深度学习必须熟悉的算法之word2vector（一）

milter 机器学习算法与自然语言处理 2018-05-08

作者:milter
链接:https://www.jianshu.com/p/1405932293ea

word2vector已经成为NLP领域的基石算法。作为一名AI 从业者，如果不能主动去熟悉该算法，应该感到脸红。本文是一篇翻译的文章，原文链接是：http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
如果你的英语很好，强烈建议直接阅读原文。这篇文章写的非常好，简明扼要，语言流畅。

这是我认为入门word2vector的最好文章，没有之一。当然，我也不是生硬的翻译，而是理解之后按照自己的逻辑再写出来，希望能更加清晰一些。

欢迎在评论中说出你的看法，多多交流。word2vector常见的有两种算法CBOW和skip gram，本文使用skip gram算法作为讲解对象。

1 算法的基本思想

word2vector，顾名思义，就是将语料库中的词转化成向量，以便后续在词向量的基础上进行各种计算。最常见的表示方法是counting 编码。假设我们的语料库中是如下三句话：
I like deep learning
I like NLP
I enjoy flying
利用counting编码，我们可以绘出如下矩阵：

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1

假设语料库中的单词数量是N，则上图矩阵的大小就是N*N，其中的每一行就代表一个词的向量表示。如第一行

0 2 1 0 0 0 0

是单词I的向量表示。其中的2代表I这个单词与like这个词在语料库中共同出现了2次。

似乎我们很简单就完成了“word2vector”是不是?

但是这种办法至少有三个缺陷：

- 1是词语数量较大时，向量维度高且稀疏，向量矩阵巨大而难以存储
- 2是向量并不包含单词的语义内容，只是基于数量统计。
- 3是当有新的词加入语料库后，整个向量矩阵需要更新

尽管我们可以通过SVD来降低向量的维度，但是SVD本身却是一个需要巨大计算量的操作。

很明显，这种办法在实际中并不好用。我们今天学习的skip gram算法可以成功克服以上三个缺陷。它的基本思想是首先将所有词语进行one-hot编码，输入只有一个隐藏层的神经网络，定义好loss后进行训练。

后面我们会讲解如何定义loss，这里暂时按下不表。训练完成后，我们就可以用隐藏层的权重来作为词的向量表示！！

这个思想乍听起来很神奇是不是？其实我们早就熟悉它了。auto-encoder时，我们也是用有一个隐藏层的神经网络进行训练，训练完成后，丢去后面的output层，只用隐藏层的输出作为最终需要的向量对象，藉此成功完成向量的压缩。

2 举例说明

1 构造训练数据

假设我们的语料库中只有一句话：

The quick brown fox jumps over the lazy dog.

这句话中共有8个词（这里The与the算同一个词）。

skip gram算法是怎么为这8个词生成词向量的呢?

