Table of Contents:

今日工作

1、由于不少知识已经忘记，观看吴恩达卷积神经网络课程进行复习。

2、基于PyTorch实现课程中的神经网络示例模型。

3、学习反向传播的过程。

边缘检测

通过不同的filter能检测图片中边缘。

在图像中，不同物体之间应该是有颜色变化的，例如明暗的变换。

在卷积神经网络的训练中，反向传播时会修改filter矩阵中的参数。

每次卷积之后，得到的矩阵中的数据是原图像的特征。

如上图，一个 `39 * 39 * 3` 的图片在三次卷积之后，得到 `7 * 7 * 40` 的输出。有 `1960(7 * 7 * 40)` 个特征。之后可以进行平滑操作，将其转化成一个向量。这个向量作为 `softmax` 函数的输入，函数将这组特征映射成一个0-1之间的数。

同时，随着神经网络深度的增加，图片的宽高不断降低，信道数不断增加。

池化层

池化层也叫下采样层，英文Polling Layer。

池化层的主要作用是**保留主要的特征同时减少参数和计算量，防止过拟合，提高模型泛化能力。**

两种常用的池化层： `Max polling`， `Average pooling`。

使用最大池化的主要原因是在实验中这样做的效果很好，因此在我们的模型中，我们也可以积极尝试这一方法。

平均池化与最大池化相似，只是不是选取最大值，而是计算平均值作为区域的值。

池化层的参数不需要学习，这层只有超参数（f和s），很少使用padding超参数。（没有权重和参数）

池化分信道进行，每个信道都进行池化。

两种计算层数的方式

1、由于池化层只有超参数，不需要学习，因此把卷积层和池化层看为一层

2、池化层单独算一层

Softmax函数

引入原因

存在两类问题：

分类问题：例如图片中的人是男生还是女生。

回归问题：例如根据图像得出体重。

softmax函数主要用于分类问题，得出属于某种类别的概率。

定义式

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

含义：对于一个输入数据

```
[a1,a2,a3,a4...an]
```

其输入softmax函数后，得到

```
[y1,y2,y3,y4...yn]
```

因此 `yk` 介于0-1之间。

反向传播过程

首先，我们需要定义**损失函数**，损失函数体现了模型输出与预期输出的误差。

我们对模型的不断训练，是为了修正模型，让**损失函数值最小。**

在之前，对于反向传播过程主要有以下疑问：

1、神经网络层数可以有这么多，每层有这么多的权重（参数），我们根据最终输出得到的误差，如何影响神经网络每一层的权重？

在 https://blog.csdn.net/weixin_40446651/article/details/81516944 文章中，我对此有了一定的了解。总的来说，根据**链式求导法则**，我们可以得到每一层参数与误差的偏导值或每层的参数与其后一层参数间的偏导关系。

方向导数

> 方向导数的定义如下：

$$\frac{\partial}{\partial l} f(x_0, x_1, \dots, x_n) = \lim_{\rho \to 0} \frac{\Delta y}{\Delta x} = \lim_{\rho \to 0} \frac{f(x_0 + \Delta x_0, \dots, x_j + \Delta x_j, \dots, x_n + \Delta x_n) - f(x_0, \dots, x_j, \dots, x_n)}{\rho}$$

$$\rho = \sqrt{(\Delta x_0)^2 + \dots + (\Delta x_j)^2 + \dots + (\Delta x_n)^2}$$

> 在导数和偏导数的定义中，均是沿坐标轴正方向讨论函数的变化率。那么当我们讨论函数沿任意方向的变化率时，也就引出了方向导数的定义，即：某一点在某一趋近方向上的导数值。 通俗的解释是： 我们不仅要知道函数在坐标轴正方向上的变化率（即偏导数），而且还要设法求得函数在其他特定方向上的变化率。而方向导数就是函数在其他特定方向上的变化率。

梯度

阅读资料：

1、 https://blog.csdn.net/qq_42169668/article/details/80361203?utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~all~sobaiduend~default-1-80361203.nonecase&utm_term=%E6%A2%AF%E5%BA%A6%E6%98%AF%E5%AF%BC%E6%95%B0%E5%90%97&spm=1000.2123.3001.4430

> 梯度的定义如下：

$$gradf(x_0, x_1, \dots, x_n) = \left(\frac{\partial f}{\partial x_0}, \dots, \frac{\partial f}{\partial x_j}, \dots, \frac{\partial f}{\partial x_n} \right)$$

> 梯度的提出只为回答一个问题： 函数在变量空间的某一点处，沿着哪一个方向有最大的变化率？ 梯度定义如下： 函数在某一点的梯度是这样一个向量，它的方向与取得最大方向导数的方向一致，而它的模为方向导数的最大值。 这里注意三点： 1）梯度是一个向量，即有方向有大小； 2）梯度的方向是最大方向导数的方向； 3）梯度的值是最大方向导数的值。

梯度下降法

在变量空间的某一点处，函数沿梯度方向具有最大的变化率，那么在优化目标函数的时候，自然是沿着 **负梯度方向**去减小函数值，以此达到我们的优化目标。

> 如何沿着负梯度方向减小函数值呢？既然梯度是偏导数的集合，如下：

$$gradf(x_0, x_1, \dots, x_n) = \left(\frac{\partial f}{\partial x_0}, \dots, \frac{\partial f}{\partial x_j}, \dots, \frac{\partial f}{\partial x_n} \right)$$

同时梯度和偏导数都是向量，那么参考向量运算法则，我们在每

$$
\begin{aligned}
&Repeat\{\\
&\quad x_0 := x_0 - \alpha \frac{\partial f}{\partial x_0}\\
&\quad \cdots\cdots\cdots\cdots\\
&\quad x_j := x_j - \alpha \frac{\partial f}{\partial x_j}\\
&\quad \cdots\cdots\cdots\cdots\\
&\quad x_n := x_n - \alpha \frac{\partial f}{\partial x_n}\\
&\}
\end{aligned}
$$

个变量轴上减小对应变量值即可，梯度下降法可以描述如下：

`f` 是损失函数值， `x` 是模型中的参数。

PyTorch中的反向传播

阅读资料：

1、 https://blog.csdn.net/sinat_28731575/article/details/90342082

当张量的 `requires_grad` 参数为 `true` 时，程序将会追踪所有对于该张量的操作， 当完成计算后通过调用 `.backward()` ，自动计算所有的梯度， 这个张量的所有梯度将会自动积累到 `.grad` (gradient:梯度)属性。

因此，在代码中，每次计算梯度前需要将所有参数的梯度初始为0，因为那是上一次迭代过程的信息。

后续任务

1、今天完成了模型的定义，后续进行损失函数等的实现，完成模型的训练和测试工作。

2、继续学习相关的理论知识。