report

编译技术 Project2 报告

小组成员：徐宁，陈楚贤，刘豪

组内分工

按照之前商定的计划，Project1由徐宁和陈楚贤完成，Project2由刘豪和另一位已退课的同学完成。最终Project2 主要由刘豪完成，徐宁，陈楚贤为解决方案思路，debug提供了大量帮助。

自动求导技术设计及例子

首先构建kernel表达式的抽象语法树。内部节点为运算符，每个节点有一个综合属性Val，表示这个节点的计算结果。 对叶子节点来说，这个计算结果就是其词法值，对内部节点来说，这个计算结果是其左子节点与右子节点的值进行该内部节点的运算得到的值。 由于例子中涉及到的运算只有加法和乘法，满足交换律，左右子节点的顺序对结果没有影响。

这是一个只含S属性的SDD，可以在自底向上的语法分析过程中实现。

再为各个节点增加一个继承属性，dVal，记录该节点获得的梯度。根节点的梯度为kernel中赋值号的左值的梯度。我们会从上到下广度优先遍历这棵语法树，将梯度向下传导至叶子节点。 其中子节点的梯度的计算公式为:

$$\frac{\partial Loss}{\partial Child} = \frac{\partial Loss}{\partial Parent} * \frac{\partial Parent}{\partial Child}$$

其中$\frac{\partial Loss}{\partial Parent}$即$Parent.dVal$，而$\frac{\partial Parent}{\partial Child}$可以根据基本求导法则得出：

父节点运算符为$+$时，

$$\frac{\partial Parent}{\partial Child} = 1$$

父节点运算符为$*$时，

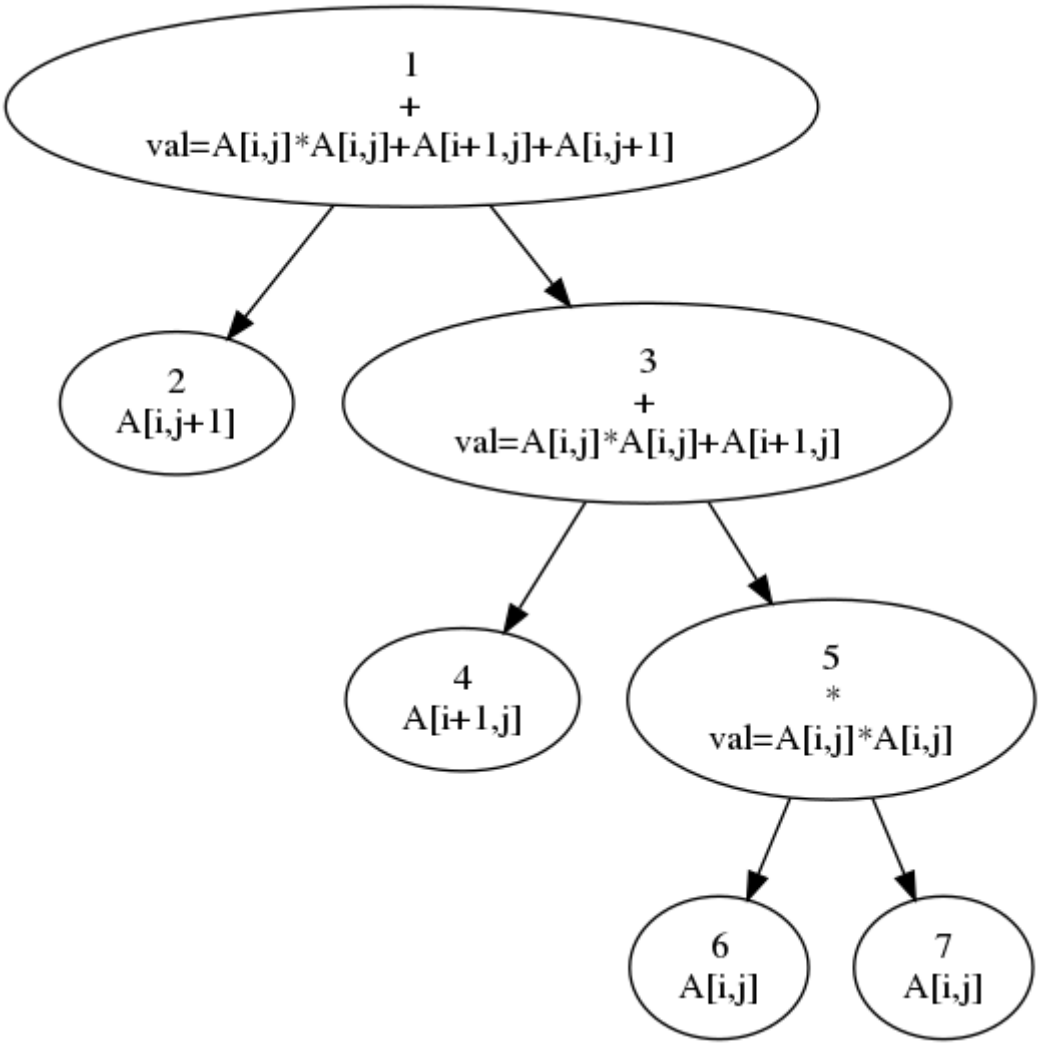
$$\frac{\partial Parent}{\partial Child} = Brother.Val$$