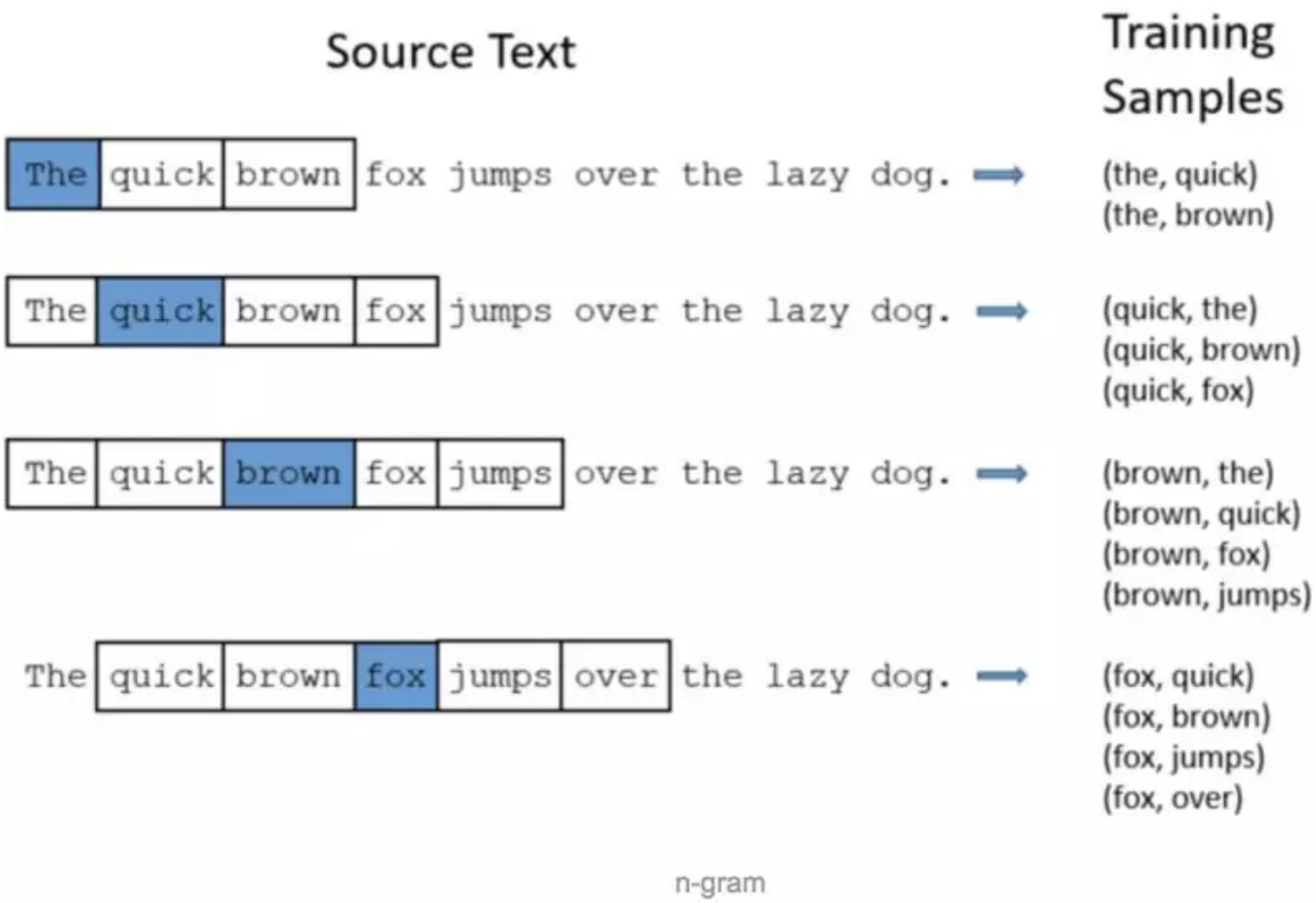
我们知道用神经网络训练，大体有如下几个步骤：

- 准备好data，即X和Y
- 定义好网络结构

- 定义好loss
- 选择合适的优化器
- 进行迭代训练
- 存储训练好的网络

所以，我们下面先来关注下如何确定X和Y的形式。
其实非常简单，（x,y）就是一个个的单词对。比如（the，quick）就是一个单词对，the就是样本数据，quick就是该条样本的标签。

那么，如何从上面那句话中生成单词对数据呢？答案就是n-gram方法。多说不如看图：



n-gram

我们以词为单位扫描这句话，每扫描到一个词，都把该词左右各两个词共4个词拿出来，分别与被扫描的单词组成单词对，作为我们的训练数据。

这里有两个细节，一个就是取被扫描单词左右各2个词，这里的2被称为**窗口尺寸**，是可以调整的，用多大的窗口生成的单词对来训练最好，需要具体问题具体分析。

一般来说，取5是很好的经验值。也就是左右各取5个单词，共10个单词。这里我们用2只是为了方便说明问题。
第二个细节就是句子头尾的单词被扫描时，其能取的单词对数要少几个，这个不影响大局，不用理会。

