词的表达

1. 给定语料库 $\mathbb{D}=\{\mathcal{D}_1,\mathcal{D}_2,\cdots,\mathcal{D}_N\}$ ，其中包含 N 篇文档 。
   - 每篇文档 $\mathcal{D}_i$ 包含单词序列 $(\text{word}_{I_1^i},\text{word}_{I_2^i},\cdots,\text{word}_{I_{n_i}^i})$ ，其中 $I_j^i\in\{1,2,\cdots,V\}$ 表示单词的编号：
     - i 表示第 i 篇文档
     - j 表示文档中的第 j 个单词
     - n_i 表示第 i 篇文档包含 n_i 个单词
     - $v=I_j^i$ 表示第 i 篇文档的第 j 个单词为 word_v
   - 所有的单词来自于词汇表 $\mathbb{V}=\{\text{word}_1,\text{word}_2,\cdots,\text{word}_V\}$，其中 V 表示词汇表的大小。

   词的表达任务要解决的问题是：如何表示每个单词 word_v。

2. 最简单的表示方式是`one-hot`编码。对于词汇表中第 v 个单词 word_v，将其表示为：

$$\text{word}_v\rightarrow(0,0,\cdots,0,1,0,\cdots,0)^T$$

   即第 v 位取值为`1`，剩余位取值为`0`。

   这种表示方式有两个主要缺点：
   - 无法表达单词之间的关系。对于任意一对单词 $(\text{word}_i,\text{word}_j)$，其向量距离均为 $\sqrt{2}$ 。
   - 向量维度过高。对于中文词汇表，其大小可能达到数十万，因此`one-hot`向量的维度也在数十万维。这对于存储、计算都消耗过大。
3. `BOW:Bag of Words`：词在文档中不考虑顺序，这称作词袋模型。

一、向量空间模型 VSM

1. 向量空间模型主要用于文档的表达。
2. 向量空间模型假设单词和单词之间是相互独立的，每个单词代表一个独立的语义单元。实际上，该假设很难满足：
   - 文档中的单词和单词之间存在一定关联性，向量空间模型忽略了上下文的作用。
   - 文档中存在很多的一词多义和多词同义的现象，每个单词并不代表一个独立的语义单元。

1.1 文档-单词 矩阵

1. 给定语料库 $\mathbb{D}$ 和词汇表 $\mathbb{V}$，定义`文档-单词`矩阵为：

$$\begin{array}{c|ccccc} & \text{word}_1 & \text{word}_2 & \text{word}_3 & \cdots & \text{word}_V\\ \hline \\ \mathcal{D}_1 & 0 & 0 & 1 & \cdots & 0\\ \mathcal{D}_2 & 1 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots\\ \mathcal{D}_N & 0 & 1 & 1 & \cdots & 0 \end{array}$$

   令矩阵为 $\mathbf{D}$ ，则：
   - $D(i,j)=1$ 表示文档 $\mathcal{D}_i$ 中含有单词 word_j

- $D(i,j)=0$ 表示文档 $\mathcal{D}_i$ 中不含单词 word_j

于是文档 $\mathcal{D}_i$ 表示为：$\mathcal{D}_i \rightarrow (0,1,0,1,\cdots,0)^T$ ，其中文档 $\mathcal{D}_i$ 中包含的单词对应的位置取值为1，其它位置取值为 0 。

1.2 权重

1. 事实上，文档的上述表达并未考虑单词的顺序，也未考虑单词出现的次数。

   一种改进策略是考虑单词出现的次数，从而赋予`文档-单词`矩阵以不同的权重：

$$\mathbf{D}=\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,V} \\ w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,V} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{N,1} & w_{N,2} & w_{N,3} & \cdots & w_{N,V} \end{bmatrix}$$

   其中 $w_{i,j}$ 表示单词 word_j 在文档 $\mathcal{D}_i$ 中的权重。

   - 如果单词 word_j 在文档 $\mathcal{D}_i$ 中未出现，则权重 $w_{i,j}=0$
   - 如果单词 word_j 在文档 $\mathcal{D}_i$ 中出现，则权重$w_{i,j}\ne 0$

2. 权重$w_{i,j}$ 有两种常用的选取方法：

   - 单词权重等于单词出现的频率`TF`：

$$w_{i,j}=TF(\mathcal{D}_i,\text{word}_j)$$

     函数 $TF(\mathcal{D}_i,\text{word}_j)$ 返回单词 word_j 在文档 $\mathcal{D}_i$ 中出现的频数。

   - 单词权重等于单词的`TF-IDF`：

$$w_{i,j}=TF(\mathcal{D}_i,\text{word}_j)\times IDF(\text{word}_j)$$

     函数 $IDF(\text{word}_j)$ 是单词的逆文档频率：$IDF(\text{word}_j)=\log\frac{N}{DF(\text{word}_j)}$ ，其中：

