推荐系统从零单排系列(五)—Word2Vec理论与实践(下)

原创 可爱又迷人的反派角色宁宁 机器学习荐货情报局 2019-05-13

- 2019年已经过去132天-

> 【导读】上一篇推荐系统从零单排系列(四)—Word2Vec理论与实践(上)中介绍了CBOW模型，是指根据上下文来预测当前词；今天介绍的Skip-gram模型则是利用当前词来预测上下文单词。同样，本文将使用Softmax最为输出层激活函数，这是最简单最原始的Word2Vec做法。相比于Softmax，使用层次softmax或Negative Sampling可以将计算复杂度由O(V)降至O(logV)，将在下一篇中介绍，敬请期待~

Skip-gram模型理论

模型结构

Skip-gram模型最原始的形式其实非常简单，但是一些优化方法，再加上抄来抄去的文章让大家学习的不是很系统，造成了很多混轮，误以为很难。我们先从最基本的模型讲起，然后一步一步给出最终优化后的形式。

Word2Vec最终是为了得到word的低维度实数向量表示，为了这个目的，我们从原始语料中构造了一个假的分类问题。CBOW是给出上下文Context，然后预测目标词；Skip-gram正好反过来，是给出目标词，然后预测上下文。但是两者都有个共同点就是，input->hidden weights就是我们想要的Embedding权重矩阵，每一行对应一个单词的embedding vector / word2vec / 低维度实数向量表示。

语言概率模型是指给出一个句子，然后判断这个句子是人话的概率。即每个单词组合在一起是正确的概率。Skip-gram是指根据一个当前词预测多个上下文单词，是一对多的关系。如下图所示：

