

# 看不懂你打我系列之word2vec详解（一）

原创 AI小老弟 AI小老弟 2019-12-26

## 看不懂你打我系列之word2vec详解（一）

### 看不懂你打我系列

看不懂你打我系列，是小老弟在学习某个知识点或概念过程中的总结，希望小老弟写的足够明白~

### 导读

word2vec将分为两篇进行推送，第一篇对其基本原理、两种训练任务和推导进行介绍，第二篇对word2vec训练过程中的加速算法进行介绍。

word2vec，如其名字"word to vector"，词语向量化，虽然新出的Bert等深度学习模型横扫各大文本任务，但word2vec仍有其独特的魅力和价值，活跃在众多项目中，了解word2vec，也是探索语言模型的一个很好的引子。

万事如意 : [-0.14924122, 0.45343554, -0.86415035, ....., -0.17407797, 0.5148031, -0.36855575]

阖家欢乐 : [-0.17345215, 0.19527681, -0.84978914, ....., -0.08482756, 0.4930622, -0.34809354]

不好意思 : [-0.07807594, 0.40587068, 0.5906859, ....., -0.18342374, -0.368896, -0.26421118]

Similarity(万事如意,阖家欢乐) = 0.94212776

Similarity(万事如意,不好意思) = 0.34795684

上图为三个词的词向量（原始为100维）以及他们之间的相似度计算，可以看到，相似性上还是非常符合逻辑的。那么这个向量到底是怎么来的呢？

这就要从word2vec的模型说起，word2vec是一个三层的神经网络，包括输入层（Input），隐藏层（Hidden）和输出层（Output）；同时有两种模型，完成两个不同的任务；词向量化，就是这两个任务的中间产物，后面会讲为什么叫中间产物。

第一种模型CBOW(Continuous Bag-of-Words Model)，其任务简单来说就是利用一个词的前后文去预测这个中心词出现的概率，而第二个模型skip-gram刚好反过来，是利用中心词去预测前后的词出现的概率，到这来可能还有些云里雾里，不要着急，继续往下看。

在介绍两个模型前，对一些符号和定义进行说明：

**词典**：对要进行word2vec训练的文本进行清洗、分词、去重后形成的一个词典。

**$x$** ：表示输入的词的向量，其长度为词典的长度，假定词典为

【“我”，“爱”，“学习”，“没有”，“什么”，“能够”，“阻挡”，“的”，“脚步”】，输入词为“学习”的时候，对应的向量就是[0,0,1,0,0,0,0,0,0]，即词典的one-hot。

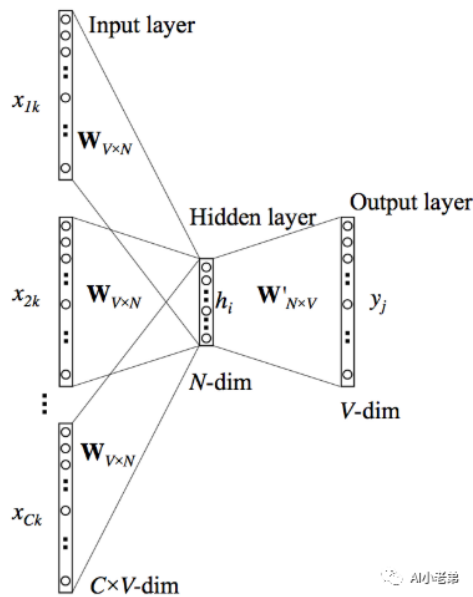
**$W$** ：一个 $V \times N$ 的矩阵，其中 $V$ 大小为词典长度， $N$ 为模型训练前的自定义参数。

