2.14　本章总结

本章主要介绍了 PyTorch 深度学习框架的安装方法和一些基本概念。作为一个成熟的深度学习框架，PyTorch 提供了一系列方便的功能，包括对张量的抽象（这是几乎所有主流深度学习框架支持的基础抽象），为了方便对张量进行各种操作，PyTorch 为张量建立一个基础的 torch.tensor 类，这个类本质上是内存里的一段连续张量数据的包装。在提供张量数据包装的同时，PyTorch 给 torch.tensor 类添加了很多内置方法，能够进行一系列的操作，包括张量大小的查询、张量的下标索引、张量的切片，以及张量的维度和形状的变换等。同时，为了满足在不同的设备上对张量进行操作，PyTorch 还提供了对设备的抽象，张量能够方便地从一个设备转移到另外一个设备上。

因为张量的运算是深度学习框架的构建基础，PyTorch 也提供了丰富的函数来对张量进行运算，包括但不限于激活函数、矩阵乘法函数等。同时，在深度学习的模型构建方面，PyTorch 也提供了一系列的模块，包括 nn.module 抽象类，通过继承这个抽象类，并且在创建类的时候构建子模块，PyTorch 实现了深度学习的模块化，即可以通过组合子模块成为更大的深度学习模块，最后将所有的模块组合在一起构造深度学习模型。这个构造过程充分利用了深度学习子模块的相似性。在后续章节中会看到很多深度学习模型都是由结构相似但参数不同的子模块组合起来的，这样通过在模型中构建一系列的不同参数的子模块实例，并且把它们组合在一起，就能方便地构造复杂的深度学习模型。同时，PyTorch 的模块抽象类还提供了一系列方法，方便在模型训练和预测状态之间进行转换，以及能够快速获取一个模型所有的参数，方便优化器的调用。

除模块外，PyTorch 还定义了一系列损失函数，包括回归的损失函数和分类的损失函数，在后续章节中还将陆续介绍一些其他的损失函数，分别用于不同的深度学习任务。有了损失函数和模块、张量的概念之后，PyTorch 引入了计算图的构建和自动微分功能。前面已经有过介绍，PyTorch 使用的是动态计算图，该计算图的特点是灵活。虽然在构建计算图的时候有性能开销，PyTorch 本身的优化抵销了一部分开销，尽可能让计算图的构建和释放过程的代价最小，因此，相对静态图的框架来说，PyTorch 本身的运算速度并不慢（网上有很多相关的速度比较文章，在大多数的比较