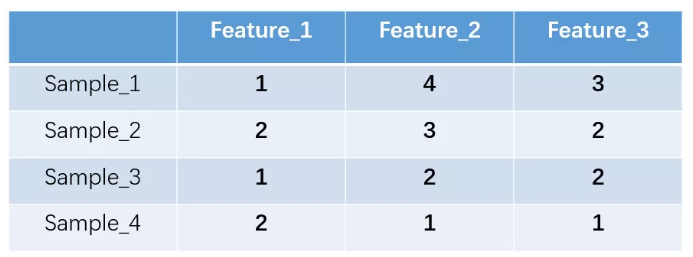
**Word2vec**

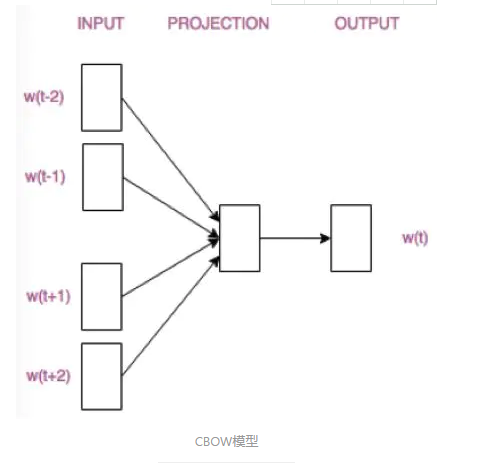
tf，tf-idf以及fast-text 已经介绍过了，那我们这次来说一下Word2vec，首先我们说下tf，tf-idf的缺陷，首先我们说下one-hot编码。One-Hot 编码，又称一位有效编码，其方法是使用N位状态寄存器来对N个状态进行编码，每个状态都有它独立的寄存器位，并且在任意时候，其中只有一位有效。举个例子，假设我们有四个样本（行），每个样本有三个特征（列），如图：

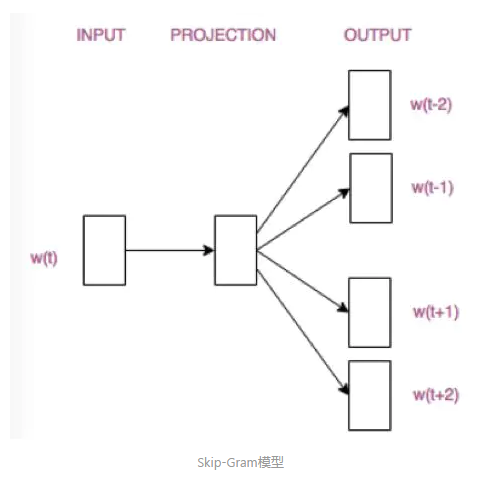


其实tf，tf-idf跟这种编码方式差不多，都是以one-hot编码为基础，tf算的是词频，而tf-idf是在词频的基础上，算这个词的重要性，但都有致命的缺点：1.忽略了上下文的顺序关系；2.每个向量之间都是独立的，如果想算这两个词地相似性是不行的，因而我们需要更加高级一点的词的表现关系Word2vec。

Word2vec其实是一种工具，现有主流的两个方法CBOW，skip-gram

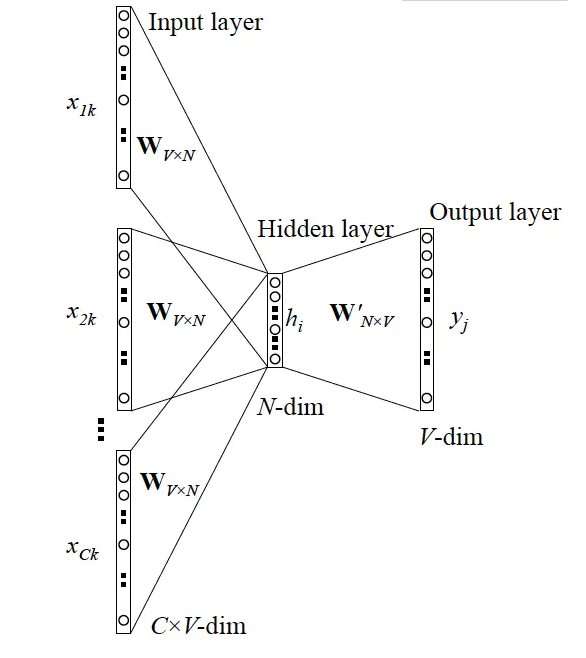
CBOW模型的训练输入是某一个特征词的上下文相关的词对应的词向量，而输出就是这特定的一个词的词向量。　Skip-Gram模型和CBOW的思路是反着来的，即输入是特定的一个词的词向量，而输出是特定词对应的上下文词向量。CBOW对小型数据库比较合适，而Skip-Gram在大型语料中表现更好。





**CBOW(Continuous Bag-of-Words)**

CBOW的训练模型如图所示



1 输入层：上下文单词的onehot. {假设单词向量空间dim为V，上下文单词个数为C}

2 所有onehot分别乘以共享的输入权重矩阵W. {VN矩阵，N为自己设定的数，初始化权重矩阵W}

3 所得的向量 {因为是onehot所以为向量} 相加求平均作为隐层向量, size为1N.