最后到叶子节点对要求导的变量的梯度按照对应的下标关系累加起来即可,由是，可以得到梯度的计算kernel。

以

$$B[i,j] = A[i,j] * A[i,j] + A[i+1,j] + A[i,j+1]$$

为例，自底向上构建起的语法树为:



节点1为根节点，$dVal = dB[i,j]$

节点2由于节点1的运算符为$+$，$dVal = dB[i,j] * 1$

节点3由于节点1的运算符为$+$，$dVal = dB[i,j] * 1$

节点4由于节点3的运算符为$+$,$dVal = dB[i,j] * 1 * 1$

节点5由于节点3的运算符为$+$,$dVal = dB[i,j] * 1 * 1$

节点6由于节点5的运算符为$*$,$dVal = dB[i,j] * 1 * 1 * A[i,j]$

节点7由于节点5的运算符为$*$,$dVal = dB[i,j] * 1 * 1 * A[i,j]$

由此得到：

$$dA[i, j+1] = dB[i, j] * 1$$

$$dA[i+1, j] = dB[i, j] * 1 * 1$$

$$dA[i, j] = dB[i, j] * 1 * 1 * A[i, j]$$

$$dA[i, j] = dB[i, j] * 1 * 1 * A[i, j]$$

将同一坐标的梯度分量收集起来得：

$$dA[i, j] = dB[i, j] * A[i, j] + dB[i, j] * A[i, j] + dB[i-1, j] + dB[i, j-1]$$

实现流程及结果

首先分析给出的case的kernel的结构，利用自底向上的方法生成语法树，与此同时计算各个节点的Val属性。

结束之后，自顶向下广度优先遍历语法树，计算各个节点的dVal。

根据case文件中给出的被求导对象，到叶子节点收集梯度分量，经过适当的坐标变换，得到一个导数值计算kernel。

根据这个kernel结合run2.h中给出的函数签名，生成一个符合Project1规范的json文件，并调用Project1的接口生成导数计算代码。

对于一些复杂的坐标变换，生成的kernel中需要判断下标范围，这在我们的Project1的实现中无法办到，因此最终结果通过了7个比较简单的case。

用到的编译知识

主要用到是语法树的生成，语法制导定义，以及语法制导定义的求值顺序。生成语法树这一部分，由于我们Project1的限制，导致我们只能做出比较简单的七个case，而这七个case中kernel的计算顺序是从左往右依次计算，我利用了这一点，简化了语法树的生成过程。