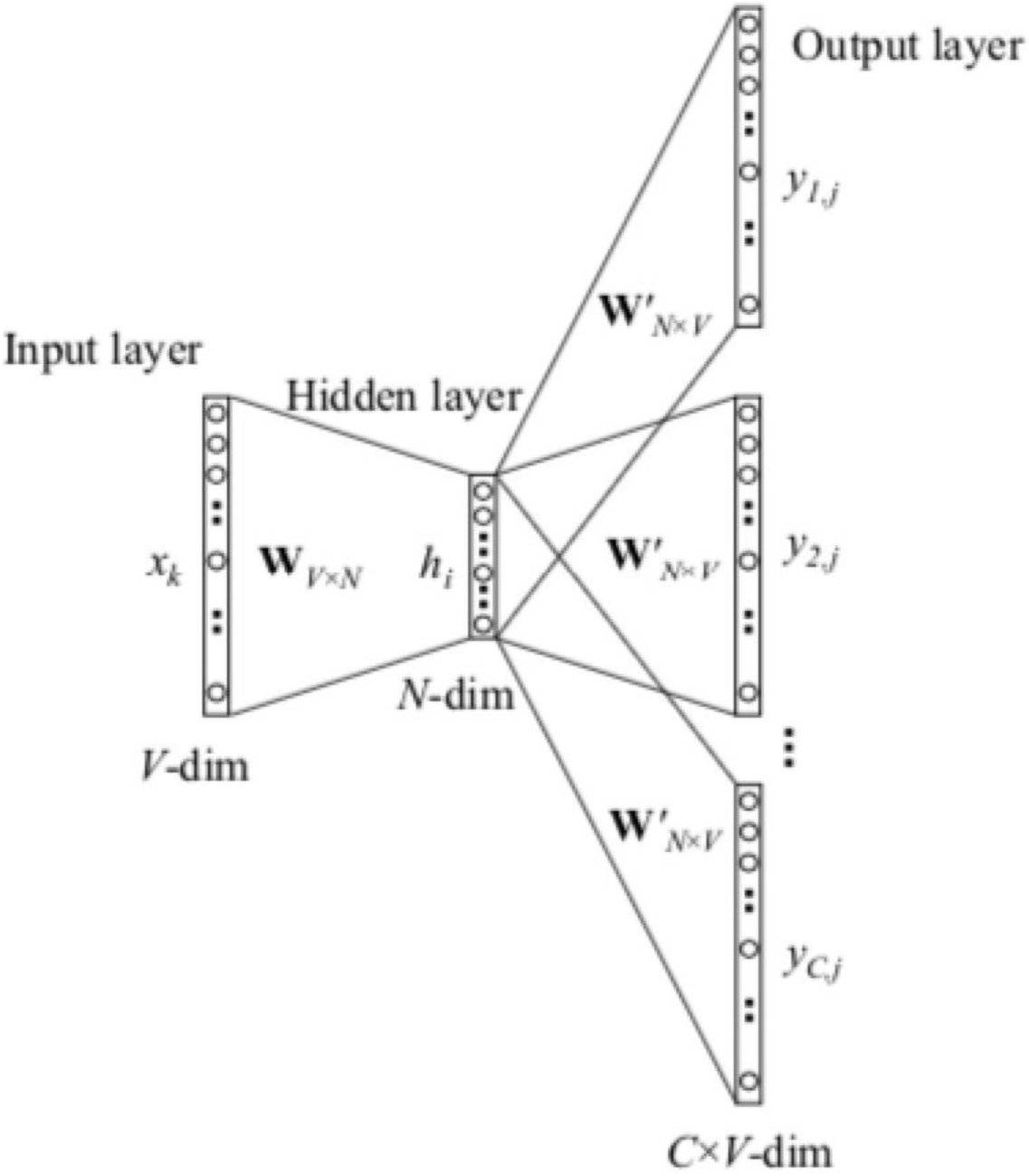


Figure 3: The skip-gram model.

模型其实非常简单，需要注意的主要有以下三点：

1. 输出是当前单词的one hot向量。跟CBOW一样，input -> hidden weights就是Embedding矩阵，V行N列，每一行对应字典中一个单词的嵌入向量，在Word2Vec中也叫做输入单词的**input vector**；

2. 中间隐藏层没有激活函数，只是线性投影；

3. 中间隐藏层到输出层共享权重矩阵 `W‘ shape=(N,V)` ，注意与input -> hidden权重矩阵是两个不同的矩阵。 `W’` 中每一列也对应一个单词，被称为该单词的**output**

vector；

4. 输出层将**input vector**和**output vector**进行点乘，然后使用softmax进行归一化，就得到了在当前单词输入下，输出是对应单词的概率；

注意：上图并不是网络结构！Skip-gram在训练的时候，输入输出都只是一个单词，和CBOW中one context的网络结构是相同的。惊不惊喜，意不意外。。。

可能有小伙伴又问了，既然同一个中心词，权重矩阵W W'也没有发生变化（共享的），那么是不是针对不同的上下文单词训练样本，每一次网络的输出向量都是一样的那？而上下文单词又是不同的，这难道不会给模型的学习带来混乱吗（到底想让模型输出什么那）？训练完之后，输入中心词，模型又会预测哪个上下文单词那？

先放答案：

1. 同一个中心词，权重是不变的，针对不同上下文单词，网络每次输出向量当然是相同的；但是在论文给出的实现代码中，每一个训练样本之后都会更新一次权重，所以权重不是不变的哦
2. 不会给网络造成混乱。因为模型的目标函数是将整体的交叉熵降到最低，模型会找到一个中间的平衡点，使得loss最低
3. 训练完之后，输入中心词，输出单词跟语料中每个单词的统计次数有关。虽然同一个中心词对应不同的上下文，但是有的上下文单词出现的多，有的出现的少，它们会影响loss的计算。毫无疑问，为了降低loss，模型倾向于预测出现较多的上下文单词。其实并不不关心预测哪个单词，关心的只是使得loss最小的训练后模型的input -> hidden权重。

重要的事情再说一遍：上图中并不是网络结构，只是示意图罢了！Skip-gram每一个训练样本都是一对单词，其网络结构和one context的CBOW模型是一样的。

训练样本生成

下面，用过例子说明下上述整个过程：

输入语料可以是一句话，或者一段话，或者一篇文章。假设语料为一句话**the quick brown fox jumped over the lazy dog** 设window为1，即我们只考虑上下各一个单词，那么训练样本则是:

```
quick -> (the, brown)
brown -> (quick, fox)
```

```
fox -> (brown, jumped)
...
```

