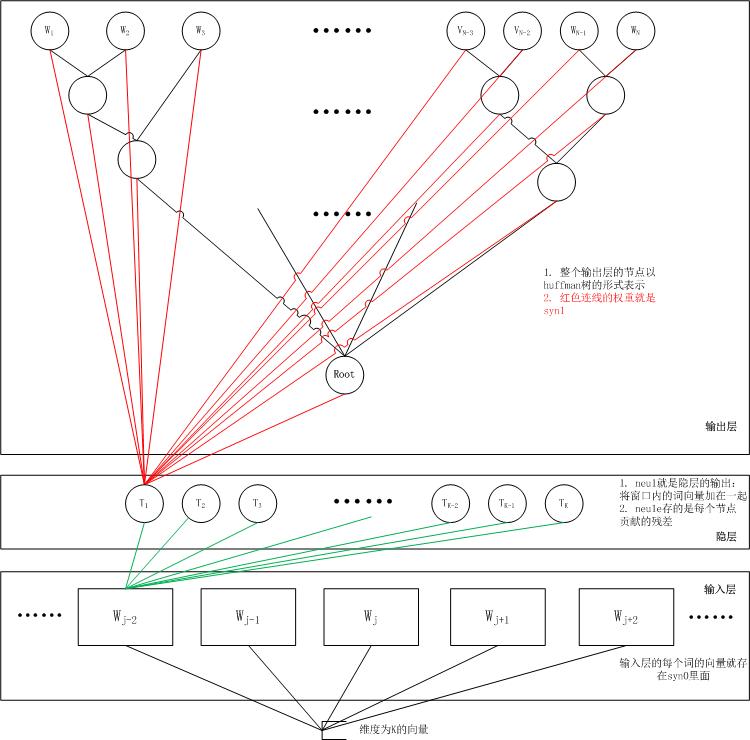
Word2vec细节

首先，它的结构就是一个三层网络——输入层、隐层（也可称为映射层），输出层。在hierarchical softmax框架中，隐层是将输入层的相加，在skip-gram中，是直接进行恒等投影，其实也可以除去。其次，代码中让人费解（没学过神经网络，是以费解）的主要是hierarchical softmax，终于弄明白其网络结果，如下图所示：

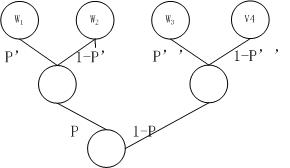
[](http://xiaoquanzi.net/wp-content/uploads/2014/02/word2vec_hs_network.jpg" \t "http://blog.csdn.net/zhoubl668/article/details/_blank)

word2vec层次softmax网络示意图

输入层读入窗口内的词，将它们的向量（K维，初始随机）加和在一起，形成隐藏层K个节点。输出层是一个巨大的二叉树，叶节点代表语料里所有的词（语料含有V个独立的词，则二叉树有|V|个叶节点）。而这整颗二叉树构建的算法就是Huffman树。这样，对于叶节点的每一个词，就会有一个全局唯一的编码，形如"010011"。我们可以记左子树为1，右子树为0。接下来，隐层的每一个节点都会跟二叉树的内节点有连边，于是**对于二叉树的每一个内节点都会有K条连边，每条边上也会有权值。**

这样，整体的结构就清晰了。在训练阶段，当给定一个上下文，要预测后面的词(Wn)的时候（word2vec的CBOW和Skip-gram都不是预测后面的词，都是在中间的词上做文章，但是本文这么写并不影响理解），实际上我们知道要的是哪个词(Wn)，而Wn是肯定存在于二叉树的叶子节点的，因此它必然有一个二进制编号，如"010011"，那么接下来我们就从二叉树的根节点一个个地去遍历，而这里的目标就是预测这个词的二进制编号的每一位！即对于给定的上下文，我们的目标是使得预测词的二进制编码概率最大。形象地说，我们希望在根节点，词向量和与根节点相连经过logistic计算得到的概率尽量接近0（即预测目标是bit=1）；在第二层，希望其bit是1，即概率尽量接近1……这么一直下去，我们把一路上计算得到的概率相乘，即得到目标词Wn在当前网络下的概率(P(Wn))，那么对于当前这个sample的残差就是1-P(Wn)。于是就可以SGD优化各种权值了。