**$W'$** ：一个 $N \times V$ 的矩阵， $N$ 和 $V$ 的值与上面对应。

**$h_i$** ：隐藏层的第 $i$ 个节点值。

**$y_i$** ：输出层第 $i$ 个节点值。

首先来看cbow模型，其结构如下图。

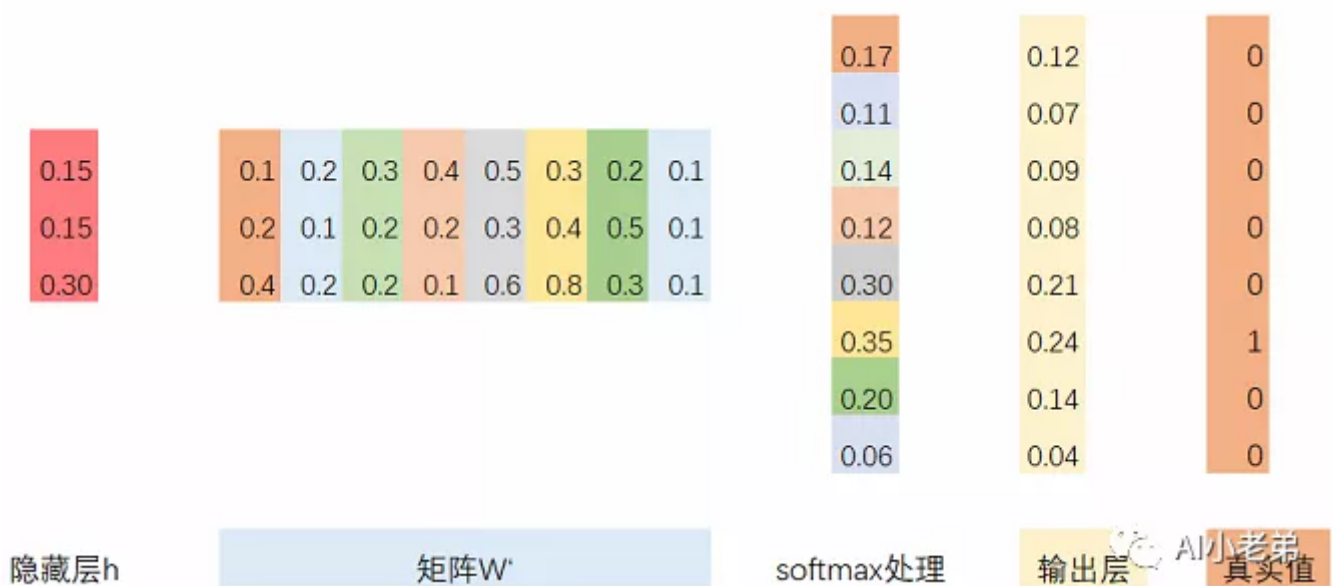
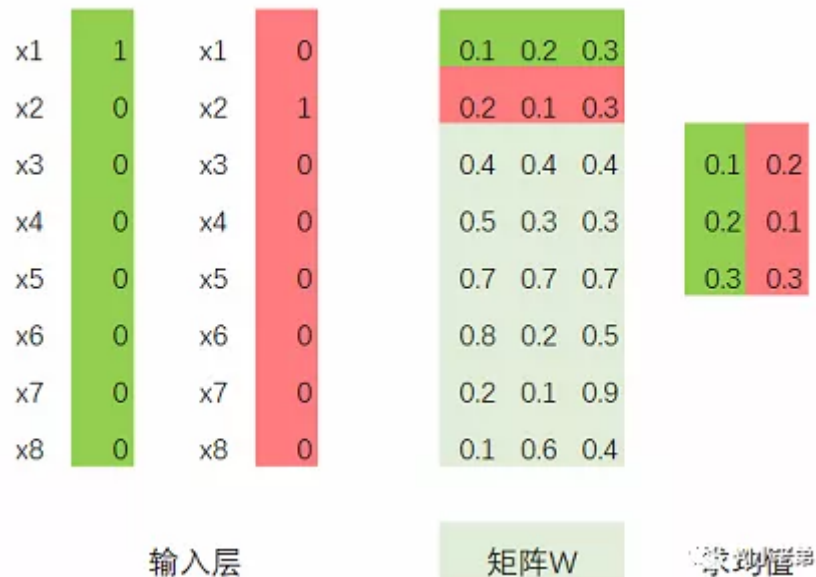


前面说过，CBOW任务是利用上下文来预测中心词，既然是预测，我们就需要构造输入变量和目标变量，如下图，假定窗口大小为3，依次滑动，即产生相应变量。

DOCUMENT										INPUT WORD	TARGET WORD	
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	我	学习	爱
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	爱	没有	学习
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	学习	什么	没有
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	没有	能够	什么
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	什么	阻挡	能够
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	能够	我	阻挡

接下来，以第一对数据为例，将one-hot后的“我”和“学习”两个向量，分别乘以矩阵 $W$ ，得到两个 $N$ 维向量，相加取均值后，即为隐藏层 $h$ 的 $N$ 个节点，然后隐藏层再与矩阵 $W'$ 相乘，得到一个 $V$ 维向量，最后通过softmax激活函数，将 $V$ 维向量中的每个值转化为一个概率值，实际上，这个 $V$ 维向量的每一个元素，对应的就是预测词为词典中的每个词的概率大小，我们希望输出层的每个值与对应的目标词向量值接近，到这里，就完成了数据的正向流动。

