前言

word2vec虽然非常流行和被广泛关注，但即使在原作者(Mikolov et al)的文章中，也没有给出CBOW和Skip-Gram两个模型的具体推导。同时涉及到的优化算法：hierarchical softmax 和 negative sampling也没有相应的数学推导。

Xin Rong在《word2vec Parameter Learning Explained》给出了了CBOW和Skip-Gram模型以及优化技巧：Hierarchical Softmax和Negative Sampling的详细公式推导，并加以理解。文章由浅入深、循序渐进，是深入理解word2vec的好文章。很痛心的是，Xin Rong于2019年因飞机失事而去世。

1. CBOW模型

1.1 一个上下文词情况

作者先从CBOW模型的简化版本(即只有一个上下文词，预测目标词)出发，对模型及参数更新进行了推导。

模型图如下：

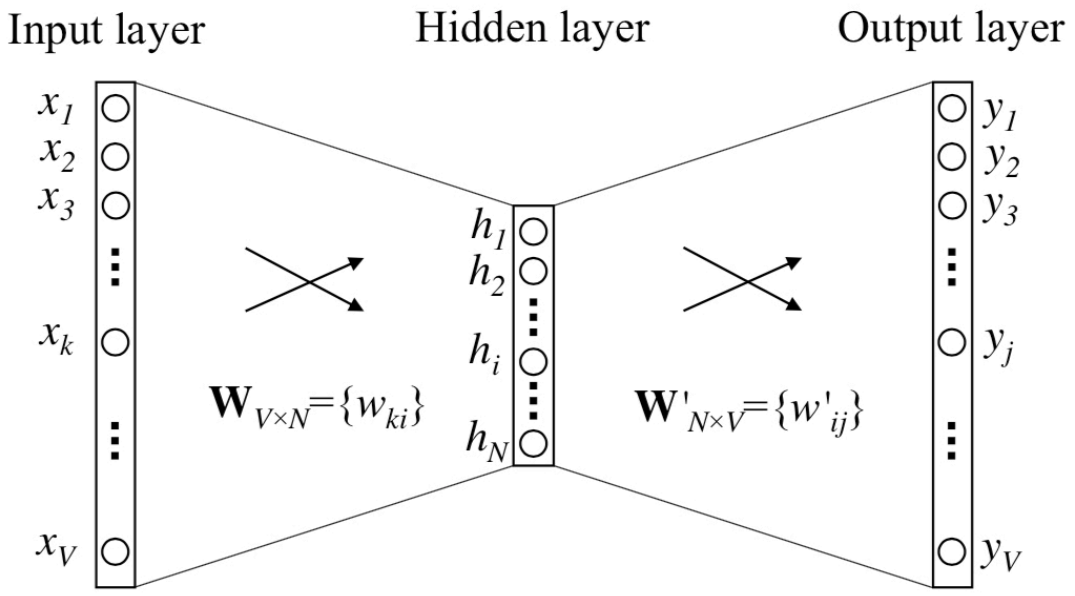


Figure 1: A simple CBOW model with only one word in the context

符号说明：

1. V代表词表大小；
2. N代表隐藏层的大小；
3. $\mathbf{W}$和$\mathbf{W}'$是两个完全不同的矩阵，不是转置矩阵！！
4. 输入向量$\mathbf{x}$是一个词的ont-hot向量，shape = $V \times 1$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_V \end{bmatrix}$$

5. Input layer到Hidden layer的权重矩阵是一个$V \times N$的矩阵，记为$\mathbf{W}$；

   它的行向量其实就是我们要求的词向量（见下文推导，可以理解为一种降维计算，将原来稀疏的one-hot向量降维为稠密的N为向量）

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ w_{V1} & w_{V2} & \cdots & w_{VN} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{w_1} \\ \mathbf{v}_{w_2} \\ \vdots \\ \mathbf{v}_{w_V} \end{bmatrix}$$

$$\mathbf{W}^T = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{V1} \\ w_{12} & w_{22} & \cdots & w_{V2} \\ \vdots & \vdots & \vdots & \vdots \\ w_{1N} & w_{2N} & \cdots & w_{VN} \end{bmatrix} = [\, \mathbf{v}_{w_1}^T \quad \mathbf{v}_{w_2}^T \quad \cdots \quad \mathbf{v}_{w_V}^T \,]$$

6. Hidden layer到Output layer的权重矩阵为$\mathbf{W}'$， shape = $N \times V$

   $\mathbf{W}'$的列向量$\mathbf{v}'_{wj}$其实也是词向量的一种表达（见下文推导）

$$\mathbf{W}' = \begin{bmatrix} w'_{11} & w'_{12} & \cdots & w'_{1V} \\ w'_{21} & w'_{22} & \cdots & w'_{2V} \\ \vdots & \vdots & \vdots & \vdots \\ w'_{N1} & w'_{N2} & \cdots & w'_{NV} \end{bmatrix} = \begin{bmatrix} \mathbf{v}'_{w_1} & \mathbf{v}'_{w_2} & \cdots & \mathbf{v}'_{w_V} \end{bmatrix}$$