这里我们需要停下来细细琢磨下，我们这样取单词对作为训练数据的目的何在？以（fox，jumps）为例，jumps可以理解为fox的上下文，我们将fox输入神经网络时，希望网络能够告诉我们，在语料库的8个单

词中，jumps是更可能出现在fox周围的。

你可能会想，（fox，brown）也是一个单词对，它输入神经网络后，岂不是希望神经网络告诉我们，在8个单词中，brown是更可能出现在fox周围？如果是这样，那么训练完成后的神经网络，输入fox，它的输出会是brown和jumps的哪一个呢？

答案是取决于（fox，brown）和（fox，jumps）两个单词对谁在训练集中出现的次数比较多，神经网络就会针对哪个单词对按照梯度下降进行更多的调整，从而就会倾向于预测谁将出现在fox周围。

3 数字化表示单词对

上面我们获得了许多单词对作为训练数据，但是神经网络不能直接接收和输出字符串形式的单词对，所以需要将单词对转化为数字的形式。

方法也很简单，就是用one-hot编码，如下图所示：

	The	Quick	Brown	Fox	Jumps	Over	Lazy	Dog
The	1	0	0	0	0	0	0	0
Quick	0	1	0	0	0	0	0	0
Brown	0	0	1	0	0	0	0	0
Fox	0	0	0	1	0	0	0	0
Jumps	0	0	0	0	1	0	0	0
Over	0	0	0	0	0	1	0	0
Lazy	0	0	0	0	0	0	1	0
Dog	0	0	0	0	0	0	0	1

image.png

（the，quick）单词对就表示成【（1,0,0,0,0,0,0,0），（0,1,0,0,0,0,0,0）】

这样就可以输入神经网络进行训练了，当我们将the输入神经网络时，希望网络也能输出一个8维的向量，并且第二维尽可能接近1，其他维尽可能接近0。

也就是让神经网络告诉我们，quick更可能出现在the的周围。当然，我们还希望这8维向量所有位置的值相加为1，WHY?

因为相加为1就可以认为这个8维向量描述的是一个概率分布，正好我们的y值也是一个概率分布（一个位置为1，其他位置为0），我们就可以用交叉熵来衡量神经网络的输出与我们的label y的差异大小，也就可以定义出loss了。

