# 第6章 深度学习基础

在前面章节中我们主要介绍了传统强化学习的内容，这些内容涵盖了基础的问题核心和解决方法。但是对应的算法并不能解决高维度的复杂问题，因此现在普遍流行将深度学习和强化学习结合起来，利用深度学习网络强大的拟合能力通过将状态、动作等作为输入，来估计对应的状态价值和动作价值等等。

在过渡到深度强化学习之前，本章将对强化学习中用到的一些深度学习知识（主要包括各种神经网络等）作一个简要归纳。这些归纳主要面向已经有一些深度学习基础的读者，不会涉及很多的公式推导。如果读者对深度学习还不是很熟悉，可以对照本章的内容，去学习由笔者共同著作的另外一本书《李宏毅深度学习笔记》。

## 6.1 强化学习与深度学习的关系

之前我们讲到了强化学习的问题可以拆分成两类问题，即预测和控制。预测的主要目的是根据环境的状态和动作来预测状态价值和动作价值，而控制的主要目的是根据状态价值和动作价值来选择动作。换句话说，预测主要是告诉我们当前状态下采取什么动作比较好，而控制则是按照某种方式决策。就好比军师与主公的关系，军师提供他认为最佳的策略，而主公则决定是否采纳这个策略。

不知道读者们是否看过《超智能足球GGO》这部热血动漫，老实讲它是笔者看过比较好的带有高科技元素的足球动漫，主要讲述的是主角团带领着他们的超智能足球机器人组队打入世界大赛的故事，也是启引笔者选择强化学习的初衷之一。

如图6-1所示，其中有一队叫做英国三狮，主要领队是尼尔逊和巴菲斯，巴菲斯是一个超级数据分析专家，他能在各种场景下计算对手传球、射门的概率，也包括我方进球和传球的各种收益，然后尼尔逊会根据他的数据分析结果来决定下一步行动。尼尔逊也是一个非常有头脑的领队，他不会只依靠巴菲斯的计算结果，而是会结合自身的经验和对足球的直觉来做出数据之外的决策。这个数据之外的决策在强化学习中叫做探索，也就是说尼尔逊会根据巴菲斯的计算结果来做出决策，但是他也会根据自己的经验和直觉来做出一些不确定的决策，这样才能保证他的队伍不会被对手轻易的猜到。

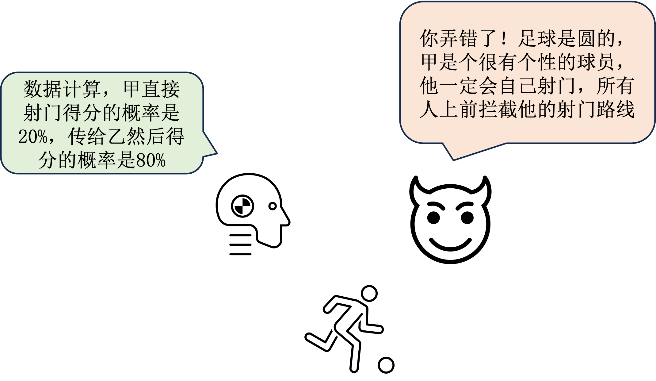


图6-1预测与控制示例

以上就是预测和控制的关系，通常在强化学习中预测和控制的部分看起来是共用一个表或者神经网络的，因此读者们可能会因为主要关注价值函数的估计而忽视掉控制这层关系，控制通常在采样动作的过程中体现出来。其实在前面也提到过，预测也相当于人的眼睛和大脑的视觉神经处理部分，而控制相当于大脑的决策神经处理部分，看似是两个独立的部分，但实际上是相互依赖的，预测的结果会影响到控制的决策，而控制的决策也会影响到预测的结果。

讲到这里，读者应该不难理解，深度学习就是用来提高强化学习中预测的效果的，因为深度学习本身就是一个目前预测和分类效果俱佳的工具。比如Q-learning的表就完全可以用神经网络来拟合。注意，深度学习只是一种非常广泛的应用，但并不是强化学习的必要条件，也可以是一些传统的预测模型，例如决策树、贝叶斯模型等等，因此读者在研究相关问题时需要充分打开思路。类似地，在控制问题中，也可以利用深度学习或者其他的方法来提高性能，例如结合进化算法来提高强化学习的探索能力。