1.1.1 前向传播

1. Input layer ——> Hidden layer

给定一个上下文词的one-hot向量，假设其第k个元素$x_k = 1$，其它元素$x_{k'} = 0\ (k' \neq k)$，则Hidden layer可表示为：

$$\underset{\text{N×1}}{\mathbf{h}} = \underset{\text{N×V}}{\mathbf{W}^T} \underset{\text{V×1}}{\mathbf{x}} = \mathbf{W}^T_{(k,\cdot)} = \mathbf{v}^T_{w_I} \tag{1}$$

因为$\mathbf{x}$是独热向量，所以Hidden layer $\mathbf{h}$ 可以看做是$\mathbf{W}$的第k行的拷贝。

2. Hidden layer ——> Output layer

分两步操作：

1. 计算输入词再字典中的score（词向量从N到V维的线性变换）；
2. 构造条件概率函数（建立词向量与条件概率的关系，有了条件概率，就可以写出最大似然目标函数）

这里先单独考察词表中的任意一个词的得分u_j（先不直接计算$\mathbf{W'}^T\mathbf{h}$）

$$u_j = \underset{\text{1×N}}{\mathbf{v}'^T_{w_j}} \underset{\text{N×1}}{\mathbf{h}} \quad \text{for} \quad j = 1, 2, \cdots, V \tag{2}$$

利用softmax函数，可以得到词的后验概率：

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'}^{V} \exp(u_{j'})}, \quad j = 1, 2, \cdots, V \tag{3}$$

y_j是output layer的第j个元素。

将式（1）和（2）带入式（3），得到：

$$p(w_j|w_I) = \frac{\exp(\mathbf{v}'^T_{w_j} \mathbf{v}_{w_I})}{\sum_{j'=1}^{V} \exp(\mathbf{v}'^T_{w_{j'}} \mathbf{v}_{w_I})}, \quad j = 1, 2, \cdots, V \tag{4}$$

理解：

1. $\mathbf{v}_w$和$\mathbf{v}'_w$都是词w的向量表示；
2. $\mathbf{v}_w$是权重矩阵$\mathbf{W}$的行向量；$\mathbf{v}'_w$是权重矩阵$\mathbf{W}'$的行向量；
3. $\mathbf{v}_w$称之为**input vector**， $\mathbf{v}'_w$称之为**output vector**；
4. $\mathbf{v}'_{w_j}$和$\mathbf{v}_{w_I}$越相似，则条件概率越大

1.1.2 反向传播更新参数

写出优化目标函数，利用反向传播算法更新参数（由式(4)可以看出，目标函数的参数应该是$\mathbf{v}_{w_I}$和$\mathbf{v}'_{w_j}$）

设真实的目标值是w_O，它在output layer的索引是j^*，则目标函数可以表示为：

$$\begin{aligned} \text{maximize } p(w_O|w_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E \end{aligned} \tag{7}$$

这里，$E = \log p(w_O|w_I)$，这个损失函数可以理解为一种特殊的交叉熵。

1. 有了目标函数，首先推导$\mathbf{v}'_{w_j}$的更新更新公式

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j, \quad j = 1, 2, \cdots, V \tag{8}$$

这里，$t_j = \mathbb{I}(j = j^*)$，只有当第$j$个unit代表的是真实的输出词时，$t_j$才等于1，否则$t_j = 0$

接着，利用链式求导法则，可以求出E关于矩阵$\mathbf{W}'$中元素$w'_{i,j}$ 的偏导数：

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i \tag{9}$$

利用随机梯度下降法，可以得到Hidden layer到output layer的权重更新公式如下：

$$w'^{(new)}_{ij} = w'^{(old)}_{ij} - \eta \cdot e_j \cdot h_i \tag{10}$$

or

$$\mathbf{v}'^{(new)}_{w_j} = \mathbf{v}'^{(old)}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \quad \text{for } j = 1, 2, \cdots, V \tag{11}$$

更新公式理解：

1. $\eta > 0$，学习率；$e_j = y_j - t_j$，$\mathbf{h} = \mathbf{v}_{w_I}$
2. 通过更新公式可以看出，每一次更新需遍历词表中的每一个单词（弊端），计算出它的输出概率y_j，并和期望输出t_j（0 or 1）进行比较：
   - 如果$y_j > t_j \Rightarrow e_j > 0$，("overestimating"，这种情况是：$t_j = 0$, 即输出的第j个单词并不是真实的上下文词)，那么就从$\mathbf{v}'_{w_j}$中减去隐藏向量$\mathbf{h}$中的一部分（比如$\mathbf{v}_{w_I}$），这样向量$\mathbf{v}'_{w_j}$就会与向量$\mathbf{v}_{w_I}$相差更远；
   - 如果$y_j < t_j \Rightarrow e_j < 0$，("underestimating"，这种情况是：$t_j = 1$，即输出的第j个词确实是真实的上下文词)，那么就从$\mathbf{v}'_{w_j}$中加上隐藏向量$\mathbf{h}$中的一部分，这样向量$\mathbf{v}'_{w_j}$就会与向量$\mathbf{v}_{w_I}$更