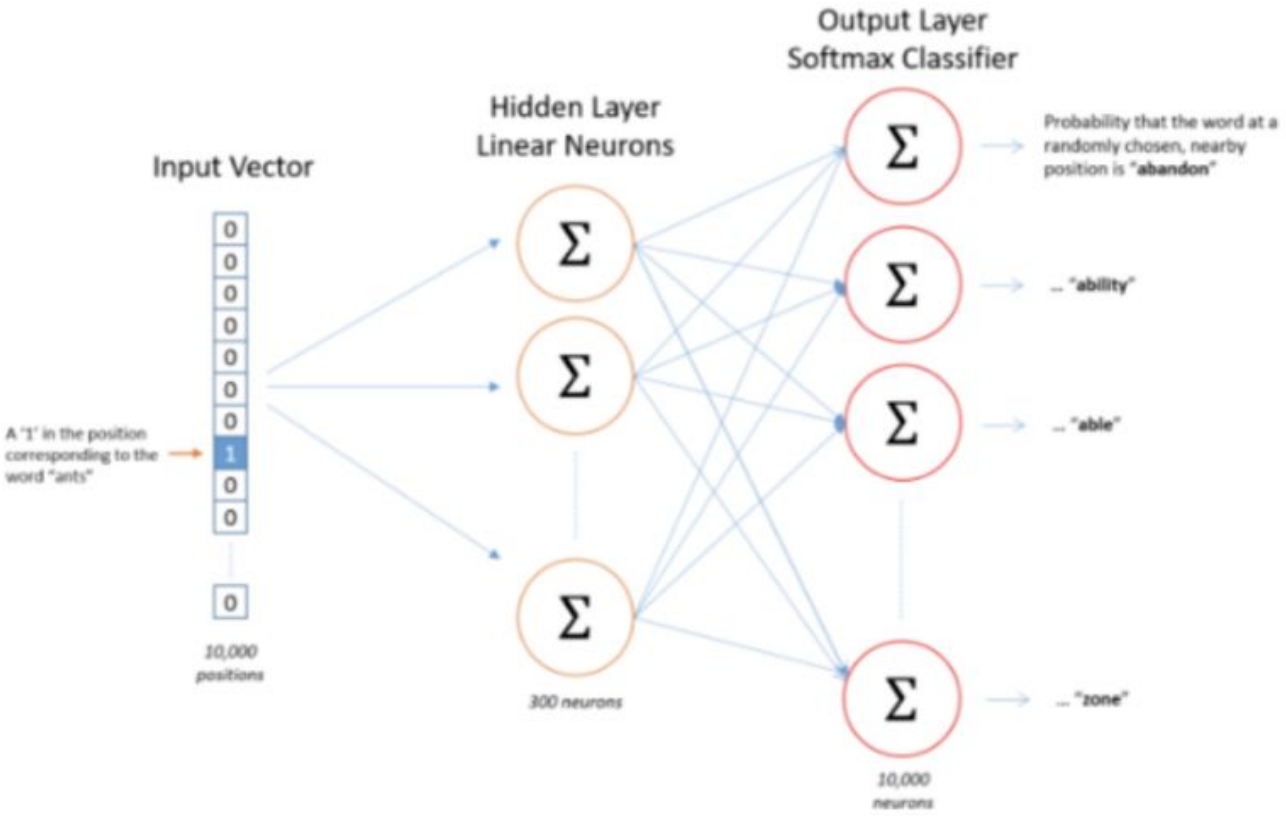
什么，你不知道啥是交叉熵？请参考我的另一篇文章【机器学习面试之各种混乱的熵】，应该不会让你失望。