     - N 为语料库的文档数量
     - $DF(\text{word}_j)$ 为出现单词 word_j 的文档的数量
     - $\frac{DF(\text{word}_j)}{N}$ 为单词 word_j 出现在一篇文档中的概率

3. `TF-IDF` 不仅考虑了单词的局部特征，也考虑了单词的全局特征。

   - 词频 $TF(\mathcal{D}_i,\text{word}_j)$ 描述了单词 word_j 在文档 $\mathcal{D}_i$ 中的局部统计特征
   - 逆文档频率 $IDF(\text{word}_j)$ 描述了单词 word_j 在语料库 $\mathbb{D}$ 中的全局统计特征

1.3 相似度

1. 给定`文档-单词`矩阵，则很容易得到文档的向量表达：

$$\mathcal{D}_i \rightarrow (w_{i,1},w_{i,2},\cdots,w_{i,V})^T$$

2. 给定文档 $\mathcal{D}_i,\mathcal{D}_j$ ，文档的相似度为：

$$similar(\mathcal{D}_i,\mathcal{D}_j)=\cos(\vec{\mathbf{w}}_i,\vec{\mathbf{w}}_j)=\frac{\vec{\mathbf{w}}_i\cdot\vec{\mathbf{w}}_j}{||\vec{\mathbf{w}}_i||\cdot||\vec{\mathbf{w}}_j||}$$

   其中 $\vec{\mathbf{w}}_i=(w_{i,1},w_{i,2},\cdots,w_{i,V})^T,\quad \vec{\mathbf{w}}_j=(w_{j,1},w_{j,2},\cdots,w_{j,V})^T$

二、LSA

1. 潜在语义分析`latent semantic analysis:LSA`的基本假设是：如果两个词多次出现在同一篇文档中，则这两个词具有语义上的相似性。

2.1 原理

1. 给定`文档-单词`矩阵 $\mathbf{D}$：

$$\mathbf{D}=\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,V} \\ w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,V} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{N,1} & w_{N,2} & w_{N,3} & \cdots & w_{N,V} \end{bmatrix}$$

   其中 $w_{i,j}$ 表示单词 word_j 在文档 $\mathcal{D}_i$ 中的权重，可以为：

   - 单词 word_j 在文档 $\mathcal{D}_i$ 中是否出现的`0/1`值。
   - 单词 word_j 在文档 $\mathcal{D}_i$ 中出现的频次。
   - 单词 word_j 在文档 $\mathcal{D}_i$ 中的`TF-IDF`值。

2. 定义 $\vec{\mathbf{v}}_v=(w_{1,v},w_{2,v},\cdots,w_{N,v})^T$， 它为矩阵 $\mathbf{D}$ 的第 v 列，代表单词 word_v 的`单词-文档向量`，描述了该单词和所有文档的关系。

   - 向量内积 $\vec{\mathbf{v}}_p\cdot\vec{\mathbf{v}}_q$ 描述了单词 word_p 和单词 word_q 在文档集合中的相似性。
   - 矩阵乘积 $\mathbf{D}^T\mathbf{D}\in\mathbb{R}^{V\times V}$ 包含了所有词向量内积的结果。

3. 定义 $\vec{\mathbf{w}}_i=(w_{i,1},w_{i,2},\cdots,w_{i,V})^T$ ，它为矩阵 $\mathbf{D}$ 的第 i 行，代表文档 $\mathcal{D}_i$ 的`文档-单词`向量，描述了该文档和所有单词的关系。

   - 向量内积 $\vec{\mathbf{w}}_i\cdot\vec{\mathbf{w}}_j$ 描述了文档 $\mathcal{D}_i$ 和文档 $\mathcal{D}_j$ 在文档集合中的相似性。
   - 矩阵乘积 $\mathbf{D}\mathbf{D}^T\in\mathbb{R}^{N\times N}$ 包含了所有文档向量内积的结果。

