This is a note for the paper: **Distributed Representations of Words and Phrases and their Compositionality**(2013)

问题背景

语言学中有一个著名的假设distributional hypothesis：具有相同上下文的单词具有相似的属性或语义。

skip-gram是根据这一假设将单词（one-hot向量）表示为稠密词向量的典型方法。skip-gram会根据上下文单词来进行梯度反向传播，更新当前单词的向量（通过上下文来塑造当前词向量）。

cbow则根据上文或下文或上下文来表示当前单词。cbow会根据当前单词来进行梯度反向传播，更新上下文单词的向量。

skip-gram可以看作特殊的cbow，即只用上文单词或下文单词来预测自己。在随机梯度下降时选择样本和更新策略还有所不同（skip-gram用将上下文的平均进行更新，cbow是同时更新上下文）。

学习技巧

输入one-hot向量经过线性运算得到词向量，再经过线性运算加softmax层得到输出。

两个线性运算分别对应输入词向量和输出词向量。

由于词典较大($10^5 \sim 10^6$)，softmax层的运算和线性运算复杂度正比于词典大小，其开销很大。目前有两种技巧解决这一问题。

负采样

负采样（negative sampling）的想法最为简单。计算输出单词时，实际上是比较中间得到的词向量和哪个输出词向量最近，用内积的指数计算相似的概率，这个概率分布和上下文的概率分布一致时损失函数最小。（对于skip-gram，目标函数为什么不是上下文概率分布，而是用词对？）。负采样就是采少量的负样本和当前词向量进行内积运算再输出概率。对这部分输出向量进行更新，而不是更新所有。

分层softmax

类似于二分搜索，搜索输出概率最大的词。先对词进行排序（随便排），构造一棵二叉搜索树，每个叶节点代表一个词，内节点不代表词。每次进行抉择（分叉时），用当前词向量和分叉向量进行内积，计算相似度，此相似度为向一侧前进的概率。这里相似度采用了sigmoid函数（直接用余弦相似度或其它的？）。从根到叶节点的路径概率乘积为这个叶节点代表词的概率。拟合参数时，在路径上进行反向传播。实际中为了节省计算量，可以构造huffman树。参数量是不变的，参数由输出向量变为了分叉向量。

短语学习时先做分词（即将短语看成词）

学习到的词向量

学习到的词向量具有很有趣的特性。比如Volga River(伏尔加河，俄罗斯母亲河) = Russian+river。为什么会这样呢？我感觉Volga River = (Russian+river)/2更合理些。