

# 深度学习框架中计算图的任务调度问题之一

## ——拓扑排序

魏梓轩

2019 年 9 月 8 日

## 1 计算图

在深度学习“吹牛”如日中天过后的后深度学习时代，如果你还没有听说过计算图的概念，那你真的是太难了。家喻户晓的 TensorFlow，拥有一张被各大 csdn 博主转发的动图（见下图，以 Adober Reader 食用更加），炙手可热、红极一时。

我们把图中 *Class Labels*, *Cross Entropy*, *Softmax*, *Gradients* 和 *SGD Trainer* 等节点去掉, 剩下光秃秃的与 **Forward Propagation** 相关的项目, 便是一个深度学习框架中典型的计算图了。如果非要追溯计算图的萌芽和发展过程, 我想应该不得不提算术、代数和逻辑的表示方法。那么, 具体到表示方法的名称, 大家一定觉得似曾相识——前缀表示法(波兰表示法)、中缀表示法、后缀表示法。是的, 你会自然而然的想到二叉树的三种遍历方法。而遍历方法和表示法其实是一一对应的。言外之意就是说, 一颗固定的二叉树, 可以转换为三种表示法, 并且三种表示法之间可以相互转换。对于算术或代数来说, 它们最终的结果是一致的。这一部分的内容可能会作为选择题、填空题, 出现在各大厂的笔试题目中。

那么, 二叉树和计算图又有什么关系呢? 它们的画法, 在很多书籍中都表现为一致, 即由 vertex  $\bigcirc$  和 edge  $\rightarrow$  组成的图画。以  $(x + y) * z$  为例, 我们可以画出它对应的二叉树或计算图:

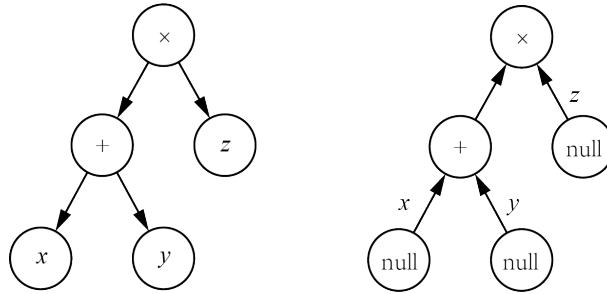


图 1:  $(x + y) * z$

上图中两个画法明显的不同在于对数值  $x, y, z$  的处理上, 一个把它们放置在  $\bigcirc$  上, 一个放置在  $\rightarrow$  上。其实, 两者并无明显不同。 $\rightarrow$  代表“取”或“送”数据的数据流, 这在两幅图中都有所体现。不同的是, 左图认为 **Scalar** 也是一种运算 **Operator**。我们可以把这种对于 Scalar 的操作, 看成是  $\exists$ 。所以, 左图中左子树可以读作:  $\exists x \& y, x + y$ 。

在计算图中, 我们要适应以下画法:

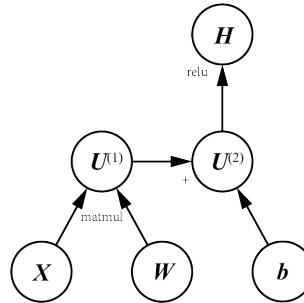


图 2:  $\mathbf{H} = \max(0, \mathbf{X}\mathbf{W} + \mathbf{b})$

又有一些不同, 但其实并无差别。如果非要找一个可以讲讲的差别, 那就是在计算图中临时变量终于拥有了姓名, 而不再是临时变量。在实际的编程中, 临时变量的创建和销毁也需要占用大量的系统资源。编写高性能运算的程序, 不得不考虑临时变量的创

建和销毁过程。关于这方面的一个简单讨论，可以阅读Expression Template Tutorial - Mshadow Guide。深度学习中的计算图站在一个比较宏观的视角来看待各个算法，是一种各个算法的组织形式，而不是算法实现的数学细节。因此，“神经网络领域里大量的研究和开发是发明计算图的新结构，而并非发明崭新的算法”。

真实应用的计算图往往比第一个动图中的还要复杂，而且不同的计算  $\bigcirc$  可能分布于不同的线程、不同的计算节点。因此，我们需要对各个  $\bigcirc$  进行调度，让他们依次准备好数据，并推送给下游的  $\bigcirc$ 。这时候，就是拓扑排序派上用场的时候了。它根据依赖关系，把各个  $\bigcirc$  执行的优先级顺序进行确定，并依次执行。这样，**下游节点计算时需要的数据，上游节点正好都准备好了**。无论在 Forward 还是 Backward 中，利用拓扑排序进行调度的方法都十分有用。

## 2 拓扑排序

拓扑排序就像是月老同志，为大家连接好了姻缘线。你喜欢他/她同时，他/她正好也喜欢你，岂不美哉。所以，这是个好算法。实现拓扑排序的方法主要有两种：**Kahn 算法**和**深度优先搜索**。Kahn 算法的伪代码如下：

```
L ← Empty list that will contain the sorted elements
S ← Set of all nodes with no incoming edge
-while S is non-empty do
  - remove a node n from S
  - add n to tail of L
  - for each node m with an edge e from n to m do
    - remove edge e from the graph
    - if m has no other incoming edges then
      - insert m into S
-if graph has edges then
  - return error (graph has at least one cycle)
-else
  - return L (a topologically sorted order)
```

深度优先搜索的伪代码如下：

```
L ← Empty list that will contain the sorted nodes
-while exists nodes without a permanent mark do
  - select an unmarked node n
  - visit(n)
```

```
-function visit(node n)
-    if n has a permanent mark then return
-    if n has a temporary mark then stop (not a DAG)
-    mark n with a temporary mark
-    for each node m with an edge from n to m do
-        visit(m)
-    remove temporary mark from n
-    mark n with a permanent mark
-    add n to head of L
```

这其中的一个坑，在上述伪代码中用**红色加粗**字体标记出来了。其实，深度优先搜索对顶点的处理有三种顺序：1) 前序；2) 后序；3) 逆后序。在拓扑排序中的这种深度优先搜索属于 3) 逆后序。

### 3 参考资料

- Graphs and Sessions | Tensorflow
- 数学表示法
- Topological sorting
- 什么是拓扑排序
- 3. 控制流与实现思路