4 定义网络结构

通过之前的叙述，我们已经基本知道神经网络应该是什么样子了，总结一下，可以确定如下情况：

- 神经网络的输入应该是8维的向量
- 神经网络只有一个隐藏层
- 神经网络的输出应该是一个8维向量，且各维的值相加为1

有了这些信息，我们可以很容易定义出如下的网络结构：



观察这个网络结构，我们可以发现，它的隐藏层并没有激活函数，但是输出层却用了softmax，这是为了保证输出的向量是一个概率分布。

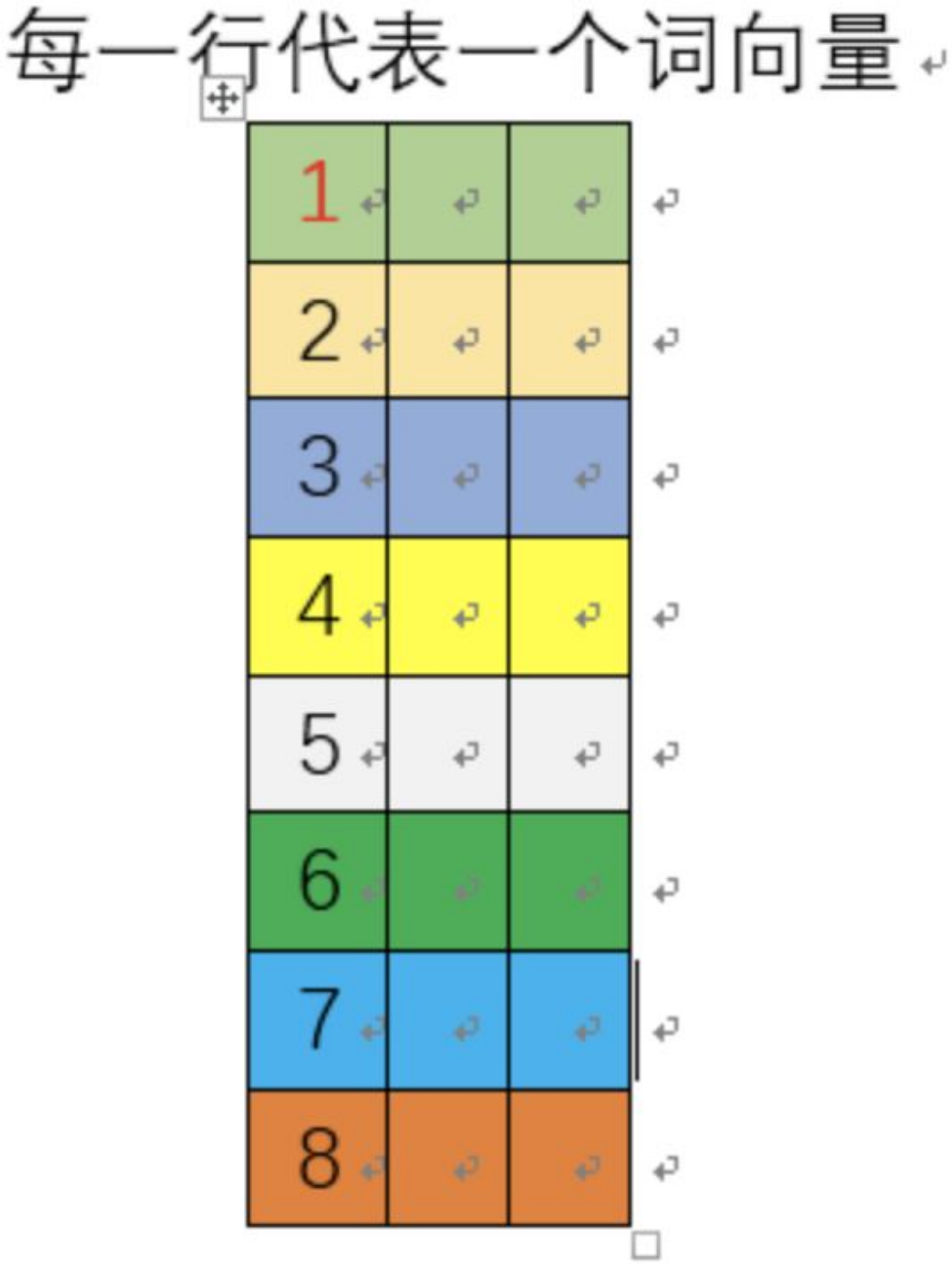
5 隐藏层

显然，输出层的神经元应该是8个，这样才能输出一个8维的向量。那么隐藏层的神经元应该是多少？

这取决于我们希望得到的词向量是多少维，有多少个隐藏神经元词向量就是多少维。每一个隐藏的神经元接收的输入都是一个8维向量，假设我们的隐藏神经元有3个（仅仅是为了举例说明使用，实际中，google推荐的是300个，但具体多少合适，需要你自己进行试验，怎么效果好怎么来），如此以来，隐藏层的权重就可以用一个8行3列的矩阵来表示。