> 接近；
> - 如果y_j和t_j非常接近，那么更新参数基本上没啥变化。

这里再强调一遍，$\mathbf{v}'_w$和$\mathbf{v}_w$是单词w的两种不同的向量表示形式。

2. 介绍完输出向量$\mathbf{v}'_w$的更新公式后，接下来介绍输入向量$\mathbf{v}_w$的更新公式

首先，对隐藏层单元h_i求偏导：

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathrm{EH}_i \quad \text{for } i = 1, 2, \cdots, N \tag{12}$$

EH是一个N维向量。

将式（1）$\mathbf{h} = \mathbf{W}^T \mathbf{x} = \mathbf{W}_{(k,\cdot)} = \mathbf{v}^T_{w_I}$展开：

$$h_i = \sum_{k=1}^{V} x_k \cdot w_{ki} \tag{13}$$

于是，利用链式求导法则，可得到E关于w_{ki}的偏导数：

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \mathrm{EH}_i \cdot x_k \tag{14}$$

利用张量乘积的方式，可以得到：

$$\frac{\partial E}{\partial \mathbf{W}} = \begin{bmatrix} \mathrm{EH}_1 \cdot x_1 & \mathrm{EH}_2 \cdot x_1 & \cdots & \mathrm{EH}_N \cdot x_1 \\ \mathrm{EH}_1 \cdot x_2 & \mathrm{EH}_2 \cdot x_2 & \cdots & \mathrm{EH}_N \cdot x_2 \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{EH}_1 \cdot x_V & \mathrm{EH}_2 \cdot x_V & \cdots & \mathrm{EH}_2 \cdot x_V \end{bmatrix} = \mathbf{x} \otimes \mathrm{EH} = \mathbf{x}\mathrm{EH}^T \tag{15}$$

对上式分析可得：

> 1. 得到一个$V \times N$矩阵；
> 2. 因为$\mathbf{x}$中只有一项不等于0，所以矩阵$\frac{\partial E}{\partial \mathbf{W}}$中只有一行不等0，其它行都为0；

于是，得到$\mathbf{W}$的更新公式为：

$$\mathbf{v}^{(new)}_{w_I} = \mathbf{v}^{(old)}_{w_I} - \eta \cdot \mathrm{EH}^T \tag{16}$$

这里，$\mathbf{v}_{w_I}$是矩阵$\mathbf{W}$的行向量，是唯一的上下文词的"input vector"，也是矩阵中唯一的导数不等于0的行。其它行元素不会更新，因为它们的导数为0。

更新公式理解：

> $\mathrm{EH}_i = \sum_{j=1}^{V} e_j \cdot w'_{ij}$，可以看到$\mathrm{EH}$向量其实是字典中每个词的output vector的加权求和（权重为预测误差$e_j = y_j - t_j$），所以：
>
> 1. 如果输出的第 j 个词w_j 被高估了（即，$y_j > t_j$），那么输入词向量w_I就会远离这个输出词向量w_j

(即 $w_I \cdot w_j >> 0$);
2. 相反地，如果输出的词w_j被低估了，(即，$y_j < t_j$)，那么输入词向量w_I就会靠近w_j($w_I \cdot w_j >> 1$);
3. 如果w_j的概率很准确，那么w_I几乎不怎么变。

1.2 多个上下文词的CBOW模型

多个上下文词的CBOW模型如下图所示：

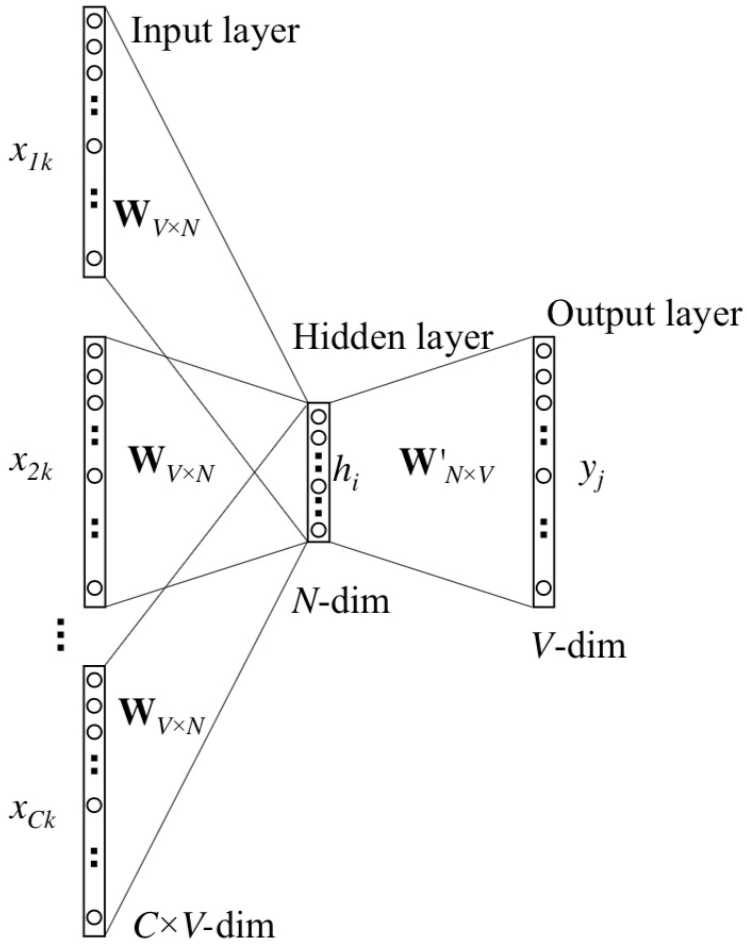


Figure 2: Continuous bag-of-word model

因为有多个上下文词，隐藏层的输出向量$\mathbf{h}$取得是每个词向量乘权重矩阵$\mathbf{W}$后的平均：

$$\mathbf{h} = \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) \tag{17}$$