那么hs(hierarchical softmax)如何保证叶节点输出的概率值（即我们一路沿二进制编号乘下去的概率）是归一化的呢（否则，所谓的残差1-P(Wn)就没什么意义了）？这点其实很简单，请看下图：

[](http://xiaoquanzi.net/wp-content/uploads/2014/02/word2vec_hs.jpg" \t "http://blog.csdn.net/zhoubl668/article/details/_blank)

hierarchical softmax说明

从根节点开始，对于一个sample而言，目标词是W2，二进制编码是"110"。我们在根节点计算得到它的第一位是'1'的概率是P，那么它第一位是'0'的概率就是1-P；在左子树里，第二位是"1"的概率是P'，那么第二位是"0"的概率就是1-P'，而在右子树里，第二位是"1"的概率是P''，那么第二位是"0"的概率就是1-P''；第三位亦如此。为方便表示记，我们只写到第二层。这样，在第二层，整个概率之和就是

(P\*(P') + P\*(1-P')) + ((1-P)\*(P'') + (1-P)\*(1-P'')) = P + (1-P) = 1

即按照目标词的二进制编码计算到最后的概率值就是归一化的，这也是为啥它被称作hierarchical softmax的原因。

如果没有使用这种二叉树，而是直接从隐层直接计算每一个输出的概率——即传统的softmax，就需要对|V|中的每一个词都算一遍，这个过程时间复杂度是O(|V|)的。而使用了二叉树（如word2vec中的Huffman树），其时间复杂度就降到了O(log2(|V|))，速度大大地加快了。

不过虽然hierarchical softmax一般被认为只是用于加速，但是仍然可以感性地理解一下为啥它会奏效：二叉树里面的每一个内节点实际上是一种隐含概念的分类器（二元分类器，因为二进制编码就是0/1），它的输出值的大小预示着当前上下文能够表达该隐含概念的概率，而一个词的编码实际上是一堆隐含概念的表达（注意，这个隐含概念的表达和词向量的维度所表达的隐含概念是不一样的）。我们的目标就在于找到这些当前上下文对于这些概念分类的最准确的那个表达（即目标词向量）。由于概念之间实际上是有互斥关系的（二叉树保证），即在根节点如果是"1"，即可以表达某一概念，那么该上下文是绝对不会再有表达根节点是"0"的其他情况的概念了，因此就不需要继续考虑根节点是"0"的情况了。因此，整个hierarchical softmax可以被看作完全不同于传统softmax的一套。

源码细节：

1，sigmoid函数涉及的e指数函数的就算事先将结果做成一个表，通过查表来减少运算量，提高运算效率，因为e指数直接计算时都是幂级数展开取其前n项近似。展开项越多，计算量越大。

1. 通过哈希来存储词典，通过线性探测的开放定址法来解决冲突
2. 低频词的处理，设置阈值参数，当小于这个参数的时候不计入词典中
3. 高频词也跟低频词一样处理
4. 定义上下文context（w）时，采用的是在词w的前后各取c个词，但在word2vec中，是先设置一个窗口阈值参数window（默认为5），每次构造context（w）时，首先生成区间[1,window]中的一个随机数c^，然后前后各取c^个词构成context（w）
5. 隐藏层采用的是输入层的初始向量直接相加，而不是神经概率模型中的词向量拼接，好处1 是使得向量更短，好处2是，对于一行的首尾的某些词，其前后的次数没有c个，如果是拼接就要填充向量，但是相加就不用了，只是少了几项求和项
6. 自适应学习率，首先预设一个初始的学习率a0，然后处理完10000个词后，按照公式a=a0（1-处理完的词数/（所有的词数+1）），同时设置了最小的学习率阈值，防止学习率随着训练的进行变得过小
7. 模型采用的是随机梯度上升法，只对语料遍历一遍，使得很高效，词向量随机初始化
8. 支持多线程并行