2022-04-22 -> 2022-05-25

1 word2vec

> word2vec原理(一) CBOW与Skip-Gram模型基础 - 刘建平Pinard - 博客园 (cnblogs.com)
>
> word2vec Parameter Learning Explained

1.1 continuous bag-of-word (CBOW) model

1.1.1 One-word context (“bigram” model)

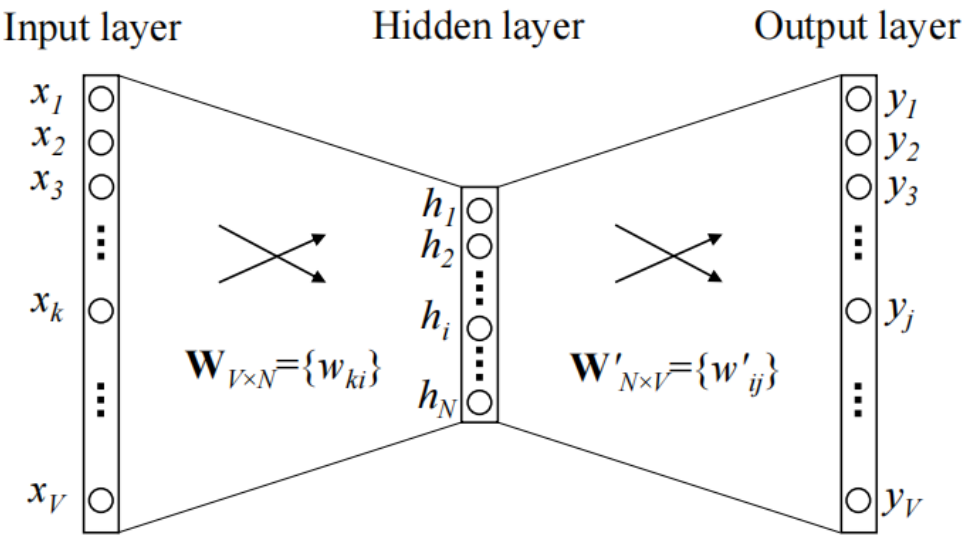

Figure 1: A simple CBOW model with only one word in the context

`V` 表示单词的个数,`N` 表示embedding dim. Input layer 实际上是 one-hot 编码，若输入为第i个单词，则除了第x_i为1外，其余全为0. Hidden layer 就是embedding的结果，没有非线性激活函数. Output layer 是$W'^T h \triangleq u$，再对 `u` 进行softmax，才得到概率 `y`，优化目标就是使中心词 `j` 对应的概率y_j越大越好，即$\max \log y_j$

