

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 $Q(s, a; \mathbf{w})$ ，策略网络 $\pi(a|s; \boldsymbol{\theta})$ 、 $\boldsymbol{\mu}(s; \boldsymbol{\theta})$ ，价值网络 $q(s, a; \mathbf{w})$ 、 $v(s; \mathbf{w})$ 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。



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, \dots, o_t 。把从初始到 t 时刻为止的所有观测记作：

$$\mathbf{o}_{1:t} = [o_1, o_2, \dots, o_t]$$

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

$$\pi(a_t | \mathbf{o}_{1:t}; \boldsymbol{\theta}).$$

该如何实现这样一个策略网络呢？请注意， $\mathbf{o}_{1:t}$ 的大小是变化的。如果 o_1, \dots, o_t 都是 $d \times 1$ 的向量，那么 $\mathbf{o}_{1:t}$ 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 $\mathbf{x}_1, \dots, \mathbf{x}_n$ 是一个序列。对于所有的 $t = 1, \dots, n$, 循环层把 $(\mathbf{x}_1, \dots, \mathbf{x}_t)$ 映射到特征向量 \mathbf{h}_t 。依次把 $\mathbf{x}_1, \dots, \mathbf{x}_n$ 输入循环层, 会得到:

$$\begin{aligned} (\mathbf{x}_1) &\implies \mathbf{h}_1, \\ (\mathbf{x}_1, \mathbf{x}_2) &\implies \mathbf{h}_2, \\ (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) &\implies \mathbf{h}_3, \\ &\vdots \\ (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_{n-1}) &\implies \mathbf{h}_{n-1}, \\ (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_{n-1}, \mathbf{x}_n) &\implies \mathbf{h}_n. \end{aligned}$$

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 \mathbf{h}_n 的大小是固定的。请特别注意, \mathbf{h}_t 并非只依赖于 \mathbf{x}_t 这一个向量, 而是依赖于 $[\mathbf{x}_1, \dots, \mathbf{x}_t]$; 理想情况下, \mathbf{h}_t 记住了 $[\mathbf{x}_1, \dots, \mathbf{x}_t]$ 中的主要信息。比如 \mathbf{h}_3 是对 $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$ 的概要, 而非是对 \mathbf{x}_3 这一个向量的概要。

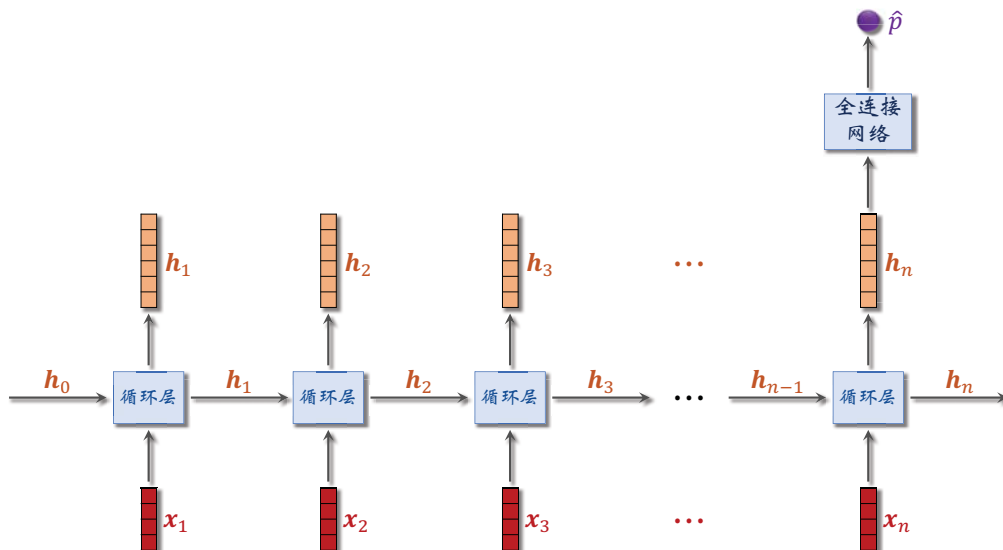


图 11.2: 输入是序列 $\mathbf{x}_1, \dots, \mathbf{x}_n$ 。向量 \mathbf{h}_n 是从所有 n 个输入中提取的特征, 可以把它看做输入序列的一个概要。把 \mathbf{h}_n 输入全连接层 (带 Sigmoid 激活函数), 得到分类结果 \hat{p} 。

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 $\mathbf{x}_1, \dots, \mathbf{x}_n$, 把它们依次输入循环层。循环层依次输出 $\mathbf{h}_1, \dots, \mathbf{h}_n$ 。我们只需要用 \mathbf{h}_n , 因为它从全部输入 $\mathbf{x}_1, \dots, \mathbf{x}_n$ 中提取的特征; 可以忽略掉 $\mathbf{h}_1, \dots, \mathbf{h}_{n-1}$ 。最后, 二元分类器把 \mathbf{h}_n 作为输入, 输出一个介于 0 到 1 之间的数 \hat{p} ,

《深度学习》2021-02-09 尚未校对, 仅供预览。
如发现错误, 请告知作者 shusen.wang@stevens.edu

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

$$\mathbf{h}_t = \tanh(\mathbf{W}[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}).$$

