2017CS231n李飞飞深度视觉识别笔记（四）——神经网络

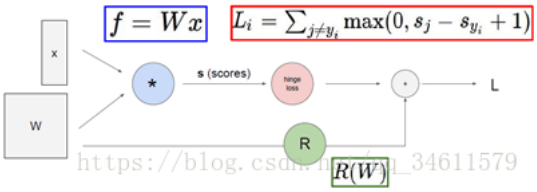
2018年07月12日 22:15:53 献世online 阅读数：405 更多

第四讲 神经网络

课时1 反向传播

目前，我们已经讲了怎么定义一个分类器、怎么定义一个损失函数以及它的正则化，也讨论了用梯度下降的方法找到最小化的损失函数。

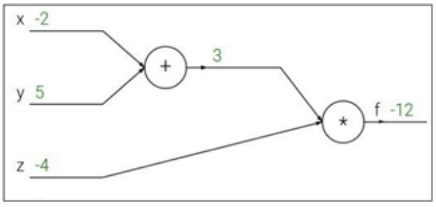
接下来，将讨论如何计算任意复杂函数的解析梯度，用到一个叫计算图的框架。大体上说，计算图就是用这类图来表示任意函数，其中图的节点表示要执行的每一步计算，比如下图中的例子：



这个线性分类器输入x和W，输出得分向量，另外一个计算节点表示hinge loss，计算数据损失项Li，同时有一个正则化项，最后输入的总的损失函数L就是正则化项与数据项的和；利用这样的计算图的好处就是能够用反向传播技术，递归地用链式法则来计算每个变量的梯度。

那么反向传播是如何工作的呢？

简单的例子：（1）假设有一个函数 $f(x,y,z)=(x+y)z$ ，要找到函数输出对应任意变量的梯度，第一步是利用计算图来表示整个函数，例如：



现在要做的是这个网络的前向传播，这里给定了每个变量对应的值，写入计算图中，最后得到的值为-12；计算对应的梯度如下：

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

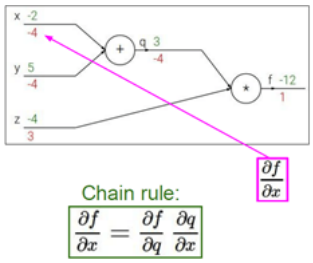
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

而反向传播是链式法则的递归调用，从后往前计算出所有的梯度。

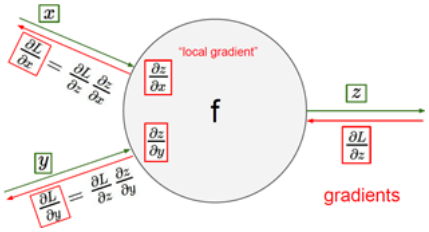
最后一个变量f的梯度为 $\frac{\partial f}{\partial f}=1$ ，接下来变量z的梯度为 $\frac{\partial f}{\partial z}=q=3$ ，变量q

的梯度为，变量y的梯度为，变量x的梯度为

。



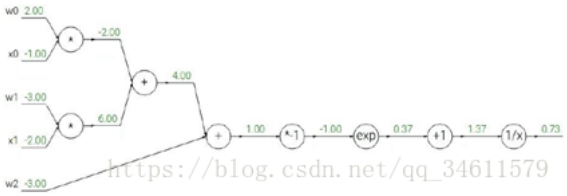
这里使用的链式法则是为了在有更复杂的计算的时候，用这种方式可以更方便的计算梯度，而在本例中比较简单的时候也可以直接计算不需要链式相乘。



主要的操作是在每个结点上计算需要的本地梯度，然后跟踪这个梯度，在反向传播过程中，接收从上游传回来的这个梯度值，直接用这个值乘以本地梯度就能得到想要传回连接点的值。

复杂的例子：（2）假设有函数 $f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$ ，同样的把它转换成