1.1.2 Multi-word context

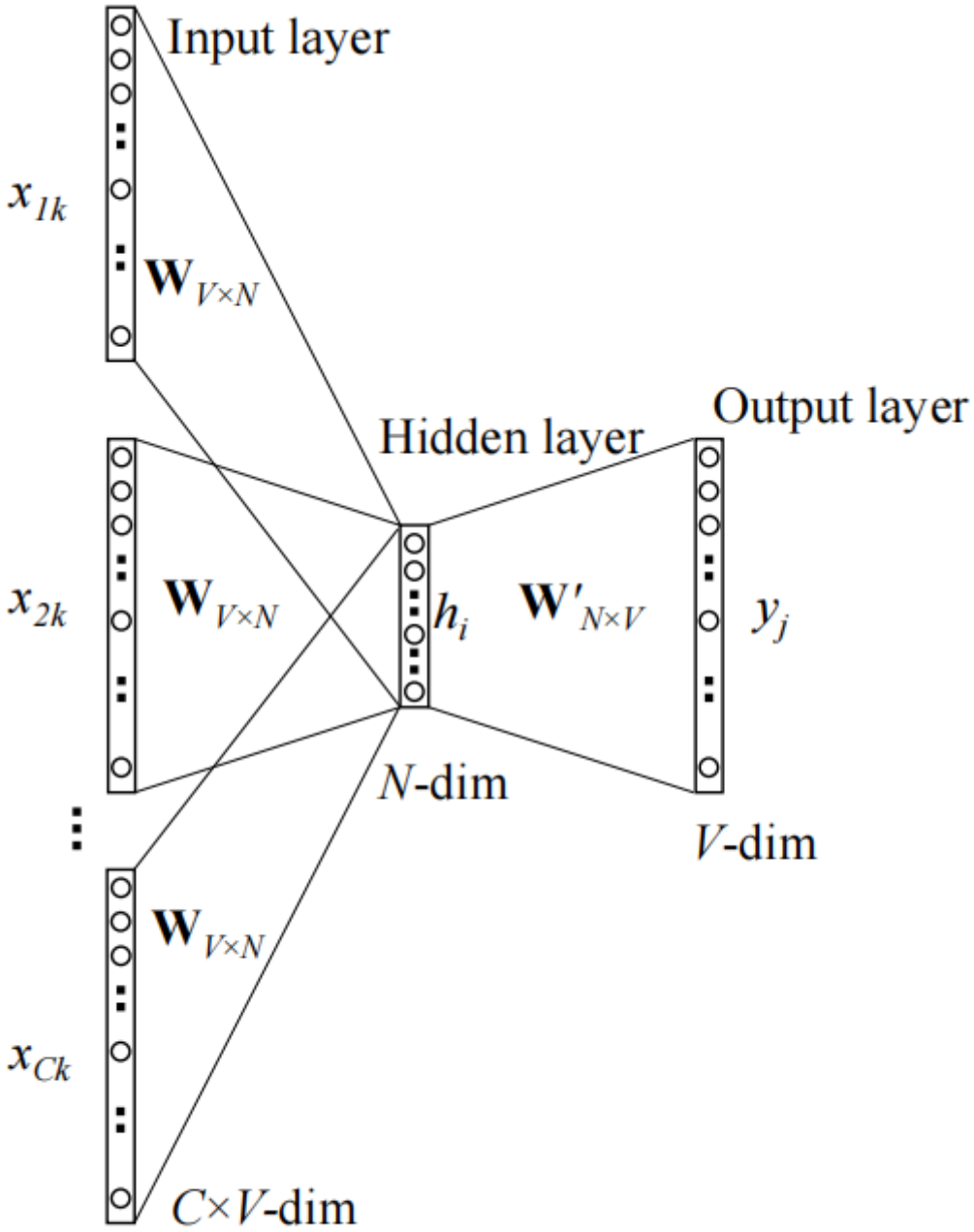

Figure 2: Continuous bag-of-word model

这里，输入端的context词由上面的一个变成了多个，平均一下就好了，输出端还是最大化中心词 j 对应的概率y_j.

1.2 skip-gram (SG) model

The skip-gram model is the opposite of the CBOW model. The target word is now at the input layer, and the context words are on the output layer.