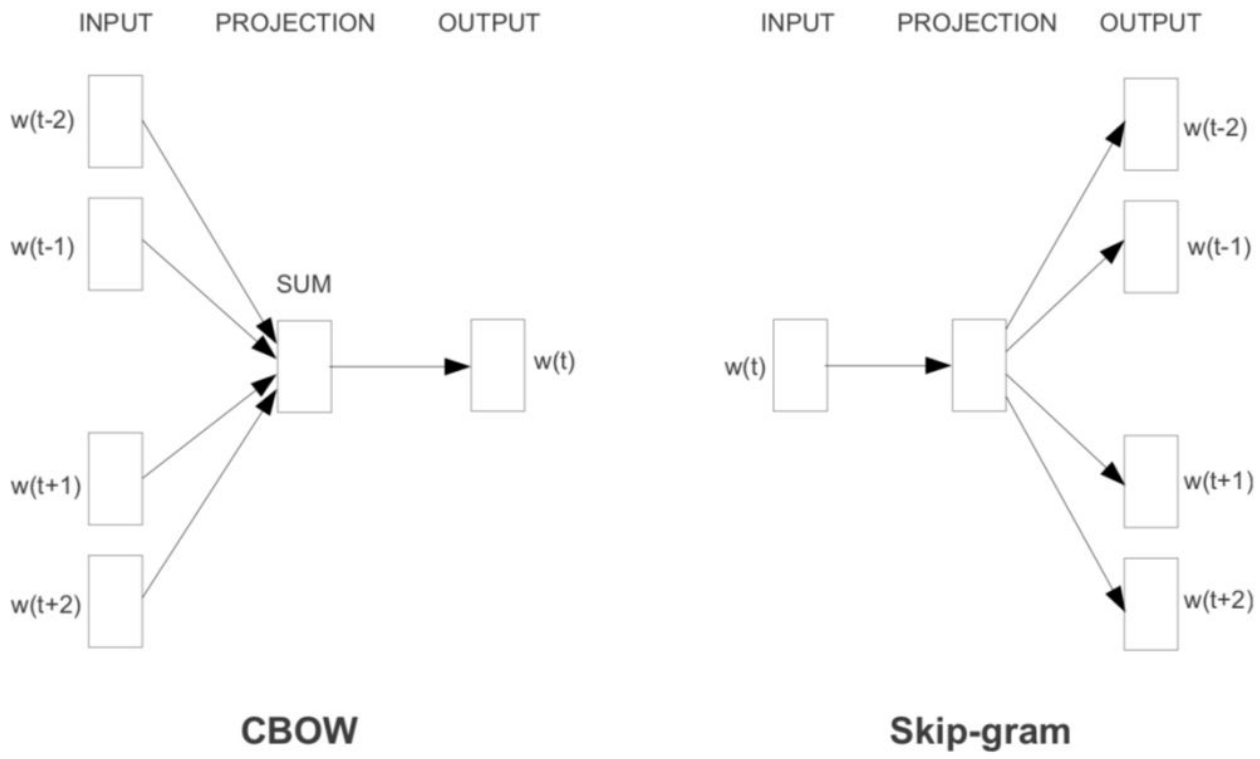
前面我们也说了，Skip-gram训练样本是一对一对的，所以实际生成的训练样本是这样的：

```
// 这才是实际的训练样本
(quick, the)
(quick, brown)

(brown, quick)
(brown, fox)

(fox, brown)
(fox, jumped)

。。。
```

CBOW vs Skip-gram

关于这两者的区别：

1. 首先CBOW是利用上下文预测中心词，Skip-gram是利用中心词预测上下文
2. Skip-gram效果比CBOW好。 — 论文作者Mikolov自己说的
3. Skip-gram训练时间长，但是对低频词(生僻词)效果好；CBOW训练时间短，对低频词效果比较差。



两者结构对比如上图所示。关于第三点多深挖一下，为什么会这样那?

CBOW是利用上下文预测中心词，训练过程是从输出的中心词的loss来学习上下文的词向量。V个中心词，对应V个训练样本，一共学习V次就结束了。训练复杂度O(V)，训练时间较快。而且上下文词向量是取得平均值，一视同仁的对待，那么低频词训练的少，而且也没有特殊处理，当然效果不好。

Skip-gram利用中心词预测上下文，假设我们考虑前后共K个上下文，那么每一个上下文都会修正一次中心词的词向量表达，训练复杂度是O(KV)训练时间会变长。但是低频词也有多个上下文，相比于CBOW，其词向量会被多次修正，自然效果也就好一些。

在通俗一点来讲，CBOW是一个老师多个学生，每个学生平等对待，学生能学多少，看你上了多少次老师的课（作为上下文被修正了多少次）。Skip-gram是多个老师一个学生，即使这个学生出现的次数很少，但是每次上课都是多个老师在教他，自然就学的多，从整个训练的角度来看，自然花费的时间也就长，毕竟每个学生都要被多个老师教一遍。

> 大家一定要把这些细节理清楚，有同学在情报局群里反映面试就被问到了。答上来答到点子上自然就加分，答不上来的话，可能就给面试官留下了一个没有技术深度的感觉，怀疑你是掉包侠哦。有时候offer就在一两个关键问题的回答上。

> 没准你今天认真看了小编的文章，回头面试被问到就回答上来了那。

Skip-gram模型实践

看了上面的分析，是不是感觉最naive的Skip-gram真的非常简单。其实单纯的网络结构和CBOW中one context是一样的，只是训练样本由(context, target)换成了（target，context）感兴趣的小伙伴去看看之前的文章。

我们把Skip-gram真正的实现留到Word2Vec优化的文章中再写。有了Negative Sampling和Hierarchical Softmax才算是真的模型实践。

Reference

Efficient Estimation of Word Representations in Vector Space

论文地址：https://arxiv.org/pdf/1301.3781.pdf

往期回顾

推荐系统从零单排系列(四)--Word2Vec理论与实践(上)
推荐系统从零单排系列(三)--再谈亚马逊Item-based推荐系统
推荐系统从零单排系列(二)--Item-Based协同过滤算法
推荐系统从零单排系列(一)--Deep Neural Network for YouTube Recommendations

如果觉得还可以，点个在看呗~
如果还想看更多内容，关注一波呗~