997.

[69] H. J. Van Den Herik, J. W. Uiterwijk, and J. Van Rijswijck. Games solved: Now and in the future. *Artificial Intelligence*, 134(1-2):277–311, 2002.

[70] H. van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[71] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[72] H. van Seijen. Effective multi-step temporal-difference learning for non-linear function approximation. *arXiv*

preprint arXiv:1608.05151, 2016.

[73] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[74] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

[75] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[76] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[77] R. J. Williams. *Reinforcement-learning connectionist systems*. College of Computer Science, Northeastern University, 1987.

[78] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[79] R. J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[80] W. Yang, X. Li, and Z. Zhang. A regularized approach to sparse optimal policy in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5940–5950, 2019.

[81] Z. Yang, Y. Chen, M. Hong, and Z. Wang. Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8353–8365, 2019.