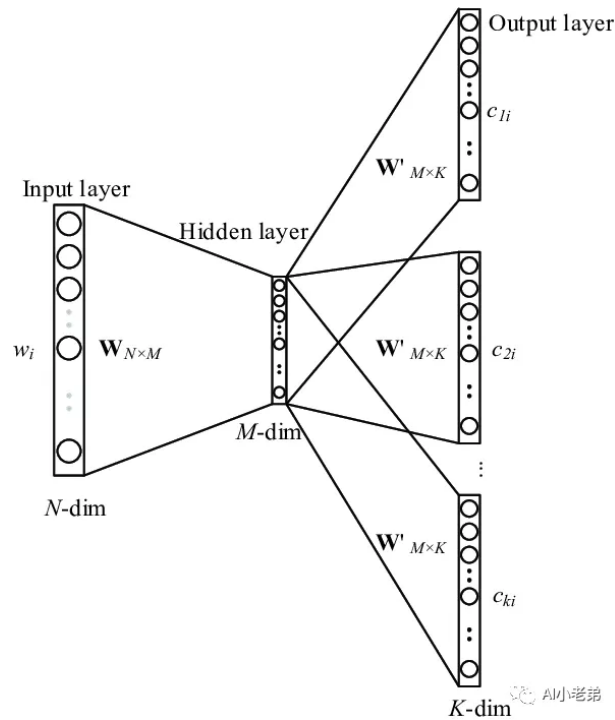
（下图假定输入两个词，同时词典大小 $V=8$ ， $W$ 矩阵 $N=3$ ）



注意： $W$ 和 $W'$ 矩阵最开始是随机生成的，通过反向传导机制（下文将展开推导），利用输出值与实际值之间的误差，对 $W$ 和 $W'$ 中的元素值进行更新。

看到这里，大家应该对前面说的“中间产物”有想法了，没错，所谓的中间产物，就是这个 $W$ 矩阵，通过多轮训练后得到的 $W$ 矩阵，就可以将V维（几万甚至更大）的稀疏向量，转变为N维（一般一百维）的稠密向量。也就完成了词向量化这一过程。

明白了CBOW模型，skip-gram模型就简单了，结构如图。



同样，我们需要构造输入变量和目标变量

DOCUMENT										INPUT WORD	TARGET WORD
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	爱	我
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	学习	爱
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	没有	学习
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	什么	能够
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	能够	阻挡
我	爱	学习	没有	什么	能够	阻挡	我	的	脚步	阻挡	能够

这次，我们需要将隐藏层向量乘以2次，得到两个V维向量，然后再经过softmax激活后，分别与相应的目标词向量进行比较。

以上就是word2vec两种模型的大体思路框架，下面的部分是反向传导过程的推导，涉及许多公式，不感兴趣的童鞋后面可以无视了。

公式较多，因此以图片形式插入，如对阅读造成影响，请谅解。



首先我们假定输入输出都只有一个词，从上面数据流动图可以看到， $W$  矩阵的每一行都是一个  $N$  维的向量，表示对应输入层的词， $W$  的第  $i$  行用  $v_w^T$  表示，假设输入向量中  $x_k = 1, x_{k'} = 0 (k' \neq k)$ ，则 (1) 式本质上是将  $W$  的第  $k$  行复制给  $h$ 。因此， $v_{w_I}$  是输入单词  $w_I$  的向量表示。

$$h = W^T x = W_{k, :}^T := v_{w_I}^T \quad (1)$$

接下来是  $N \times V$  的  $W'$  矩阵，隐藏层向量与  $W'$  矩阵相乘，可以计算出词典中每一个词的“得分”  $u_j$ ：

$$u_j = v_{w_I}^T h \quad (2)$$

$v_{w_j}$  表示矩阵  $W'$  的第  $j$  列。

最后的 softmax 层把输出的  $u_j$  转变为一个元素值和为 1 的向量，每一个值代表着其对应的词典中这个词的概率，如式(3)， $j$  表示输出层输出的第  $j$  个值，也是词典中的第  $j$  个值。

AI小老弟

$$p(w_j | w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^V \exp(u_{j'})} \quad (3)$$