从训练模式上来看，深度学习和强化学习，尤其是结合了深度学习的深度强化学习，都是基于**大量的样本**来对相应算法进行迭代更新并且达到最优的，这个过程我们称之为**训练**。但与另外两者不同的是，强化学习是在交互中产生样本的，是一个产生样本、算法更新、再次产生样本、再次算法更新的动态循环训练过程，而不是一个准备样本、算法更新的静态训练过程。

这本质上还是跟要解决的问题不同有关，强化学习解决的是序列决策问题，而深度学习解决的是“打标签”问题，即给定一张图片，我们需要判断这张图片是猫还是狗，这里的猫和狗就是标签，当然也可以让算法自动打标签，这就是监督学习与无监督学习的区别。而强化学习解决的是“打分数”问题，即给定一个状态，我们需要判断这个状态是好还是坏，这里的好和坏就是分数。当然，这只是一个比喻，实际上强化学习也可以解决“打标签”问题，只不过这个标签是一个连续的值，而不是离散的值，比如我们可以给定一张图片，然后判断这张图片的美观程度，这里的美观程度就是一个连续的值，而不是离散的值。

如图6-2所示，除了训练生成模型之外，强化学习相当于在深度学习的基础上增加了一条回路，即继续与环境交互产生样本。相信学过控制系统的读者很快会意识到，这个回路就是一个典型的反馈系统机制，模型的输出一开始并不能达到预期的值，因此通过动态地不断与环境交互来产生一些反馈信息，从而训练出一个更好的模型。

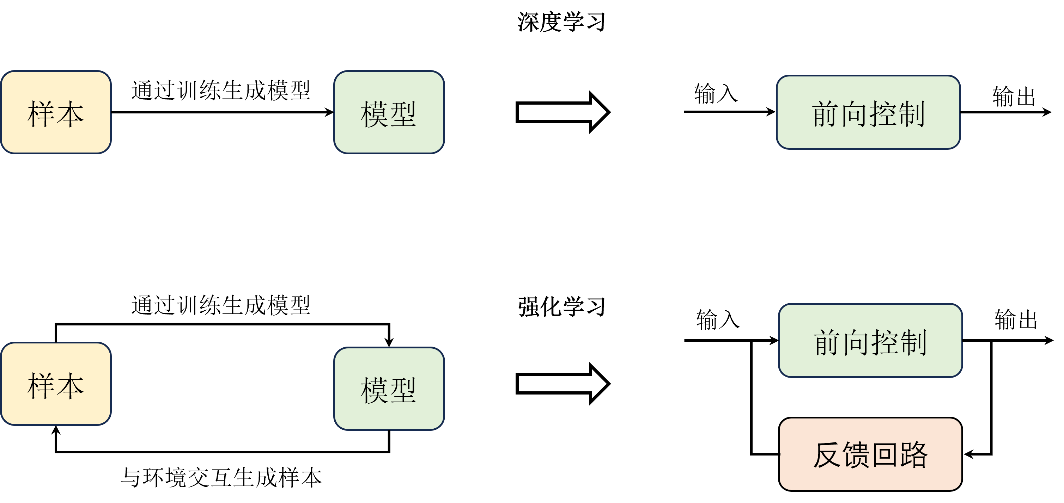


图6-2深度学习与强化学习示例

## 6.2线性回归

本小节开始总结归纳强化学习用到的一些深度学习模型，首先是线性模型。严格来说，线性模型并不是深度学习模型，而是传统的机器学习模型，但它是深度学习模型的基础，在深度学习中相当于单层的神经网络。在线性模型中，应用较为广泛的两个基础模型就是线性回归和逻辑回归，通常分别用于解决回归和分类问题，尽管后者也可以用来解决回归问题。

