Dilate Gated Convolutional Neural Network

Dilate Gated Convolutional Neural Network，即“膨胀门卷积神经网络”，简称“DGCNN”，从名字中可以看出，此模型融合了两个比较新的卷积用法：膨胀卷积、门卷积。此外，还融入了最近比较火的Attention的思想。由于此模型没有RNN和比较复杂的注意力交互机制，因此在速度上也是更胜一筹。以下文章将一一解析此模型中的关键技术和方法，并在最后给出总结以及实现代码。

首先解析一下模型中的门卷积结构。假设我们要处理的向量序列是X={x1,x2,x3…xn}，那么我们得出普通的一维卷积的表达式：

Y=Conv1D(X) （1）

按照一般的做法，想要加入一个门只需要将结果Y乘上一个能当作门的函数即可。假设引入一个门G，得出如下表达式：

Y=Conv1D(X) ⊗ G （2）

“⊗”表示逐位相乘。观察表达式，如果G是一个用sigmoid激活后的值，其值域为[0,1]，从直觉上来看，就是给Conv1D的每个输出都加了一个“阀门”来控制流量，这种结构也可看成一个激活函数，称为GLU（Gated Linear Unit）。在DGCNN中，门G是激活函数为sigmoid的Conv1D，因此表达式更改为：

Y=Conv1D1(X) ⊗ σ(Conv1D2(X)) （3）

注意这里的两个Conv1D形式一样（比如卷积核数、窗口大小都一样），但权值是不共享的，也就是说参数翻倍了，其中Conv1D2用sigmoid函数激活，Conv1D1不加激活函数，然后将它们逐位相乘。Conv1D1没有加任何激活函数，换句话说，这部分求导是个常数，因此梯度消失的概率会减少很多，这也是DGCNN的另一个优势。如果输入X与输出Y的维度是一样的，我们可以引入一个残差结构，将输入直接引入到输出中，表达式如下所示：

Y=X+Conv1D1(X)⊗σ(Conv1D2(X)) （4）

残差结构不仅有助于解决梯度消失使得模型的深度能够更深，还可以使信息能够在多通道传输。从模型集成的角度理解此模型的残差结构，假设有一个模型包含三层DGCNN，输入为X输出为Y，这三层DGCNN分别命名为f1、f2、f3 ，将模型展开为二叉树结构，如下图所示：

f1

f2f1

f1

f3f2f1

f2f1

f3f1

f1

f3f2

f2

f3

f2

Y

X

由上图可以很清楚的观察到，由于残差结构的存在，使得X可以直接到达Y，也可以经过多种路径到达Y，极大的丰富了模型中信息的流动方式，使信息能够在多通道传输。为了更清晰的观察信息的流向，需要将公式进一步推导。由于Conv1D1并没有加激活函数，所以它只是一个线性变换，从而可以认为Conv1D1(X)−X等效于Conv1D1。更直白的说，训练过程中，Conv1D1(X)−X能做到的事情，Conv1D1(X)也能做到。从而Conv1D1(X)−X和Conv1D1(X)两者是等价的。因此可以将(4)式改写为：

Y=X+(Conv1D1(X)−X)⊗σ(Conv1D2(X)) （5）

经过一系列推到后，(5)式变为：

Y=X⊗[1−σ(Conv1D2(X))]+Conv1D1(X)⊗σ(Conv1D2(X)) （6）

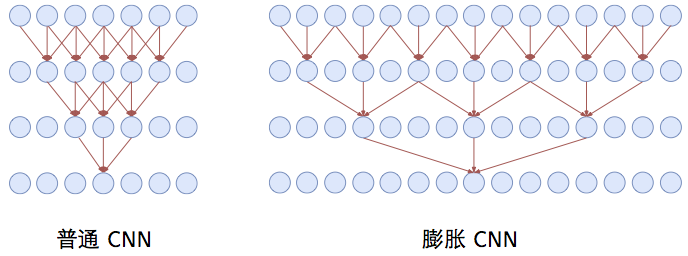
即：

Y=X⊗(1−σ)+Conv1D1(X)⊗σ （7）

σ=σ(Conv1D2(X))

从上式能够更清晰的理解信息的流向：以1−σ的概率直接通过，以σ的概率经过Conv1D2后在通过。这个形式与递归神经网络的GRU模型类似。至此，DGCNN的门结构已经推导完成，并且阐述了残差结构在DGCNN中的作用，下面将继续解释膨胀卷积。

为了使得CNN模型能够捕捉更远的的距离，并且又不至于增加模型参数，因此就有了膨胀卷积。下图为普通卷积与膨胀卷积的对比图像：



上图中的两个CNN均是三层的卷积神经网络（第一层是输入层），窗口大小为3。普通卷积每个节点只能捕捉到相邻的3个输入，与其他输入没有任何联系。但第三层的膨胀卷积可以捕捉到前后7个输入，但参数量和速度都没有变化。这是因为在第二层卷积时，膨胀卷积跳过与中心直接相邻的输入，直接捕捉中心和次相邻的输入（膨胀率为2），也可以看成是一个“窗口大小为5的、但被挖空了两个格的卷积”，所以膨胀卷积也叫空洞卷积（Atrous Convolution）。在第三层卷积时，则连续跳过了三个输入（膨胀率为4），也可以看成一个“窗口大小为9、但被挖空了6个格的卷积”。而如果在相关的输入输出连一条线，就会发现第三层的任意一个节点，跟前后7个原始输入都有联系。

按照“尽量不重不漏”的原则，膨胀卷积的膨胀率一般是按照1、2、4、8、...这样的几何级数增长。当然，这里指明了是“尽量”，因为还是有些重复的。这个比例参考了Google的wavenet模型。

def dilated\_gated\_conv1d(seq, dilation\_rate):

"""膨胀门卷积（残差式）

"""

dim= K.int\_shape(seq)[-1]

h = Conv1D(dim\*2, 3, padding='same', dilation\_rate=dilation\_rate)(seq)

def \_gate(x):

dropout\_rate = 0.1

s, h = x

g, h = h[:, :, :dim], h[:, :, dim:]

g = K.in\_train\_phase(K.dropout(g, dropout\_rate), g)

g = K.sigmoid(g)

return g \* s + (1 - g) \* h

seq = Lambda(\_gate)([seq, h])

return seq