带入  $u_j$ ，得：

$$p(w_j | w_I) = y_j = \frac{\exp(v_{w_j}^T v_{w_I})}{\sum_{j'=1}^V \exp(v_{w_{j'}}^T v_{w_I})} \quad (4)$$

这里需要注意下， $v_w$  来自输入层和隐藏层间的权重矩阵  $W$  的行， $v_{w_j}$  来自隐藏层和输出层间的权重矩阵  $W'$  的列。可以认为， $v_w$  为词语  $w$  的输入向量，称  $v_{w_j}$  为词语  $w$  的输出向量。输入向量和输出向量并不是代表同一个词，本例中进行了简化，输入为一个词，实际上，这里输入的是多个词，所以这里的输入向量也就代表着上下文，而输出向量，则代表着对中心词（目标词）的预测。

按照反向传导的机制，我们首先求隐藏层到输出层的矩阵的更新方式。

由式(3)，我们的目标是最大化  $p(w_o | w_I)$   $w_o$  表示输出词，假设目标词为  $j^*$ ，则要最大化输出层  $j^*$  节点对应的概率，即：

AI小老弟

$$\begin{aligned}
 \max p(w_0 | w_1) &= \max y_{j^*} (5) \\
 &= \max \log y_{j^*} (6) \\
 &= u_{j^*} - \log \sum_{j'=1}^V \exp(u_{j'}) := -E (7)
 \end{aligned}$$

上式中，最大化 (6) 即相当于最小化它的相反数，也就是我们的损失函数。

我们要对 W 矩阵的值进行更新，所以对 E 求关于  $w_{ij}$  的偏导，以得到隐藏层到输出层权重的梯度：

AI小师弟

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i \quad (8)$$

上式为链式求导法则求出，具体：

$$\begin{aligned}
 \frac{\partial E}{\partial u_j} &= \frac{\partial (u_{j^*} - \log \left( \sum_{j'=1}^V \exp(u_{j'}) \right))}{\partial u_j} \\
 &= \frac{\partial u_{j^*}}{\partial u_j} - \frac{\partial \log \left( \sum_{j'=1}^V \exp(u_{j'}) \right)}{\partial u_j}
 \end{aligned} \quad (9)$$

$$\frac{\partial u_{j^*}}{\partial u_j} = t_j \quad t_j = 1 \text{ when } j = j^* \text{ else } t_j = 0$$

AI小师弟

$$\frac{\partial \log \left( \sum_{j=1}^V \exp(u_j) \right)}{\partial u_j} = \frac{\exp(u_j)}{\sum_{j=1}^V \exp(u_j)} = y_j \quad (11)$$

由式(2)

$$\frac{\partial u_j}{\partial w'_{ij}} = h_i \quad (12)$$

AI小老师

所以

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta \cdot e_j \cdot h_i \quad (13)$$

向量表示：

$$V_{w_j}^{(new)} = V_{w_j}^{(old)} - \eta \cdot e_j \cdot h, j = 1, 2, \dots, V$$

$\eta > 0$  为学习率。

再来看，输出层到隐藏层的更新，同样是链式求导法则

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w'_{ki}} = EH_i \cdot x_k$$

141小弟弟



$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^V \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^V e_j \cdot w'_{ij} := EH_i \quad (15)$$

这里的求和，是因为  $h_i$  与输出层每个  $j$  都有联系，具体请参考反向传播相关理论， $EH$  定义为一个  $N$  维向量。

$$h_i = \sum_{k=1}^V x_k \cdot w_{ki} = x_k \cdot w_{ki}$$

$$\frac{\partial h_i}{\partial w_{ki}} = x_k \quad (16)$$

最后，将 (14) 写为矩阵形式：

$$\frac{\partial E}{\partial W} = \mathbf{x} \otimes EH = \mathbf{x}EH^T \quad (17)$$

则 (17) 中得到一个  $V \times N$  的矩阵，由于  $\mathbf{x}$  只有一个元素不为 0，所以相乘后也即  $\mathbf{E}$  对  $\mathbf{W}$  的偏导只有一行不全为 0，而这一行也就是  $EH^T$ ，最终，我们得到  $\mathbf{W}$  矩阵的更新公式：

$$\mathbf{v}_{w_I}^{(new)} = \mathbf{v}_{w_I}^{(old)} - \eta EH^T \quad (18)$$

注意，由前文， $\mathbf{v}_{w_I}$  是  $\mathbf{W}$  的一行，代表输入向量，因此在每次更新中，只有这一行的值会变化，其他行都保持不变，因为梯度为零。