4 乘以输出权重矩阵W' {NV}

5 得到向量 {1V} 激活函数处理得到V-dim概率分布 {PS: 因为是onehot嘛，其中的每一维斗代表着一个单词}

6 概率最大的index所指示的单词为预测出的中间词（target word）与true label的onehot做比较，误差越小越好（根据误差更新权重矩阵）

所以，需要定义loss function（一般为交叉熵代价函数），采用梯度下降算法更新W和W'。训练完毕后，输入层的每个单词与矩阵W相乘得到的向量的就是我们想要的词向量（word embedding），这个矩阵（所有单词的word embedding）也叫做look up table（其实聪明的你已经看出来了，其实这个look up table就是矩阵W自身），也就是说，任何一个单词的onehot乘以这个矩阵都将得到自己的词向量。有了look up table就可以免去训练过程直接查表得到单词的词向量了。

**Skip-Gram**

从直观上理解，Skip-Gram是给定input word来预测上下文。

接下来我们来看看如何训练我们的神经网络。假如我们有一个句子“The dog barked at the mailman”。

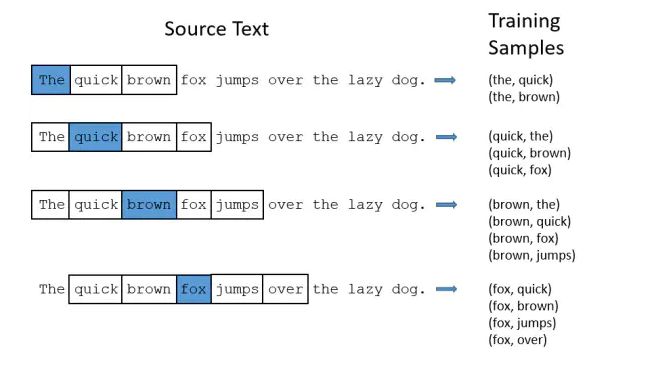
首先我们选句子中间的一个词作为我们的输入词，例如我们选取“dog”作为input word；

有了input word以后，我们再定义一个叫做skip\_window的参数，它代表着我们从当前input word的一侧（左边或右边）选取词的数量。如果我们设置skip\_window=2，那么我们最终获得窗口中的词（包括input word在内）就是['The', 'dog'，'barked', 'at']。skip\_window=2代表着选取左input word左侧2个词和右侧2个词进入我们的窗口，所以整个窗口大小span=2x2=4。另一个参数叫num\_skips，它代表着我们从整个窗口中选取多少个不同的词作为我们的output word，当skip\_window=2，num\_skips=2时，我们将会得到两组 (input word, output word) 形式的训练数据，即 ('dog', 'barked')，('dog', 'the')。

神经网络基于这些训练数据将会输出一个概率分布，这个概率代表着我们的词典中的每个词是output word的可能性。这句话有点绕，我们来看个栗子。第二步中我们在设置skip\_window和num\_skips=2的情况下获得了两组训练数据。假如我们先拿一组数据 ('dog', 'barked') 来训练神经网络，那么模型通过学习这个训练样本，会告诉我们词汇表中每个单词是“barked”的概率大小。

模型的输出概率代表着到我们词典中每个词有多大可能性跟input word同时出现。举个栗子，如果我们向神经网络模型中输入一个单词“中国“，那么最终模型的输出概率中，像“英国”， ”俄罗斯“这种相关词的概率将远高于像”苹果“，”蝈蝈“非相关词的概率。因为”英国“，”俄罗斯“在文本中更大可能在”中国“的窗口中出现。我们将通过给神经网络输入文本中成对的单词来训练它完成上面所说的概率计算。

面的图中给出了一些我们的训练样本的例子。我们选定句子“The quick brown fox jumps over lazy dog”，设定我们的窗口大小为2（window\_size=2），也就是说我们仅选输入词前后各两个词和输入词进行组合。下图中，蓝色代表input word，方框内代表位于窗口内的单词。Training Samples（输入， 输出）



我们的模型将会从每对单词出现的次数中习得统计结果。例如，我们的神经网络可能会得到更多类似（“中国“，”英国“）这样的训练样本对，而对于（”英国“，”蝈蝈“）这样的组合却看到的很少。因此，当我们的模型完成训练后，给定一个单词”中国“作为输入，输出的结果中”英国“或者”俄罗斯“要比”蝈蝈“被赋予更高的概率。

再次提醒，最终我们需要的是训练出来的权重矩阵。

打卡完毕！！！！

参考资料：

https://www.jianshu.com/p/471d9bfbd72f