以典型的房价预测问题为例，假设一套房子有个特征，例如建造年份、房子面积等，分别记为 ，用向量表示为式 (6.1) 。

那么房价可以表示为式 (6.2) 。

其中和是模型的参数，是模型的输出，也就是我们要预测的房价。出于简化考虑，通常我们会用一个符号来表示和，如式 (6.3) 所示。

在这类问题中，这样的关系可以用模型来表述，我们的目标是求得一组最优的参数 ，使得该模型尽可能地能够根据房屋的个特征准确预测对应的房价。这类问题也叫做拟合问题，比如我们可以用一条直线来拟合一组散点，这条直线代表的就是模型。用来拟合最优参数的这些散点或者说数据称作样本，实际应用中由于需要拟合的模型是未知且复杂的，不可能用一个简单的函数来表示，因此需要大量的样本来训练模型，这些样本也就是训练集。

另外注意，这里是近似求解，因为几乎不可能找到一种模型能够完美拟合所有的样本，即找到最优解，甚至这个最优解不一定存在。因此，这类问题也普遍存在过拟合和欠拟合的问题，过拟合是指在训练集上表现很好，但在测试集上表现很差，而欠拟合则是指在训练集上表现很差，但在测试集上表现很好。这两种情况都是不理想的，本质上都是陷入了局部最优解的问题，因此我们有时候需要一些方法来解决这个问题，比如正则化、数据增强等。

## 6.3梯度下降

回到问题本身，这类问题的解决方法也有很多种，例如最小二乘法、牛顿法等，但目前最流行的方法还是梯度下降。其基本思想如下：

* 初始化参数：选择一个初始点或参数的初始值。
* 计算梯度：在当前点计算函数的梯度，即函数关于各参数的偏导数。梯度指向函数值增加最快的方向。
* 更新参数：按照负梯度方向更新参数，这样可以减少函数值。这个过程在神经网络中一般是以反向传播算法来实现的。
* 重复上述两个步骤，直到梯度趋近于 0 或者达到一定迭代次数。

梯度下降本质上是一种基于贪心思想的方法,它的泛化能力很强，能够基于任何**可导的函数**求解最优解。如图6-3所示，假设我们要找到一个山谷中的最低点，也就是下山，那么我们可以从任意一点出发，然后沿着最陡峭的方向向下走，这样就能够找到山谷中的最低点。这里的最陡峭的方向就是梯度方向，而沿着这个方向走的步长就是学习率，这个学习率一般是一个超参数，需要我们自己来设定。

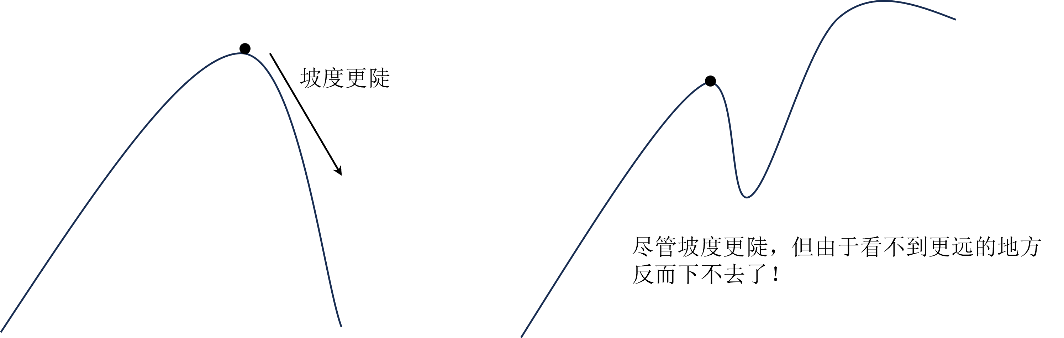


图6-3梯度下降示例

除了调整学习率和批量大小之外，我们还可以对梯度下降的机制进行一些处理，比如加入动量、Adam等，这类工具我们一般称之为优化器（optimizer）。动量法的基本思想是在梯度下降的过程中，不仅仅考虑当前的梯度，还要考虑之前的梯度，这样可以加快梯度下降的速度，同时也可以减少梯度下降过程中的震荡。