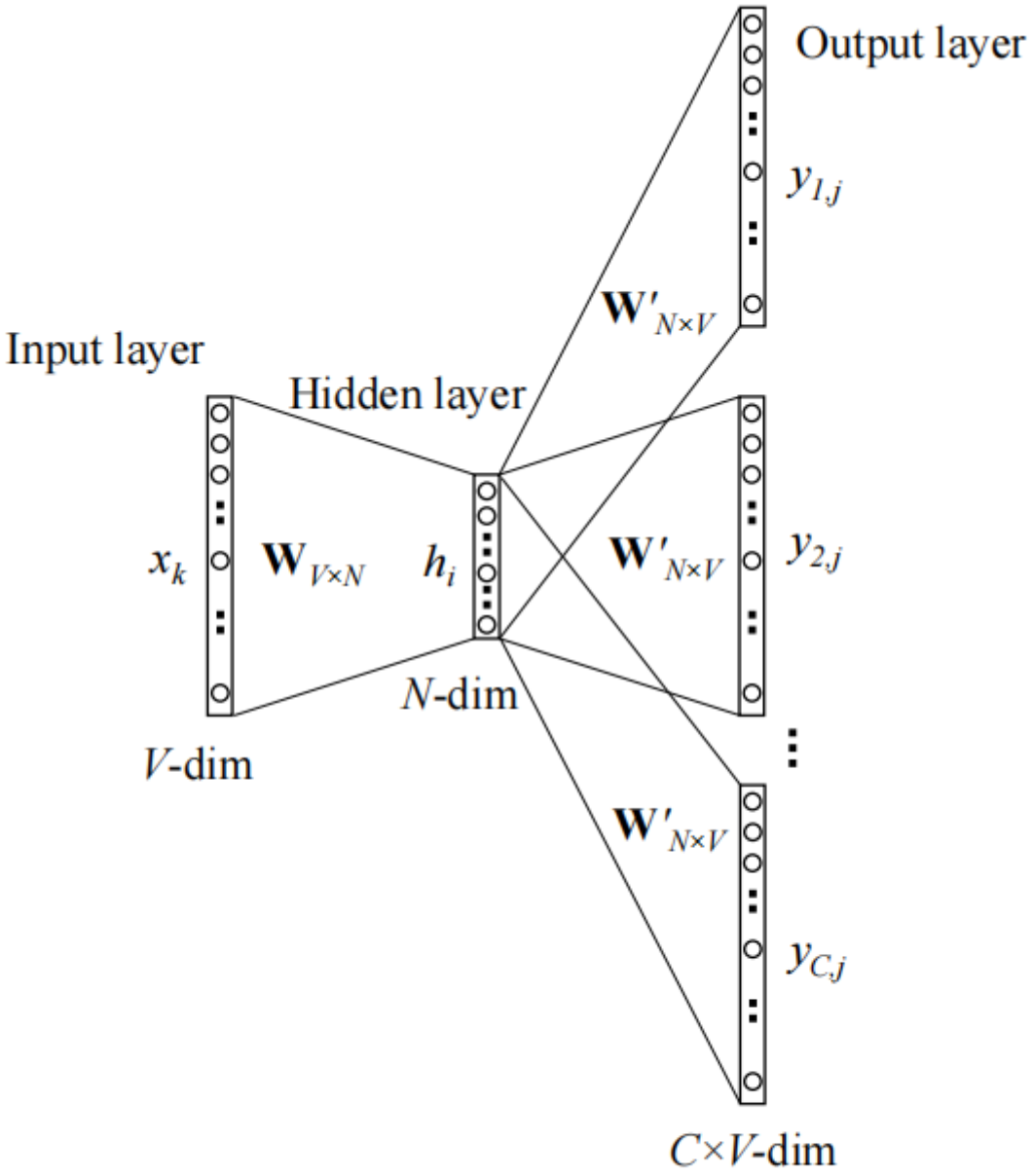

Figure 3: The skip-gram model.

这里，中心词 j 在输入端，context词在输出端，优化目标就是context词对应的概率：
$\max \sum_{i\in \text{context}_j} \log y_i$.

1.3 advanced optimization techniques

目标：在保证模型容量 (体现为output 向量的个数) 的情况下，减小计算量.

1.3.1 hierarchical softmax

层次化的softmax，还不如说是层次化的logistic。哈夫曼数的每个非叶子节点都对应一次logistic 回归，即sigmoid二分类，输入都是embedding，即隐变量。每个非叶子节点对应了一个参数vector，对应前面输出端的全连接层的参数，称为output vector。单词的数量为V，则哈夫曼树的非叶子节点数量为$V-1$，相比之前全连接的输出端，参数数量只少了一个vector，但计算量则从$O(V)$降为$O(\log V)$。

1.3.2 negative sampling

采集负样本后损失函数还是基于sigmoid而不是softmax，损失函数的形式是直接给出的，并没有什么理由，效果不错就用了。

负采样的采样分布称为 noise distribution，根据经验, word2vec uses a unigram distribution raised to the 3/4th power for the best quality of results.

注意，虽然我们基于一个分类任务得到了单词的embedding，但使用负采样却无法对单词进行预测，因为其输出端没有定义 well-defined prob distribution，这也是fastText用于分类任务时不能使用负采样的原因。

2 fastText

> 文本分类：
>
> - Bag of Tricks for Efficient Text Classification
>
> 词嵌入：
>
> - Enriching Word Vectors with Subword Information
> - FastText.zip: Compressing text classification models：模型压缩
>
> fastText（一）：从词嵌入到句嵌入 - 知乎 (zhihu.com)
>
> fastText（二）：从词嵌入到子词嵌入 - 知乎 (zhihu.com)

2.1 文本分类

基于 CBOW，只不过输出端不再是各个单词的概率，而是分类类别的概率，此外：

1. 多线程异步训练，学习率线性衰减；
2. 分类类别较多时，可以在输出端进行 Hierarchical softmax，另一个好处是预测阶段输出top类别的计算量也较小(DFS，深度优先搜索)；
3. 输入端，加入了n-gram特征，引入词序信息，助力分类；
4. n-gram的存储与检索基于hash映射；
5. 注意，词的embedding vector最开始是随机初始化的，最终在分类的过程中学习得到。

注意:

1. 虽然加入了n-gram特征，但是，n-gram的embedding与组成它的单词的embedding并不存在直接关系，都是独立学到的，没有平均关系，n-gram相当于是另外的一个单词，加入到字典中。
2. 类别不多的时候，直接softmax即可；当类别很多时，使用层次softmax；但是无法使用负采样，因为如前面所说，负采样在输出端没有定义完整的概率分布，而我们这里处理的是分类任务而不是只要学到embedding即可。另外，skip-gram model在这里也不适用。因此，**对分类任务，fastText只能是 CBOW + softmax/层次 softmax。**

2.2 词嵌入

文本分类的n-gram是**单词**级别的，但词嵌入的n-gram是**字符**级别的，也就是文章中提到的 **character** n-gram，也即文章标题提到的 Subword Information，通过考虑“子词”，在学习单词的表示的过程中考虑了单词的形态信息 (the morphology of words).

- 文章介绍时，只提到了基于 skip-gram model + 负采样，扩展到 CBOW 及 层次 softmax理论上来说也是可以的，但估计一般就是只用 skip-gram model + 负采样。
- 输入端 Each word *w* is represented as a bag of character *n*-gram，注意，这个n-gram bag还包含`<word w>`本身，其中`<,>`为起止的标识符；单词的最终表示应该是该单词 n-gram bag 中所有元素对应向量的 average（文章前面说sum，后面又说average，个人认为average合理些）；

- 输出端，以完整的单词作为分类类别，并没有n-gram；
- 最终，字典是所有单词的n-gram bag的并集。由此，我们还可以得到out-of-vocabulary words的embedding：生成该word的n-gram bag，在字典中查找其n-gram bag中元素的vector再平均即可，对于无法查找到的元素，忽略即可，比如对于 OOV word，`<OOV word>` 本身就一定不在已学好的字典中。