$$= \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C}) \tag{18}$$

符号说明：

1. $w_1, w_2, \cdots, w_C$为C个上下文词；
2. $\mathbf{x}_w$是词w的输入向量

损失函数：

$$
\begin{aligned}
E &= -p(w_O|w_{I,1}, w_{I,2}, \cdots, w_{I,C}) \\
&= -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \\
&= -\mathbf{v'_{w_O}}^T \cdot \mathbf{h} + \log \sum_{j'=1}^{V} \exp(\mathbf{v'_{w_j}}^T \cdot \mathbf{h})
\end{aligned} \tag{21}
$$

目标函数与1.1中一个词的情况是一样的，除了$\mathbf{h}$的定义不一样。

权重矩阵$\mathbf{W'}$的更新公式也不变：

$$
\mathbf{v'_{w_j}}^{(new)} = \mathbf{v'_{w_j}}^{(old)} - \eta \cdot e_j \cdot \mathbf{h} \quad \text{for } j = 1, 2, \cdots, V \tag{22}
$$

权重矩阵 $\mathbf{W}$的更新公式也类似：

$$
\mathbf{v}_{w_{I,c}}^{(new)} = \mathbf{v}_{w_{I,c}}^{(old)} - \frac{1}{C} \cdot \eta \cdot \mathrm{EH}^T \quad \text{for } c = 1, 2, \cdots, C \tag{23}
$$

2. Skip-Gram模型

Skip-Gram模型正好与CBOW相反，它是根据中心词预测上下文词，模型示意图如下：

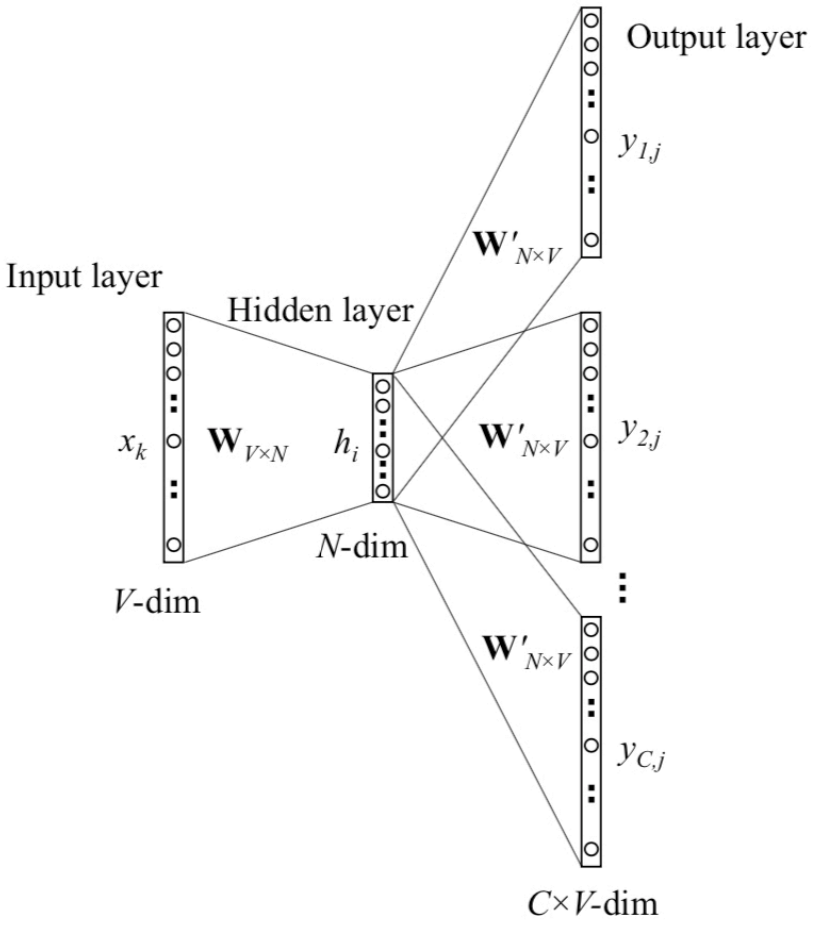


Figure 3: The skip-gram model.