Adam是一种自适应的优化算法，它不仅仅考虑了当前的梯度，还考虑了之前的梯度的平方，这样可以更加准确地估计梯度的方向，从而加快梯度下降的速度，也是目前最流行的优化器之一。注意在做强化学习应用或研究的时候，我们并不需要太纠结于优化器的选择，因为这些优化器的效果并没有太大的差别，而且我们也不需要去了解它们的具体原理，只需要知道它们的大致作用就可以了。

此外，从训练中样本选择的方式来看，梯度下降可以分为单纯的梯度下降和随机梯度下降（stochastic gradient descent, SGD）。前者是按照样本原本的顺序不断迭代去拟合模型参数，后者则是随机抽取样本，这样做的好处就是利用随机性可能帮助算法跳出一些局部最优解，从而使得算法的收敛性更高，增强鲁棒性。从批的大小来看，又可以分为批量梯度下降和小批量梯度下降（mini-batch gradient descent），前者每次使用整个训练样本来迭代，也就是batch很大，这样做的好处是每次迭代的方向比较准确，但是计算开销比较大。

后者则每次使用一小部分样本来迭代，也就是batch很小，这样做的好处是计算开销比较小，但是每次迭代的方向比较不准确。综合来看，我们通常使用小批量的随机梯度下降mini-batch stochastic gradient descent，这样可以兼顾到所有的优点，从而使得训练更加稳定，算法效果也会更好。

## 6.4逻辑回归

简单介绍完梯度下降之后，我们就可以继续介绍一些模型了，现在是逻辑回归。注意，虽然逻辑回归名字中带有回归，但是它是用来解决分类问题的，而不是回归问题（即预测问题）。在分类问题中，我们的目标是预测样本的类别，而不是预测一个连续的值。例如，我们要预测一封邮件是否是垃圾邮件，这就是一个二分类问题，通常输出0和1等离散的数字来表示对应的类别。在形式上，逻辑回归和线性回归非常相似，如图6-4所示，就是在线性模型的后面增加一个sigmoid函数，我们一般称之为激活函数。

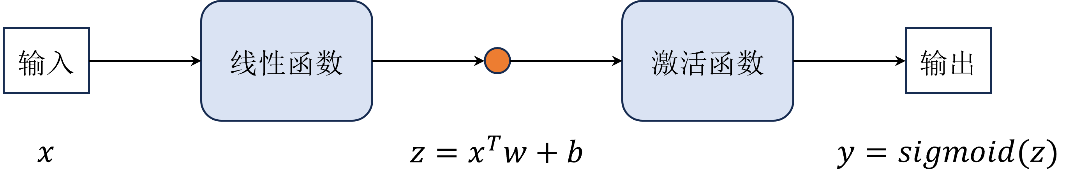


图6-4梯度下降示例

sigmoid函数定义为式 (6.4) 。

如图6-5所示， sigmoid函数可以将输入的任意实数映射到(0,1) 的区间内，对其输出的值进行判断，例如小于0.5我们认为预测的是类别0，反之是类别1 ，这样一来通过梯度下降来求解模型参数就可以用于实现二分类问题了。注意，虽然逻辑回归只是在线性回归模型基础上增加了一个激活函数，但两个模型是完全不同的，包括损失函数等等。线性回归的损失函数是均方差损失，而逻辑回归模型一般是交叉熵损失，这两种损失函数在深度学习和深度强化学习中都很常见。

