CBOW最强理解

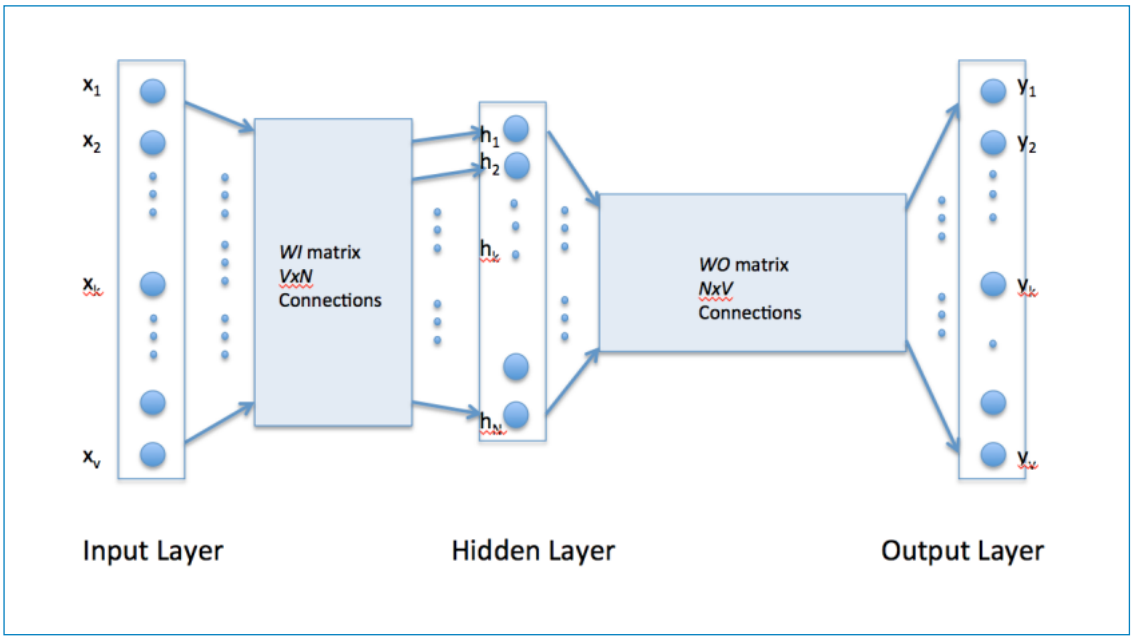
2018年07月25日 15:15:19 yangliuyi1994 阅读数：4698

翻译自：https://iksinc.online/tag/continuous-bag-of-words-cbow/

向量空间模型在信息检索中是众所周知的，其中每个文档被表示为向量。矢量分量表示文档中每个单词的权重或重要性。使用余弦相似性度量计算两个
似性。

尽管对单词使用矢量表示的想法也已经存在了一段时间，但是对于嵌入单词的技术，将单词映射到向量的技术，最近一直在飙升。其中一个驱动因素是
TomášMikolov的Word2vec算法，该算法使用大量文本来创建高维（50到300维）的单词表示，捕获单词之间的关系，无需外部注释。这种表述似乎
言规律。例如，它产生一个近似于*vec('Rome')*表示的向量，作为向量运算*vec('Paris') - vec('France') + vec('Italy')的结果*。

Word2vec使用单个隐藏层，完全连接的神经网络如下所示。隐藏层中的神经元都是线性神经元。输入层设置为具有与用于训练的词汇中的单词一样多
藏图层大小设置为生成的单词向量的维度。输出图层的大小与输入图层相同。因此，假设用于学习单词向量的词汇表由*V*个单词组成并且*N*为单词向量的
隐藏层连接的输入可以由大小为*VxN*的矩阵*WI*表示，其中每行表示词汇单词。以相同的方式，可以通过矩阵*WO*来描述从隐藏层到输出层的连接大小为
种情况下，*WO*矩阵的每列 表示来自给定词汇表的单词。使用“*1-out of -V*”表示对网络的输入进行编码，这意味着只有一条输入线被设置为1，其
置为零。



为了更好地处理Word2vec的工作原理，请考虑具有以下句子的训练语料库：

“狗看到了一只猫”，“狗追着猫”，“猫爬上了一棵树”