2.1.1 前向传播

符号定义还是不变，

$$\mathbf{h} = \mathbf{W}^T\mathbf{x} = \mathbf{W}^T_{(k,\cdot)} = \mathbf{v}^T_{w_I} \tag{24}$$

在输出层，skip-gram模型的输出是C个多项式分布，但它们共用一个权重矩阵$\mathbf{W}'_{N\times V}$

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V}\exp(u_{j'})} \tag{25}$$

符号说明：

> 1. $w_{c,j}$是输出层第c个pannel(有几个上下文词，就有几个pannel)的第j 个词；
> 2. $w_{O,c}$是输出上下文词中的第c 个词；
> 3. w_I是唯一的输入单词；
> 4. $y_{c,j}$为输出层的第c个panel上的第j个神经单元的概率输出值；
> 5. $u_{c,j}$是输出层第c 个pannel上的第j 个神经元的输入值（score）

注意到输出层所有pannel都共享一个权重矩阵$\mathbf{W}'$，因此每个pannel的输入$u_{c,j}$都是一样的：

$$u_{c,j} = u_j = {\mathbf{v}'_{w_j}}^T \cdot \mathbf{h}, \quad \text{for } c = 1, 2, \cdots, C \tag{26}$$

2.1.2 反向更新参数

损失函数定义为：

$$\begin{aligned} E &= -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} | w_I) \\ &= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'}^{V} \exp(u_{j'})} \\ &= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'}^{V} \exp(u_{j'}) \end{aligned} \tag{29}$$

> 说明：j_c^*是输出层第c个pannel(也即是第j个上下文词)在字典中的索引。

对E求$u_{c,j}$的偏导：

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j} \tag{30}$$

为了表示方便，作者定义了一个V维向量$\mathrm{EI} = \{\mathrm{EI}_1, \mathrm{EI}_2, \cdots, \mathrm{EI}_V\}$表示每一个上下文词的预测误差之和：

$$\mathrm{EI}_j = \sum_{c=1}^{C} e_{c,j} \tag{31}$$

接下来，求损失函数E关于W'的偏导数：

$$\begin{aligned} \frac{\partial E}{\partial w'_{i,j}} &= \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w'_{i,j}} = \sum_{c=1}^{C} e_{c,j} \cdot h_i \\ &= h_i \cdot \sum_{c=1}^{C} e_{c,j} = \mathrm{EI}_j \cdot h_i \end{aligned} \tag{32}$$

从而，可以得到W'（$N \times V$维）的更新公式为：

$${w'_{ij}}^{(new)} = {w'_{ij}}^{(old)} - \eta \cdot \mathrm{EI}_j \cdot h_i \tag{33}$$

or

$${\mathrm{v}'_{w_j}}^{(new)} = {\mathrm{v}'_{w_j}}^{(old)} - \eta \cdot \mathrm{EI}_j \cdot \mathrm{h} \quad \text{for } j = 1, 2, \cdots, V \tag{34}$$

> 说明：
>
> 1. 对更新公式的理解和式(11)一样，只是输出层的预测误差是基于C个上下文词；
> 2. 对每一个训练样本，需要利用该更新公式更新W'的每一项。

Input layer到Hidden layer的权重矩阵$\mathbf{W}$的更新公式与(16)一样：

$$\mathbf{v}_{w_I}^{(new)} = \mathbf{v}_{w_I}^{(old)} - \eta \cdot \mathrm{EH}^T \tag{35}$$

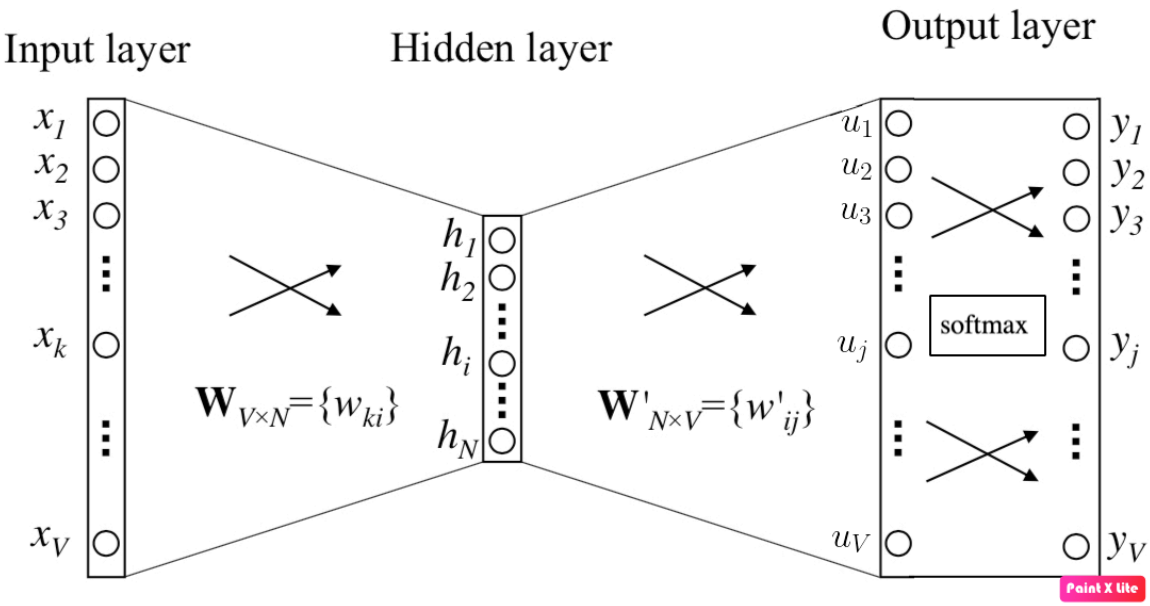
$$\text{where} \qquad \mathrm{EH}_i = \sum_{j=1}^{V} \mathrm{EI}_j \cdot w'_{ij} \tag{36}$$