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 \mathbf{W} 和 \mathbf{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 \mathbf{z} ，有

$$\tanh(\mathbf{z}) = [\tanh(z_1), \tanh(z_2), \dots, \tanh(z_d)]^T.$$

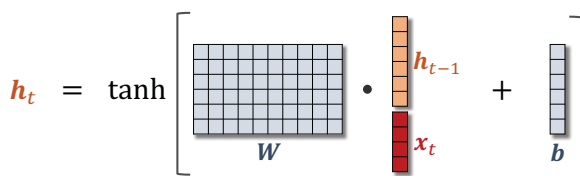


图 11.3: 简单循环层。

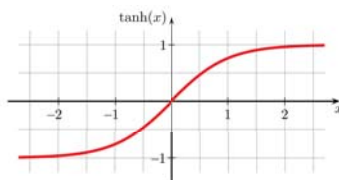


图 11.4: 双曲正切函数。

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 $\mathbf{o}_{1:t} = [o_1, \dots, o_t]$ 做决策。定义策略网络为 $\mathbf{f}_t = \pi(a_t | \mathbf{o}_{1:t}; \boldsymbol{\theta})$ ，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 \mathbf{x}_t 。循环层把 \mathbf{x}_t 作为输入，然后输出 \mathbf{h}_t 。 \mathbf{h}_t 是从 $\mathbf{x}_1, \dots, \mathbf{x}_t$ 中提取出的特征，是对所有观测 $\mathbf{o}_{1:t} = [o_1, \dots, o_t]$ 的一个概要。全连接网络（输出层激活函数是 Softmax）把 \mathbf{h}_t 作为输入，然后输出向量 \mathbf{f}_t ，作为 t 时刻决策的依据。 \mathbf{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。

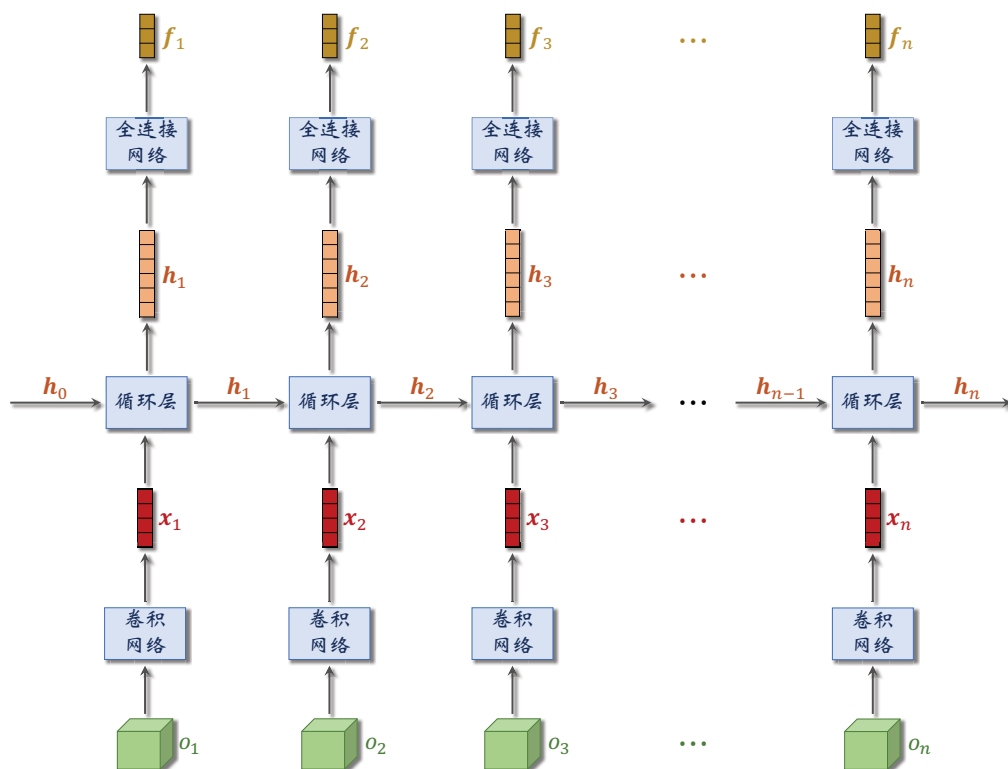


图 11.5: 基于 RNN 的策略网络。图中所有的全连接网络都有相同的参数；所有的循环层都有相同的参数；所有的卷积层都有相同的参数。

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：

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

价值网络可以定义为：

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

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。

❧ 第十一章 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 [51], 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 [50] 和 GRU [28]。

注意力机制 (Attention) 由 2015 年的论文 [6] 提出, 将注意力机制与 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 [111] 提出 Transformer 模型, 去掉 RNN, 只保留注意力机制, 在机器翻译任务上的表现远优于 RNN+ 注意力。2018 年的论文 [35] 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 [46] 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 [68, 38, 80]。

《深度强化学习》2021-02-09 尚未校对, 仅供预览。
如发现错误, 请告知作者 shusen.wang@stevens.edu