语料库词汇有八个单词。按字母顺序排序后，每个单词都可以通过其索引引用。对于这个例子，我们的神经网络将有八个输入神经元和八个输出神经元
我们决定在隐藏层中使用三个神经元。这意味着*WI*和*WO*将分别是*8×3*和*3×8*矩阵。在训练开始之前，这些矩阵被初始化为小的随机值，如通常在神经
样。仅为了说明，让我们假设*WI*和*WO*初始化为以下值：

WI =

```
-0.094491  -0.443977   0.313917
-0.490796  -0.229903   0.065460
 0.072921   0.172246  -0.357751
 0.104514  -0.463000   0.079367
-0.226080  -0.154659  -0.038422
 0.406115  -0.192794  -0.441992
 0.181755   0.088268   0.277574
-0.055334   0.491792   0.263102
```

W0 =

```
 0.023074   0.479901   0.432148   ^ 275480  -0.364732  -0.119840   0.266070  -0.351000
-0.368008   0.424778  -0.257104     48817   0.033922   0.353874  -0.144942   0.130904
 0.422434   0.364503   0.467865     20302  -0.423890  -0.438777   0.268529  -0.446787
```

假设我们希望网络学习单词“cat”和 ”之间的关系。也就是说，当“猫”输入到网络时，网络应该显示“攀爬”的高概率。在单词嵌入术语中
词“cat”被称为*上下文*单词，单词“climbed”被称为*目标*单词。在这种情况下，输入矢量*X*将是[0 1 0 0 0 0 0 0] t。请注意，只有向量的第二个分量
输入的单词是“cat”，它在语料库单词的排序列表中保持第二个位置。鉴于目标字是“爬升”，目标矢量看起来像[0 0 0 1 0 0 0 0] t。

利用表示“cat”的输入向量，可以将隐藏层神经元的输出计算为

H t = X t WI = [-0.490796 -0.229903 0.065460]

我们不应该感到惊讶的是隐藏神经元输出的向量*H*模仿了*WI*矩阵的第二行的权重，因为*V*表示为*1-out-of-V。因此隐藏层连接的输入功能基本上是将输到隐藏层*。对隐藏到输出层执行类似的操作，输出层神经元的激活矢量可以写成

H t WO = [0.100934 -0.309331 -0.122361 -0.151399 0.143463 -0.051262 -0.079686 0.112928]

由于目标是为输出层中的单词生成概率， 对于*k = 1，V的Pr（单词k |单词上下文）*，为了反映它们与输入处的上下文单词的下一个单词关系，我们需的总和。输出图层添加到一个。Word2vec通过使用softmax函数将输出层神经元的激活值转换为概率来实现此目的。因此，第*k个*神经元的输出通过以算，其中*activation（n）*表示第*n个*输出层神经元的激活值：

$$y_k = \Pr(word_k \mid word_{context}) = \frac{\exp(activation(k))}{\sum_{n=1}^{V} \exp(activation(n))}$$