3. 计算优化

3.1 问题分析

前面两节讨论了CBOW和Skip-Gram两个模型的原始形式。

为了更好的引出优化技巧，对模型示意图中的output layer详细画了下：



对模型的原始形式进行分析：

1. 模型的参数是input vector $\mathbf{v}_w$和output vector $\mathbf{v}'_w$，它们其实是一个词的两个表示向量；
2. 参数$\mathbf{v}_w$的学习成本不高，但是$\mathbf{v}'_w$的学习成本(时间复杂度)就很高了，分析如下：

首先回顾下更新$\mathbf{v}'_w$时涉及到的计算：

$$\begin{cases} \mathbf{v}'^{(new)}_{w_j} = \mathbf{v}'^{(old)}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \\ e_j = y_j - t_j \\ y_j = \dfrac{\exp(u_j)}{\sum_{j'}^{V} \exp(u_{j'})} \\ u_j = \mathbf{v}'^{T}_{w_j} \mathbf{h} \end{cases} \qquad \text{for } j = 1, 2, \cdots, V$$

通过观察上面的更新计算公式可以发现，对于每一个训练样本，都需要遍历字典中的每一个词w_j，计算u_j、y_j、e_j，最终更新$\mathbf{v}'_{w_j}$，这个计算开销是很大的！

为了解决这个问题，一种直观的方法是每次迭代限制需要更新的输出向量，一种有效的实现是使用hierarchical softmax；另外一种方法是通过采样解决。

3.2 Hierarchical Softmax

Hierarchical Softmax是计算softmax的一种高效的方法。Hierarchical Softmax将词典构建成一个棵Huffman Tee。

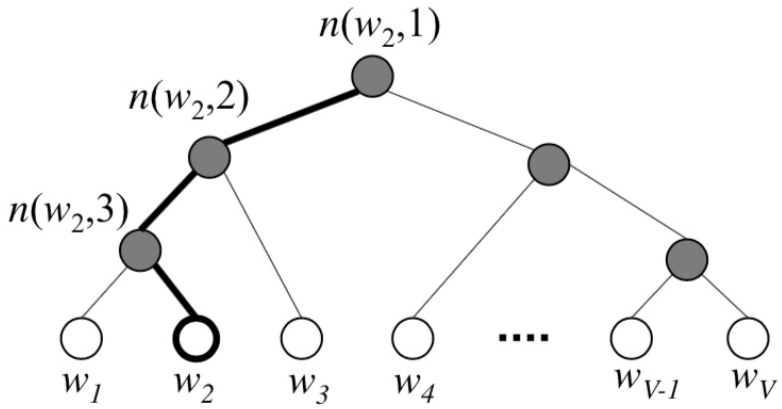


Figure 4: An example binary tree for the hierarchical softmax model

关于这棵树模型的说明：

1. V个单词必须是在叶节点上；如此，就有$V-1$个内部节点；
2. 对于每个叶节点，只存在一条路径从根节点通向该节点；
3. 到达单词w的路径长度记为$L(w)$，如图，$L(w_2)=4$；
4. $n(w,j)$表示w路径上的第j个节点；

在Hierarchical Softmax模型中，不再有词的输出向量（output vector）这种表达。而是$V-1$个内部节点都有一个输出向量$\mathbf{v}'_{n(w,j)}$。因此一个单词作为输出单词的概率计算公式定义如下：：

$$p(w=w_O)=\prod_{j=1}^{L(w)-1}\sigma\left\{\mathbb{I}\left[n(w,j+1)=\mathrm{ch}(n(w,j))\right]\cdot {\mathbf{v}'_{n(w,j)}}^T\cdot\mathbf{h}\right\} \tag{37}$$

公式符号说明：

1. $\mathrm{ch}(n)$表示节点n的左子节点；
2. $\mathbf{v}'_{n(w,j)}$是内部节点$n(w,j)$的向量表达；
3. $\mathbf{h}$是隐藏层的输出向量（Skip-Gram模型对应$\mathbf{h}=\mathbf{v}_{w_I}$，CBOW模型对应$\mathbf{h}=\frac{1}{C}\sum_{c=1}^{C}\mathbf{v}_{w_v}$）；
4. $\mathbb{I}[x]$指示函数，定义如下：

$$\mathbb{I}[x]=\begin{cases}1 & \text{if x is true}\\ -1 & \text{otherwise}\end{cases} \tag{38}$$

结合Figure 4对公式（37）的理解：

> 如果定义w_2是输出词的概率呢？这里将其概率定义为从根节点随机游走到w_2的概率；

> 在每个内部节点（包括根节点），需要定义接下来走到左子树和右子树的概率。这里将在节点n处，走向左子树的概率定义为：
>
> $$p(n,left)=\sigma({\mathbf{v}'_n}^T\cdot\mathbf{h}) \tag{39}$$
>
> 相应地，走向右子树的概率定义为：
>
> $$p(n,right)=1-\sigma({\mathbf{v}'_n}^T\cdot\mathbf{h})=\sigma(-{\mathbf{v}'_n}^T\cdot\mathbf{h}) \tag{40}$$
>
> 如此，那么w_2是输出词的概率为：
>
> $$\begin{aligned}p(w_2=w_O)&=p(n(w_2,1),left)\cdot p(n(w_2,2),left)\cdot p(n(w_2,3),right)\\&=\sigma({\mathbf{v}'_{n(w_2,1)}}^T\cdot\mathbf{h})\cdot\sigma({\mathbf{v}'_{n(w_2,2)}}^T\cdot\mathbf{h})\cdot\sigma(-{\mathbf{v}'_{n(w_2,3)}}^T\cdot\mathbf{h})\end{aligned} \tag{42}$$
>
> 不难证明，所有概率求和等于1：
>
> $$\sum_{i=1}^{V}p(w_i=w_O)=1 \tag{43}$$

