Pytorch 介绍

你可能已经听说过图像分类，机器翻译，自动驾驶，阿法狗等一系列人工智能的成功案例

这些成功都是用神经网络实现的，也叫深度学习

神经网络相对于传统模型的神奇之处在于：传统模型使用一种算法进行特征提取，然后使用机器学习分类器，但神经网络则统一进行优化

最初的几层网络将输入转化为特征，使得其更加容易被分类

最后的积层将数据基于特征分类

我们将用pytorch库实现深度学习最核心的算法

有很多学习框架，我们用pytorch的原因在于他的简便, 强大的GPU支持，以及它自带的很多深度学习算法

计算导数和斜率是深度学习算法中十分重要的一环，pytorch十分擅长这一点

这个库也和numpy十分相似，这使得从numpy向pytorch的转变十分简便

矩阵在神经网络中也十分重要。权重和数据被以矩阵的方式进行存储，许多操作也通过矩阵乘法的方式被实现

让我们快速过一下矩阵的乘法

Pytorch中与numpy中ndarray相似的类型叫做torch tensor

你可以将tensor理解为任意维度的数组

我们可以用很多方法实现tensor的定义

我们也可以查看tensor的形状

Matmul函数实现了矩阵乘法，这和numpy中的dot函数十分相似

我们也可以定义零矩阵和单位矩阵

我们也可以通过函数进行torch tensor和numpy类型的相互转换

这里我们进行了一个罗列

前向传播算法

这是对所有分类器训练与评估的一个重要步骤

让我们看一个网络，这是许多分类器的原理，不用担心你不能完全看懂

第一步在于评估你需要对你的数据用哪一种分类器

Data是黄色的一层

模型是蓝色的两层

输出是棕色的一层

这一步叫做向前步骤

我们来看一个模型，这个模型中有abcd四个节点以及一些对应的操作

我们的操作是想得到最终的结果g

首先我们对ab相加，最终将结果放入e

然后我们对cd相乘，最终将结果放入f

然后最终将ef相乘，并将结果放入g

我们看一下如何用pytorch实现

虽然这个例子较为简单，但是通过这个来理解模型前向传播的概念十分重要

在神经网络中，这个模型会变得十分庞大并且复杂

反向传播

导数在微积分中是一个十分重要的概念，代表着一个函数变化的快慢

如果一个函数的值变得十分快，那么它斜率的绝对值就会很大，如果不变化那就是0

这也可以用函数的坡度来形容

比如在这个函数当中，ac两点的斜率很大，b点则很小

这是导数的一些公式

和导数密切相关的就是斜率，在神经网络中我们会大量的用到斜率这个概念

让我们再来看一个例子

我们有三个变量xyz

我们首先将xy相加得到q

然后我们将zq相乘得到f

F相对与它自身的导数是1

从导数的乘法原理中我们可以得出q的导数是z\*f’，而z的导数是q\*f’

而xy的导数都是q

在pytorch中，我们也可以用函数自动化这些操作

神经网络的介绍

我们将先学习全连接的神经网络，这也是最简单的一种

我们有许许多多的分类器

这些分类器在数据是向量类型的特征时表现优秀

但是很多数据通常不会提供特征，比如图像，音频以及文字

在以前，一种图像特征提取的算法时SIFT，然后提取的特征会用SVM分类器进行分类

我们可以看到问题在于，我们需要同时优化两种算法，而这两种算法又没有相互的联系

神经网络则不同，它又输入层，隐藏层，以及输出层

隐藏层为了提取特征，输出层着重于分类特征

而自始至终，我们用一种算法同时满足二者的需求，这也让从前的方法逐渐被淘汰

我们来看一下一个神经网络，这是一个全连接的网络，每一个连接成为权重，以矩阵的形式呈现

每一层都与之前一层有着联系

在pytorch中我们有一个面向对象的方法

我们定义一个class，继承于nn

在init method中我们定义了我们的参数：tensor形式的权重

最后我们使用net