一个计算图：



图中标出就是前向传播的对应的梯度值，而现在要对它们进行反向传播，看一下它的反向计算过程：

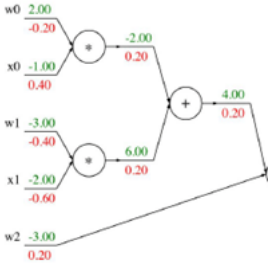
$$f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$$

$$f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$$

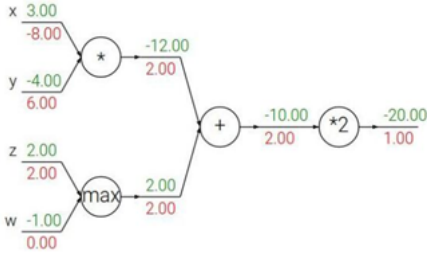
$$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$$

$$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$$

接着把剩余的梯度也给填充上去：

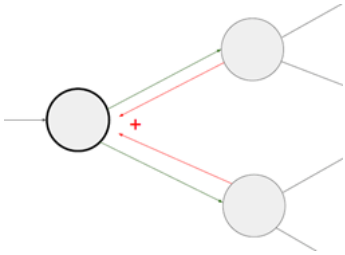


所以如果利用这样的计算图，然后应用反向传播和链式法则，就能很快的计算出所需要的梯度。



问题：对于这max运算，它的梯度值是多少？

答：z的梯度是2，w的梯度是0。其中的一个变量将会得到刚传递回来的梯度完整值，并且再传递给那个变量，然后另一个变量的梯度会取0。

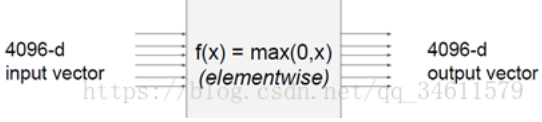


另一个需要说明的情况是上图所示，当有一个节点连接到多个节点时，梯度会在这个节点累加。在这些分支上，根据多元链式法则，只会获取每个节点的返回的上游梯度值，然后将它们加起来获得这个节点总的上游梯度。

可以这样思考，如果要改变这个节点一点点，当通过这个图进行前向传递时，它会影响在前向传递中影响到所有连接这个节点的节点，然后当进行反向传播时，所有传回的梯度都会影响到这个节点，这就是为什么将这些加起来得到回流到这个点的总上游梯度值。

接下来，讨论变量是高维的情况：

例如有一个向量作为输入，其中有4096个元素，在卷积神经网络中，这种数据尺寸是比较常见的，中间的运算节点是对每个元素求最大值的运算，最后的输出也是一个包含4096个元素的向量。



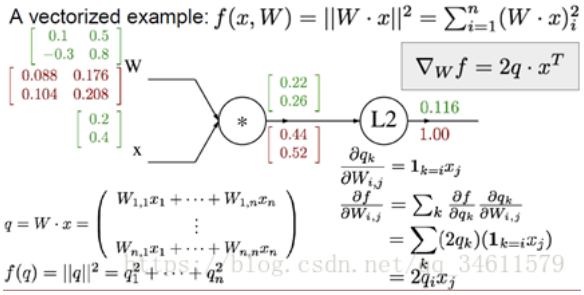
问题：这个例子中的雅克比矩阵是尺寸是几乘几的？（雅克比矩阵的每一行都是偏导数，矩阵的每个元素是输出向量的每个元素，是对输入向量每个元素分别求导的结果）