下面就是见证奇迹的时刻！

网络训练完成后，这个8行3列的矩阵的每一行就是一个单词的词向量！如下图所示：



so，训练完成后，我们只需要保存好隐藏层的权重矩阵即可，输出层此时已经完成历史使命，可以丢掉了。

那么怎么使用去掉了输出层的网络呢？

我们知道，网络的输入是one-hot编码的单词，它与隐藏层权重矩阵相乘实际上是取权重矩阵特定的行，如下图所示：

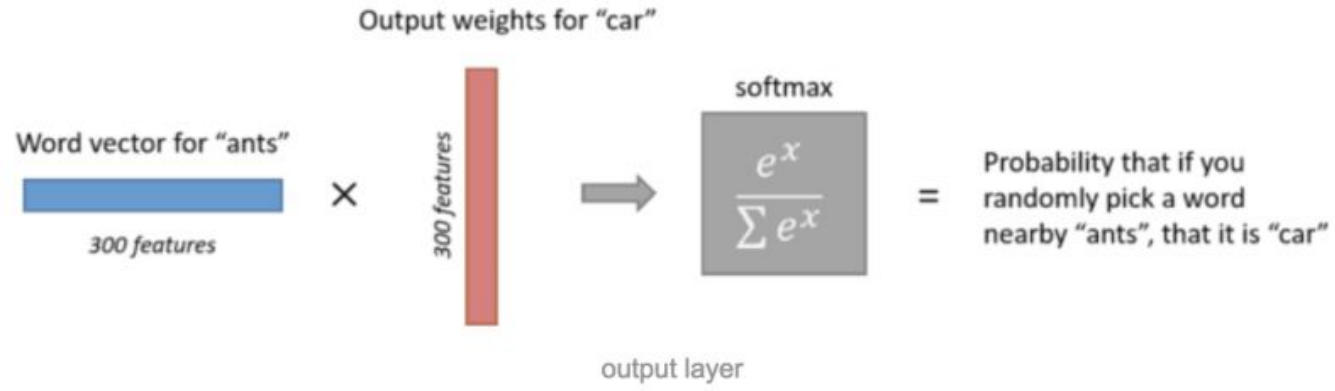
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

这意味着，隐藏层实际上相当于是一个查找表，它的输出就是输入的单词的词向量。

6 输出层

当我们从隐藏层获得一个单词的词向量后，就要经过输出层了。
输出层的神经元数量和语料库中的单词数量一样。

每一个神经元可以认为对应一个单词的**输出权重**，词向量乘以该**输出权重**就得到一个数，该数字代表了输出神经元对应的单词出现在输入单词周围的可能性大小，通过对所有的输出层神经元的输出进行softmax操作，我们就把输出层的输出规整为一个概率分布了。如下图所示：



这里有一点需要注意，我们说输出的是该单词出现在输入单词周围的概率大小，这个“周围”包含单词的前面，也包含单词的后面。

7 直觉的启示

前面，我们表示，skip gram算法生成的词向量可以包含语义信息，也就是说，语义相近的词其词向量也相近。这里，我们给一个直觉的解释。

首先，语义相近的词往往有着类似的上下文。这是什么意思呢？举例来说，“聪明”和“伶俐”两个词语义是相近的，那么它们的使用场景也是相似的，它们周围的词很大程度上是相近或相同的。

语义相近的词有着相似的上下文，让我们的神经网络在训练过程中对相近的词产生相近的输出向量。网络

如何做到这一点呢？答案就是训练完成后，网络能够对语义相近的词产生相近的词向量。因为此时的输出层已经训练完成，不会改变了。

这个直觉式的思考显然是不严谨的，但却能让我们对神经网络有很好的洞见。

记得李宏毅说过，有人问，LSTM设计那么复杂，设计的人怎么知道这样的结构就能达到记忆的效果呢？事实上，不是知道这样做会有记忆的效果才去这样做，而是这样做了，才有这样的效果。是不是有点鸡生蛋蛋生鸡的赶脚？这就是为什么深度学习被称为玄学的原因吧。

8 下篇预告

你可能注意到了，skip-gram算法构造的神经网络神经元太多了，导致权重矩阵非常大。 假设我们的语料库中的词有10000个，生成的词向量为300维。

那么权重系数就有10000*300*2那么多，训练如此巨大的网络难度很大。下一篇文章中会介绍一些小技巧，帮助我们减小训练的难度，使得训练是可行的。

END

推荐阅读：

当RNN神经网络遇上NER（命名实体识别）：双向LSTM，条件随机场（CRF），层叠Stack LSTM， 字母嵌入

【深度学习实战】pytorch中如何处理RNN输入变长序列padding

【机器学习基本理论】详解最大后验概率估计（MAP）的理解

欢迎关注公众号学习交流~

长按二维码扫描关注

机器学习算法与自然语言处理

ID: yizhennotes

通俗笔记，分享交流