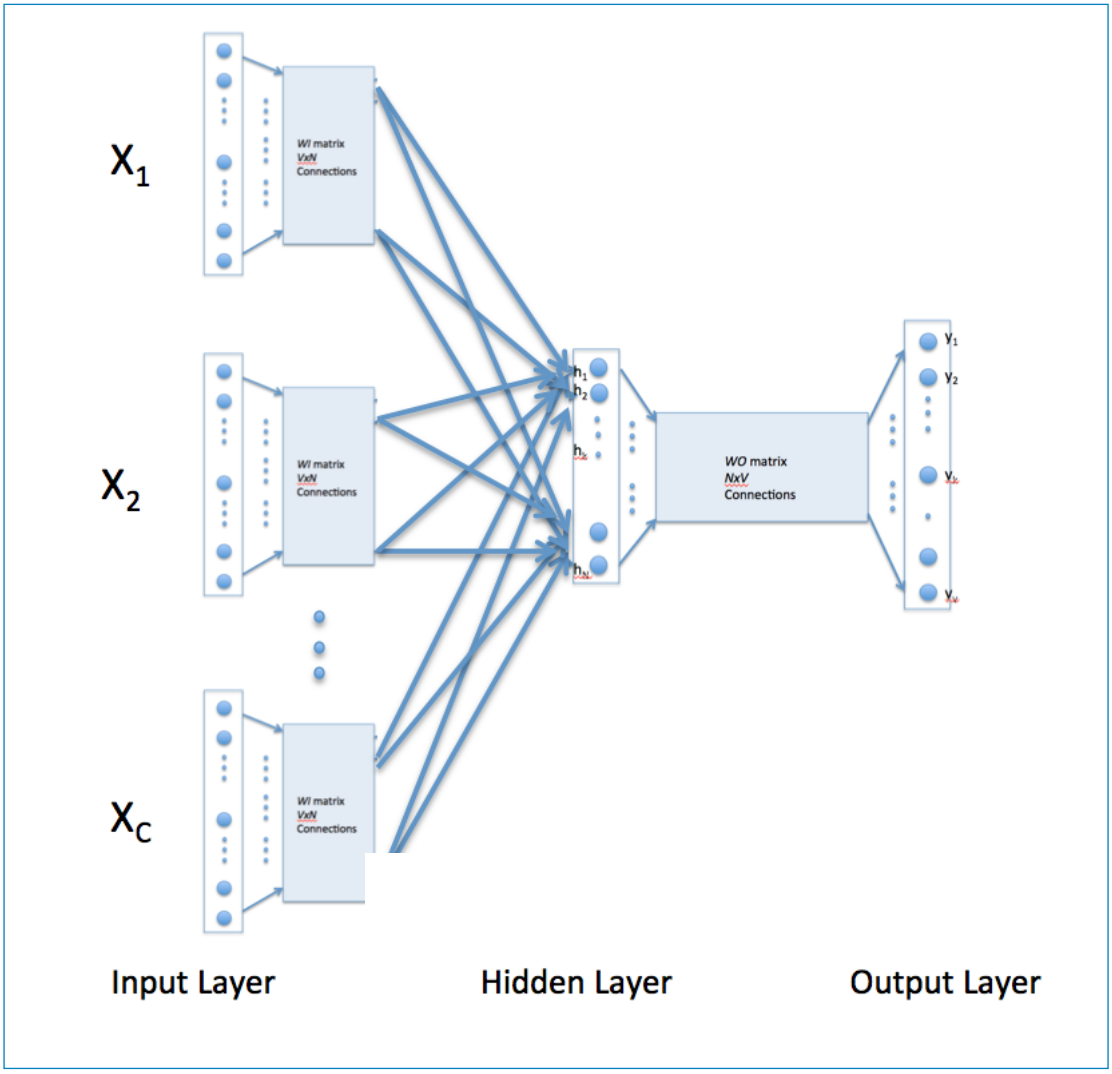
因此，语料库中八个单词的概率是：

0.143073 0.094925 0.114441 **0.111166** 0.149289 0.122874 0.119431 0.144800

粗体的概率是针对所选择的目标词“攀爬”。给定目标矢量[0 0 0 1 0 0 0 0] t，通过从目标矢量中减去概率矢量，可以容易地计算输出层的误差矢量。错误，就可以使用反向传播来更新矩阵*WO*和*WI中*
的权重。因此，训练可以通过从语料库呈现不同的上下文目标词对来进行。从本质上讲，这就是Word2vec如何学习单词之间的关系，并在此过程中开词的向量表示。

连续词汇（CBOW）学习

以上描述和架构旨在用于学习单词对之间的关系。在连续的单词模型中，上下文由给定目标单词的多个单词表示。例如，我们可以使用“cat”和“tre爬”的上下文单词作为目标单词。这需要修改神经网络架构。如下所示，修改包括将隐藏层连接的输入复制*C次*，上下文单词的数量，以及在隐藏层神*C*操作。*[警报读者指出，下图可能会让一些读者认为CBOW学习使用了几个输入矩阵。不是这样。它是相同的矩阵WI，它接收代表不同上下文词的多*



利用上述配置来指定*C*上下文字，使用1-out-of-V表示编码的每个字意味着隐藏层输出是与输入处的上下文字相对应的字矢量的平均值。输出层保持不面讨论的方式完成训练。

Skip-Gram模型

Skip-gram模型反转了目标和上下文单词的使用。在这种情况下，目标字在输入处被馈送，隐藏层保持相同，并且神经网络的输出层被多次复制以适应下文字。以“cat”和“tree”为例，作为上下文单词，“爬”作为目标词，skim-gram模型中的输入向量为[0 0 0 1 0 0 0 0] t，而两个输出层将具有0] t和[0 0 0 0 0 0 0 1] t分别作为目标向量。代替产生一个概率向量，将为当前示例产生两个这样的向量。以上面讨论的方式产生每个输出层的误差向来自所有输出层的误差向量相加以通过反向传播来调整权重。这确保了每个输出层的权重矩阵*WO*在整个训练中保持相同。