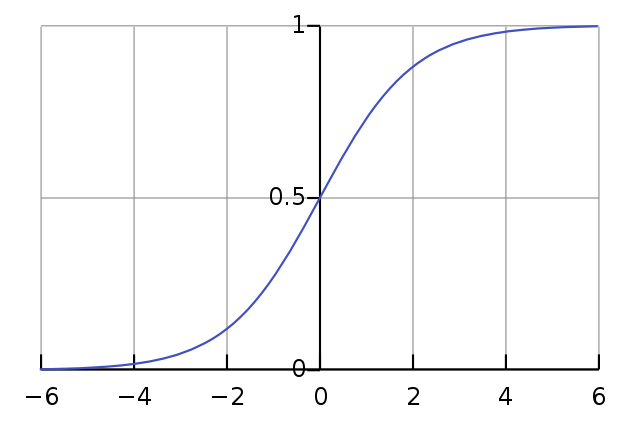


图6-5 sigmoid函数图像

逻辑回归的主要优点在于增加了模型的非线性能力，同时模型的参数也比较容易求解，但是它也有一些缺点，例如它的非线性能力还是比较弱的，而且它只能解决二分类问题，不能解决多分类问题。在实际应用中，我们一般会将多个二分类问题组合成一个多分类问题，例如将sigmoid函数换成 softmax回归函数等。

其实，逻辑回归的模型结构已经跟生物神经网络的最小单位神经元很相似了。如图 6-6所示，我们知道神经元之间是通过生物电信号来传递信息的，在每个神经元的末端会有一个叫做突触的结构，会根据信号的不同来激活不同的受体并传递给下一个神经元。当然，每个神经元也会同时接收来自不同神经元的信号并通过细胞核处理，人工神经网络中这个处理过程就相当于线性加权处理，即, 然后通过激活函数来判断是否激活。

图片包含 图表

描述已自动生成

图6-6生物神经网络与人工神经网络的对比

同时，逻辑回归这类模型的结构也比较灵活多变，可以通过横向堆叠的形式来增加模型的复杂度，例如增加隐藏层等，这样就能解决更复杂的问题，这就是接下来要讲的神经网络模型。并且，我们可以认为逻辑回归就是一个最简单的人工神经网络模型。

## 6.5全连接网络

如图 6-7所示，将线性层横向堆叠起来，前一层网络的所有神经元的输出都会输入到下一层的所有神经元中，这样就可以得到一个全连接网络。其中，每个线性层的输出都会经过一个激活函数（图中已略去），这样就可以增加模型的非线性能力。

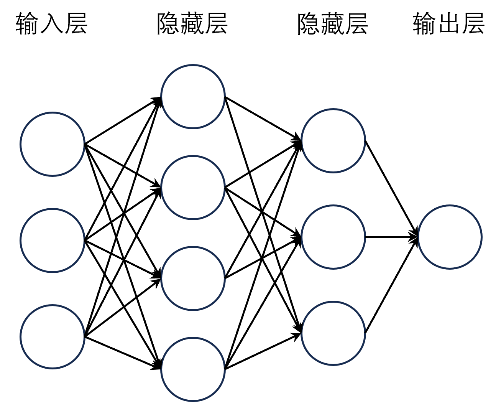


图6-7全连接网络

我们把这样的网络叫做全连接网络（fully connected network），也称作多层感知机（multi-layer perceptron，MLP），是最基础的深度神经网络模型。把神经网络模型中前一层的输入向量记为，其中第一层的输入也就是整个模型的输入可记为，每一个全连接层将前一层的输入映射到，也就是后一层的输入，具体定义为式 (6.5) 。

其中是权重矩阵，为偏置矩阵，与线性模型类似，这两个参数我们通常看作一个参数 。是激活函数，除了sigmoid函数之外，还包括softmax函数、ReLU函数和 tanh函数等等激活函数。其中最常用的是 ReLU函数 和 tanh函数，前者将神经元也就是线性函数的输出映射到 (0,1) 之间，后者则映射到-1到1之间。

前面讲到，在强化学习中我们会用神经网络来近似动作价值函数，动作价值函数的输入是状态，输出是各个动作对应的价值，在有些连续动作问题中比如汽车方向盘转动角度是-90度到90度之间，这种情况下使用tanh激活函数能够使得神经网络负值以便于更好地近似状态动作函数。顺便提一句，这里还有一种做法是我们可以把动作空间映射到正值的范围，例如(0,180)区间，这样一来对应的神经网络模型激活函数使用 ReLU函数会更好些。总而言之，激活函数的选择需要根据具体的问题来定，没有一种激活函数适用于所有的问题。