接下来推导内部节点上的向量的更新公式，首先考虑只有一个上下文词的情况，记符号：

$$\mathbb{I}[\cdot]:=\mathbb{I}[n(w,j+1)=\mathrm{ch}(n(w,j))] \tag{44}$$

$$\mathbf{v}'_j:=\mathbf{v}'_{n_{w,j}} \tag{45}$$

给定一个训练样本，误差函数定义为：

$$E=-\log p(w=w_O|w_I)=-\sum_{j=1}^{L(w)-1}\log\sigma(\mathbb{I}[\cdot]{\mathbf{v}'_\mathrm{j}}^T\mathbf{h}) \tag{46}$$

求E对${\mathbf{v}'_j}^T\mathbf{h}$的偏导：

$$\begin{aligned}\frac{\partial E}{\partial {\mathbf{v}'_j}^T\mathbf{h}}&=\left(\sigma(\mathbb{I}[\cdot]{\mathbf{v}'_\mathrm{j}}^T\mathbf{h})-1\right)\mathbb{I}[\cdot]\\&=\begin{cases}\sigma({\mathbf{v}'_j}^T\mathbf{h})-1 & (\mathbb{I}[\cdot]=1)\\\sigma({\mathbf{v}'_j}^T\mathbf{h}) & (\mathbb{I}[\cdot]=-1)\end{cases}\\&=\sigma({\mathbf{v}'_j}^T\mathbf{h})-t_j\end{aligned} \tag{49}$$

上式中，当$\mathbb{I}[\cdot]=1$时，$t_j=1$，否则$t_j=0$.

> 补充：$\sigma(x)$函数性质：
>
> 1. $\sigma(x)=\frac{1}{1+e^{-x}}$
> 2. $\sigma'(x)=\sigma(x)(1-\sigma(x))$
> 3. $\sigma(-x)=1-\sigma(x)$
> 4. $[\log\sigma(x)]'=1-\sigma(x)$

5. $[\log(1-\sigma(x))]' = -\sigma(x)$

进一步，可以求得损失函数E对于内部节点的向量表达$\mathbf{v}'_j$的偏导：

$$\frac{\partial E}{\partial \mathbf{v}'_j} = \frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_j \mathbf{h}}{\partial \mathbf{v}'_j} = \left(\sigma({\mathbf{v}'_j}^T \mathbf{h}) - t_j\right) \cdot \mathbf{h} \quad \text{for } \ j = 1, 2, \cdots, L(w) - 1 \tag{50}$$

因此，可以得到输出向量$\mathbf{v}'_j$的更新公式：

$${\mathbf{v}'_j}^{(new)} = {\mathbf{v}'_j}^{(old)} - \eta \left(\sigma({\mathbf{v}'_j}^T \mathbf{h}) - t_j\right) \cdot \mathbf{h} \quad \text{for } \ j = 1, 2, \cdots, L(w) - 1 \tag{51}$$

对更新公式的理解：

1. $\sigma({\mathbf{v}'_j}^T \mathbf{h}) - t_j$可理解为在内部节点$n(w, j)$上的预测误差；
2. 学习的"任务"可理解为在随机游走过程中，在某个内部节点上，下一步分别走向左子树和右子树的概率；
3. $t_j = 1$意味着节点的路径指向左子树，$t_j = 0$意味着节点的路径指向右子树，$\sigma({\mathbf{v}'_j}^T \mathbf{h})$是预测结果；
4. 在训练过程中，$\mathbf{v}'_j$会根据预测的结果做更新（靠近或远离$\mathbf{h}$）；
5. 该更新公式同时适用于CBOW模型和Skip-Gram模型，对于Sikp-Gram模型，因为有C个上下文词，因此在一次训练过程中，需执行C遍更新操作。

推导出$\mathbf{v}'_j$后，接下来推导输入向量$\mathbf{v}_w$的更新公式：

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{h}} &= \sum_{j=1}^{L(w)-1} \frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_j \mathbf{h}}{\partial \mathbf{h}} \\ &= \sum_{j=1}^{L(w)-1} \left(\sigma({\mathbf{v}'_j}^T \mathbf{h}) - t_j\right) \cdot \mathbf{v}_j \\ &:= \mathrm{EH} \end{aligned} \tag{54}$$

带入式(23)可以得到CBOW模型输入向量的更新公式：

$$\mathbf{v}_{w_{I,c}}^{(new)} = \mathbf{v}_{w_{I,c}}^{(old)} - \frac{1}{C} \cdot \eta \cdot \mathrm{EH}^T \quad \text{for } \ c = 1, 2, \cdots, C$$