4. 对矩阵 $\mathbf{D}$ 进行`SVD`分解。假设矩阵 $\mathbf{D}$ 可以分解为：

$$\mathbf{D}=\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T$$

   其中：

   - $\mathbf{P}\in\mathbb{R}^{N\times N},\mathbf{Q}\in\mathbb{R}^{V\times V}$ 为单位正交矩阵。
   - $\mathbf{\Sigma}\in\mathbb{R}^{N\times V}$ 为广义对角矩阵。

$$\mathbf{\Sigma}=\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}$$

     其中 $\sigma_1\ge\sigma_2\ge\cdots\ge\sigma_r>0$ 称作奇异值。

5. `SVD`分解的物理意义为：主题。

   - 所有的文档一共有 r 个主题，每个主题的强度分别为：$\sigma_1,\sigma_2,\cdots,\sigma_r$ 。

- 第 i 篇文档 $\mathcal{D}_i$ 由这 r 个主题组成，文档的主题分布（称作`文档-主题向量`）为：
$$\vec{\mathbf{u}}^{(i)}=(P(i,1),P(i,2),\cdots,P(i,r))^T$$
- 第 j 个主题由 V 个单词组成，主题的单词分布（称作`主题-单词向量`）为：
$$\vec{\mathbf{r}}^{(j)}=(Q(1,j),Q(2,j),\cdots,Q(V,j))^T$$
- 第 v 个单词也可以由 r 个主题描述，单词的主题分布 （称作`单词-主题向量`）为：
$$\vec{\mathbf{z}}^{(v)}=(Q(v,1),Q(v,2),\cdots,Q(v,r))^T$$
- 根据 $\mathbf{D}=\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T$ 有：
$$\mathbf{D}=\mathbf{P}\begin{bmatrix}1&0&\cdots&0\\0&1&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&1\\0&0&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&0\end{bmatrix}_{N\times r}\begin{bmatrix}\sigma_1&0&\cdots&0\\0&\sigma_2&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&\sigma_r\end{bmatrix}\begin{bmatrix}\sigma_1&0&\cdots&0&0&\cdots&0\\0&\sigma_2&\cdots&0&0&\cdots&0\\\vdots&\vdots&\ddots&\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&\sigma_r&0&\cdots&0\end{bmatrix}_{r\times V}\mathbf{Q}^T$$
则该分解的物理意义为：`文档-单词` 矩阵 = `文档-主题` 矩阵 x `主题强度` x `主题-单词` 矩阵。

6. 将奇异值从大到小进行排列，选择 `top k` 的奇异值来近似 $\mathbf{D}$，即：
$$\mathbf{D}_k=\mathbf{P}_k\mathbf{\Sigma}_k\mathbf{Q}_k^T$$

 其中：
 - $\mathbf{P}_k\in\mathbb{R}^{N\times k}$ ：由 $\mathbf{P}$ 的前 k 列组成。
 - $\mathbf{\Sigma}_k\in\mathbb{R}^{k\times k}$ ：由从大到小排列的最大的 `k` 个奇异值组成的对角矩阵。
 - $\mathbf{Q}_k\in\mathbb{R}^{V\times k}$ ：由 $\mathbf{Q}$ 的前 k 列组成。
7. 选择`top k` 的物理意义为：选择权重最大的前 `k` 个主题。
 - 第 i 篇文档 $\mathcal{D}_i$ 由这 k 个主题组成，文档的主题分布为：
$$\vec{\mathbf{u}}^{(i)}=(P(i,1),P(i,2),\cdots,P(i,k))^T$$
 - 第 j 个主题由 V 个单词组成，主题的单词分布为：
$$\vec{\mathbf{r}}^{(j)}=(Q(1,j),Q(2,j),\cdots,Q(V,j))^T$$
 - 第 v 个单词也可以由 k 个主题描述，单词的主题分布为：
$$\vec{\mathbf{z}}^{(v)}=(Q(v,1),Q(v,2),\cdots,Q(v,k))^T$$

2.2 应用

1. 得到了文档的主题分布、单词的主题分布之后，可以获取文档的相似度和单词的相似度。
 - 文档 $\mathcal{D}_i$ 和文档 $\mathcal{D}_j$ 的相似度：

$$sim(\mathcal{D}_i,\mathcal{D}_j)=\frac{\vec{\mathbf{u}}^{(i)}\cdot\vec{\mathbf{u}}^{(j)}}{||\vec{\mathbf{u}}^{(i)}||\times||\vec{\mathbf{u}}^{(j)}||}$$

- 单词 word_i 和单词 word_j 的相似度：

$$sim(\text{word}_i,\text{word}_j)=\frac{\vec{\mathbf{z}}^{(i)}\cdot\vec{\mathbf{z}}^{(j)}}{||\vec{\mathbf{z}}^{(i)}||\times||\vec{\mathbf{z}}^{(j)}||}$$