答：矩阵的尺寸是4096*4096。实际中还会遇到更大的矩阵，所以实际运算时，多数情况下并不会计算如此大的矩阵。

问题：这个雅克比矩阵的特点？

答：对角矩阵。

一个向量的例子如下图：



同样的，用相同的方法计算出$\nabla_x f = 2W^T \cdot q$。记住一个重要的事情：检查变量梯度的向量大小，应该和变量向量大小一致。

所以，我们可以将上述的前向传播和后向传播的方法模块化成一个API，如下所示：

```
class ComputationalGraph(object):
    #...
    def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward():
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```

总结：

（1）神经网络都将会是非常庞大和复杂，所以将所有参数的梯度公式写下来是不现实的；

（2）为了得到这些梯度，应该使用反向传播——神经网络中的一个核心技术就是使用反向传播来计算梯度，我们利用计算图和链式法则，从后开始计算出所有中间变量的梯度；

（3）正向：希望得到计算结果，并存储所有将会在后面的梯度计算中用到的中间值；

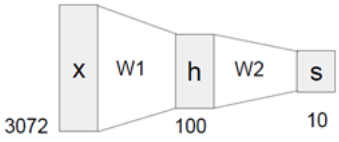
（4）反向：使用链式法则、上游梯度将它与本地梯度相乘，计算在输出节点方向上的梯度，然后将它传递给下一个连接的节点。

课时2 神经网络

在此前我们已经使用了很多这种计分函数：$f = Wx$；

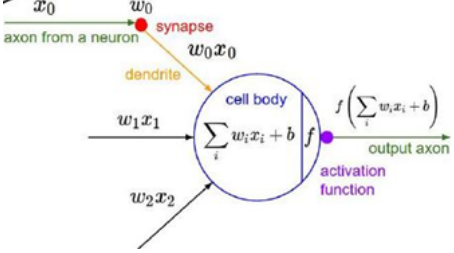
现在使用一个2层的神经网络：$f = W_2 \max(0, W_1 x)$；

或者使用一个3层的神经网络：$f = W_3 \max(0, W_2 \max(0, W_1 x))$；



一般来说，神经网络就是由简单函数构成的一组函数，使用一种层次化的方式将它们堆叠起来，形成一个更复杂的非线性函数；这也正是深度神经网络的由来，可以堆积很多层形成深度网络。

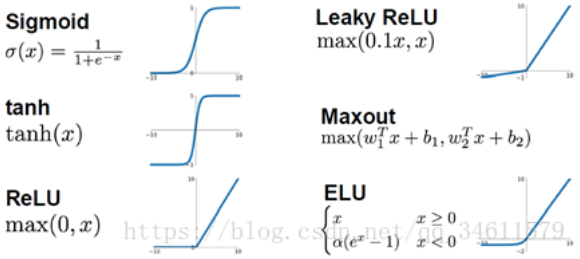
有很多人在谈论神经网络如何从生物学中获得灵感；说起神经元，每个神经元有很多树突用来接收脉冲信号，然后通过细胞体处理这些信号，接着通过轴突将处理后的信号输出；所以和神经元很类似，神经网络的结构和流程也是这样。



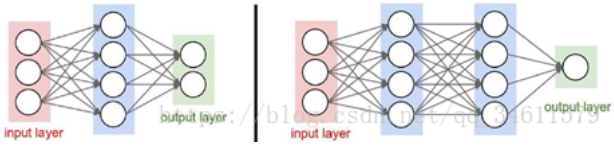
计算图里的节点相互连接，我们需要输入“信号x”，所有x的输入量比如x0、x1、x2等，采用比如赋予权重W的方法，叠加汇合到一起，将结果整合起来后得到一个激活函数，将激活函数应用在神经元的端部，得到的值作为输出。

注意：在进行这种类比时要特别小心，因为生物学上的神经元实际上比我们描述的要复杂的多，它们的树突会比表现出异常复杂的非线性，而并非像我们描述的那样只有简单的权重。

另外，提到激活函数，我们已经讨论过了多种不同的激活函数，之后我们会对所有的激活函数进行更加详细的讨论。



并且，接下来我们也将讨论神经元的不同架构形式，比如刚刚提到的2层和3层神经网络结构：



总结：

（1）本节中讨论了如何将神经元组织起来进行运算；

（2）神经元抽象的好处使我们可以采用非常高效的向量化代码进行运算；

我们已经大致了解了神经网络的一个工作的流程，类似于神经元的信号传递过程，下一章中我们将继续讨论卷积神经网络的相关内容。

RedHarden： 博主你好，有个地方好像写错了 在文中 复杂的例子：（2）..... 一个计算图：... 图中标出就是前向传播的对应的梯度值，而现在要对它们进行反向传播，看一下它的反向计算过程：... 这段话中，“图中标出就是前向传播的对应的梯度值”这里标出的应该不是对应的梯度值，绿色数字 是输入计算到输出的数值 (1个月前 #1楼)