带入式(35)可以得到Skip-Gram模型的输入向量的更新公式：

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

总结：

对比下原始形式和Hierarchical Softmax形式下的$\mathbf{v}'_{w_j}$和$\mathbf{v}'_j$的更新公式，参考公式(2)~(11)和公式(51)：

1. 原始形式：

$$\mathbf{v}'^{(new)}_{w_j} = \mathbf{v}'^{(old)}_{w_j} - \eta \cdot \left(\frac{\exp(\mathbf{v}'^{T}_{w_j}\mathbf{h})}{\sum_{j'}^{V} \exp(\mathbf{v}'^{T}_{w_{j'}}\mathbf{h})} - t_j \right) \cdot \mathbf{h}$$

2. Hierarchical Softmax形式：

$$\mathbf{v}'^{(new)}_{j} = \mathbf{v}'^{(old)}_{j} - \eta \left(\sigma(\mathbf{v}'^{T}_{j}\mathbf{h}) - t_j \right) \cdot \mathbf{h}$$

可以看到Hierarchical Softmax形式，更新$\mathbf{v}'_j$时，与其它内部节点无关。两个模型在参数几乎相等(原始形式有V个单词，Hierarchical Softmax形式有$V-1$个内部节点)的情况下，时间复杂度从$O(V)$降到了$O(\log(V)$.

3.3 Negative Sampling

Negative Sampling的思想比较直接：每次迭代的时候，根据采样结果，部分更新输出矩阵。

很显然，正样本应该在我们的样本集合中，同时需要采样一部分词作为负样本。在采样的过程中，我们可以任意选择一种概率分布。我们将这种概率分布称为“噪声分布”（the noise distribution），用$P_n(w)$来表示。我们可以根据经验选择一种较好的分布。

在 word2vec中，作者没有使用一种能够产生良好定义的后验多项式分布的负采样形式(不是特别理解？)，而是使用下面的训练目标函数：

$$E = -\log \sigma \left(\mathbf{v}'^{T}_{w_O}\mathbf{h} \right) - \sum_{w_j \in \mathcal{W}_{neg}} \log \sigma \left(-\mathbf{v}'^{T}_{w_j}\mathbf{h} \right) \tag{55}$$

上式中：

1. w_O是输出词(即正样本)，$\mathbf{v}'_{w_O}$是其对应的输出向量；
2. $\mathbf{h}$是隐藏层的输出向量：
   - COBW：$\mathbf{h} = \frac{1}{C}\sum_{c=1}^{C} \mathbf{v}_{w_c}$；
   - Skip-Gram：$\mathbf{h} = \mathbf{v}_{w_I}$
3. $\mathcal{W}_{neg} = \{w_j | j = 1, 2, \cdots, K\}$负样本集合

接下来的推导与之前并无二致。

$$\frac{\partial E}{\partial \mathbf{v}'^{T}_{w_j}\mathbf{h}} = \begin{cases} \sigma(\mathbf{v}'^{T}_{w_j}\mathbf{h}) - 1 & \text{if } w_j = w_O \\ \sigma(\mathbf{v}'^{T}_{w_j}\mathbf{h}) & \text{if } w_j \in \mathcal{W}_{neg} \end{cases} \tag{56}$$

$$= \sigma(\mathbf{v}'^{T}_{w_j}\mathbf{h}) - t_j \tag{57}$$

上式中，当w_j是正样本时，$t=1$；否则，$t=0$.

进一步对$\mathbf{v}'_{w_j}$求导：

$$\frac{\partial E}{\partial \mathbf{v}'_{w_j}} = \frac{\partial E}{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}}{\partial \mathbf{v}'_{w_j}} = \left(\sigma(\mathbf{v}'_{w_j}{}^T \mathbf{h}) - t_j\right) \mathbf{h} \tag{58}$$

因此$\mathbf{v}'_{w_j}$的更新公式为：

$$\mathbf{v}'_{w_j}{}^{(new)} = \mathbf{v}'_{w_j}{}^{(old)} - \eta \left(\sigma(\mathbf{v}'_{w_j}{}^T \mathbf{h}) - t_j\right) \cdot \mathbf{h} \quad \text{for } w_j \in \{w_O\} \cup \mathcal{W}_{neg} \tag{59}$$

每次迭代，只更新样本集中词的输出向量（output vector），而不是更新整个字典中词的输出向量（output vector），因此，能显著提高计算效率。上述更新公式对CBOW和Skip-Gram模型都适用。

继续计算E关于$\mathbf{h}$的偏导：

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{h}} &= \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{neg}} \frac{\partial E}{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}}{\partial \mathbf{h}} \\ &= \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{neg}} \left(\sigma(\mathbf{v}'_{w_j}{}^T \mathbf{h}) - t_j\right) \cdot \mathbf{v}'_{w_j} \\ &:= \mathrm{EH} \end{aligned} \tag{61}$$

将EH带入式(23)得到CBOW模型输入向量(input vector)的更新公式；

将EH带入式(35)得到Skip-Gram模型输入向量(input vector)的更新公式。

4. 参考

1. word2vec Parameter Learning Explained
2. 《word2vec Parameter Learning Explained》论文学习笔记