在了解到神经网络前后层的关系之后，我们就可以表示一个层的神经网络模型，如式 (6.6) 所示。

从上面的式子可以看出，神经网络模型的参数包括每一层的权重矩阵和偏置矩阵，也就是 ，这些参数都是需要我们去学习的，也就是说我们需要找到一组参数使得神经网络模型的输出尽可能地接近真实值，这个过程就是神经网络的训练过程。同基础的线性模型类似，神经网络也可以通过梯度下降的方法来求解最优参数。

## 6.6更高级的神经网络

通常来说，基于线性模型的神经网络已经足够适用于大部分的强化学习问题。但是对于一些更复杂更特殊的问题，我们可能需要更高级的神经网络模型来解决。这些高级的神经网络理论上能够取得更好的效果，但从实践上来看，这些模型在强化学习上的应用并不是很多，因为这些模型的训练过程往往比较复杂，需要调整的参数也比较多，而且这些模型的效果并不一定比基础的神经网络模型好很多。

因此，读者在解决实际的强化学习问题时还是尽量简化问题，并使用基础的神经网络模型来解决。在这里我们只是简要介绍一些常用的高级神经网络模型，感兴趣的读者可以自行深入了解。

#### 卷积神经网络

卷积神经网络（convolutional neural network，CNN}$）适用于处理具有网格结构的数据，如图像（2D网格像素点）或时间序列数据（1D网格）等，其中图像是用得最为广泛的。比如在很多的游戏场景中，其状态输入都是以图像的形式呈现的，并且图像能够包含更多的信息，这个时候我们就可以使用卷积神经网络来处理这些图像数据。在使用卷积神经网络的时候，我们需要注意以下几个主要特点：

* 局部感受野：传统的线性神经网络每个节点都与前一层的所有节点相连接。但在CNN中，我们使用小的局部感受野（例如3x3或5x5的尺寸），它只与前一层的一个小区域内的节点相连接。这可以减少参数数量，并使得网络能够专注于捕捉局部特征。
* 权重共享：在同一层的不同位置，卷积核的权重是共享的，这不仅大大减少了参数数量，还能帮助网络在图像的不同位置检测同样的特征。
* 池化层：池化层常常被插入在连续的卷积层之间，用来减少特征图的尺寸、减少参数数量并提高网络的计算效率。最常见的池化操作是最大池化（ $\text{Max-Pooling}$ ），它将输入特征图划分为若干个小区域，并输出每个区域的最大值。
* \* 归一化和dropout：为了优化网络的性能和防止过拟合，可以在网络中添加归一化层（如 batch normalization）和 dropout。

#### 循环神经网络

循环神经网络（recurrent neural network，RNN）适用于处理序列数据，也是最基础的一类时序网络。在强化学习中，循环神经网络常常被用来处理序列化的状态数据，例如在Atari游戏中，我们可以将连续的四帧图像作为一个序列输入到循环神经网络中，这样一来就能够更好地捕捉到游戏中的动态信息。但是基础的RNN结构很容易产生梯度消失或者梯度爆炸的问题，因此我们通常会使用一些改进的循环神经网络结构，例如 LSTM和 GRU等。$\text{LSTM}$ 主要是通过引入门机制（输入门、遗忘门和输出门）来解决梯度消失的问题，它能够在长序列中维护更长的依赖关系。而 GRU则是对LSTM的简化，它只有两个门（更新门和重置门），并且将记忆单元和隐藏状态合并为一个状态向量，性能与LSTM相当，但通常计算效率更高。

还有一种特殊的结构，叫做Transformer。虽然它也是为了处理序列数据而设计的，但是是一个完全不同的结构，不再依赖循环来处理序列，而是使用自注意机制 (self-attention mechanism来同时考虑序列中的所有元素。并且Transformer的设计特别适合并行计算，使得训练速度更快。自从被提出以后，Transformer就被广泛应用于自然语言处理领域，例如 $\text{BERT}$ 以及现在特别流行的GPT等模型。