以上就是两个矩阵反向传导的全部过程。


 AI小塔第

上面为假定输入输出都只有一个词的情况，对于 CBOW 来说，计算隐藏层输出的时，需要输入多个词，因此取所有上下文单词输入向量乘以  $W$  矩阵再求平均值。

$$\begin{aligned} h &= \frac{1}{C} W^T (x_1 + x_2 + \dots + x_C) \\ &= \frac{1}{C} (v_{w_1} + v_{w_2} + \dots + v_{w_C}) \end{aligned} \quad (19)$$

同时，损失函数基本一致，只是由于  $h$  发生了变化，所以有

$$\begin{aligned} E &= -\log p(w_o | w_{I,1}, w_{I,2}, \dots, w_{I,C}) \\ &= -u_{j^*} + \log \left( \sum_{j=1}^V \exp(u_j') \right) \\ &= -v_{w_o}'^T \cdot h + \log \sum_{j=1}^V \exp(v_{w_j}'^T \cdot h) \end{aligned} \quad (20)$$

 AI小老弟


相应的， $W'$  矩阵的更新公式没有发生变化，依然为：

$$V_{w_j}'^{(new)} = V_{w_j}'^{(old)} - \eta \cdot e_j \cdot h, j = 1, 2, \dots, V$$

同时，输入层到隐藏层间的权重更新等式跟式 (16) 类似，只是现在需要将下面这个等式应用到每一个输入的上下文单词  $w_{I,c}$ ，则：

$$V_{w_{I,c}}^{(new)} = V_{w_{I,c}}^{(old)} - \frac{1}{C} \eta E H^T \quad c=1, 2, \dots, C$$

式中  $v_{w_{I,c}}$  是输入上下文的第  $c$  个单词的输入向量。

 AI小老弟

对于 skip-gram 来说，就要稍微复杂点，因为输出的不再是一个词，所以损失函数发生变化：

$$E = -\log p(w_{o,1}, w_{o,2}, \dots, w_{o,c} | w_i)$$

$$= -\log \prod_{c=1}^c \frac{\exp(u_{c,j_c})}{\sum_{j'=1}^V \exp(u_{j'})}$$

$$= -\sum_{c=1}^c u_{j_c} + C \log \left( \sum_{j'=1}^V \exp(u_{j'}) \right)$$

AI小老弟

对  $E$  求输出的每一个词  $u_{c,j}$  求偏导：

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j}$$

定义一个  $V$  维向量  $EI = \{EI_1, EI_2, \dots, EI_V\}$  作为所有上下文单词的预测误差之和，则：

$$EI_j = \sum_{c=1}^c e_{c,j}$$

同样利用链式求导法则：

$$\frac{\partial E}{\partial w'_{ij}} = \sum_{c=1}^c \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w'_{ij}} = EI_j \cdot h_i$$

得：

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta \cdot EI_j \cdot h_i$$

向量形式：

$$\mathbf{v}_{w_j}^{(new)} = \mathbf{v}_{w_j}^{(old)} - \eta \cdot EI_j \cdot \mathbf{h} \quad j = 1, 2, \dots, V$$

AI小老弟

输出-隐藏层的更新等式推导与(14)-(18)一致，此时  $e_i$  为  $EI_i$ ，故：

$$\mathbf{v}_{w_I}^{(new)} = \mathbf{v}_{w_I}^{(old)} - \eta EH^T$$

其中， $EH$  是一个  $N$  维向量，每个元素定义为：

$$EH_i = \sum_{j=1}^V EI_j \cdot w'_{i,j}$$

AI小老师

至此，CBOW和skip-gram的反向传递推导完成。当然，实际应用远不止这么简单，对于输入层到隐层的矩阵  $W$ ，从更新公式来看我们每次训练只需要更新一行向量即可，但对于隐层到输出层的矩阵  $W'$ ，每次训练需要更新全部  $N \times V$  个元素，同时实际应用中词典往往是几万甚至上百万，这些都对word2vec性能提出了很大的挑战，因此word2vec使用了两种加速方法，即负采样（Negative Sample）和层次 softmax（Hierarchical Softmax），极大的加快了word2vec在大语料上的训练速度，这也是word2vec能够广受欢迎的原因之一，对于这两种加速方法的介绍，将在第二部分中展开。

感谢您的阅读

感觉还行？请点下“在看”，谢谢！