2. 虽然获得了文档的单词分布，但是并不能获得主题的相似度。

   因为 $\mathbf{Q}$ 是正交矩阵，因此有：

$$(Q(1,i),Q(2,i),\cdots,Q(V,i))^T\cdot(Q(1,j),Q(2,j),\cdots,Q(V,j))^T=0,\quad i\ne j$$

   则有：

$$sim(\text{theme}_i,\text{theme}_j)=\frac{\vec{\mathbf{r}}^{(i)}\cdot\vec{\mathbf{r}}^{(j)}}{||\vec{\mathbf{r}}^{(i)}||\times||\vec{\mathbf{r}}^{(j)}||}$$
$$=\frac{(Q(1,i),Q(2,i),\cdots,Q(V,i))^T\cdot(Q(1,j),Q(2,j),\cdots,Q(V,j))^T}{||\vec{\mathbf{r}}^{(i)}||\times||\vec{\mathbf{r}}^{(j)}||}=0,\quad i\ne j$$

   因此，任意两个主题之间的相似度为 0 。

3. 因为`文档-主题向量`由 $\mathbf{P}$ 决定，而

$$\mathbf{D}=\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T\to\mathbf{P}=\mathbf{D}\mathbf{Q}\mathbf{\Sigma}^{-1}\to\mathbf{P}^T=\mathbf{\Sigma}^{-1}\mathbf{Q}^T\mathbf{D}^T$$

   而`文档-主题向量`为 $\mathbf{P}$ 的行向量，也就是 $\mathbf{P}^T$ 的列向量。`文档-单词向量`为 $\mathbf{D}$ 的行向量，也就是 $\mathbf{D}^T$ 的列向量。

   因此，对于一篇新的文档 $\mathcal{D}_s$，假设其`文档-单词向量`为 $\vec{\mathbf{w}}_s$ ，则其`文档-主题向量`为：

$$\vec{\mathbf{u}}^{(s)}=\mathbf{\Sigma}_k^{-1}\mathbf{Q}_k^T\vec{\mathbf{w}}_s$$

4. 因为`单词-主题向量`由 $\mathbf{Q}$ 决定，而：

$$\mathbf{D}=\mathbf{P}\mathbf{\Sigma}\mathbf{Q}^T\to\mathbf{Q}^T=\mathbf{\Sigma}^{-1}\mathbf{P}^T\mathbf{D}$$

   而`单词-主题向量` 为 $\mathbf{Q}$ 的行向量，也就是 $\mathbf{Q}^T$ 的列向量。`单词-文档向量` 为 $\mathbf{D}$ 的列向量。

   因此，对于一个新的单词，假设其`单词-文档向量`为 $\vec{\mathbf{v}}_s$ ，则`单词-主题向量`为：

$$\vec{\mathbf{z}}^{(s)}=\mathbf{\Sigma}_k^{-1}\mathbf{P}_k^T\vec{\mathbf{v}}_s$$

5. `LSA` 可以应用在以下几个方面：
   - 通过对文档的`文档-主题向量` 进行比较，从而进行基于主题的文档聚类或者分类。
   - 通过对单词的`单词-主题向量`进行比较，从而用于同义词、多义词进行检测。
   - 通过将`query` 映射到主题空间，进而进行信息检索。

2.3 性质

1. `LSA` 的本质是将矩阵 $\mathbf{D}$ 通过 `SVD` 进行降维。降维主要是由于以下原因：
   - 原始的`文档-单词` 矩阵 $\mathbf{D}$ 太大，计算机无法处理。通过降维，得到原始矩阵的一个近似。
   - 原始的`文档-单词` 矩阵 $\mathbf{D}$ 含有噪音。通过降维，去掉原始矩阵的噪音。

- 原始的 `文档-单词` 矩阵 $\mathbf{D}$ 过于稀疏。通过降维，得到一个稠密的矩阵。

2. `LSA` 的降维可以解决一部分同义词的问题，也可以解决一部分二义性的问题。
    - 经过降维，意义相同的同义词的维度会因降维被合并。
    - 经过降维，拥有多个意义的词，其不同的含义会叠加到对应的同义词所在的维度上。
3. `LSA` 的缺点包括：
    - 产生的主题维度可能存在某些主题可解释性差。即：它们并不代表一个自然语言上的主题。
    - 由于 `Bag of words:BOW` 模型的局限性，它无法捕捉单词的前后顺序关系。

      一个解决方案是：采用二元词组或者多元词组。
    - `LSA` 假设单词和文档形成联合高斯分布。实际观察发现，它们是一个联合泊松分布。

      这种情况下，用 `pLSA` 模型效果会更好。

三、Word2Vec

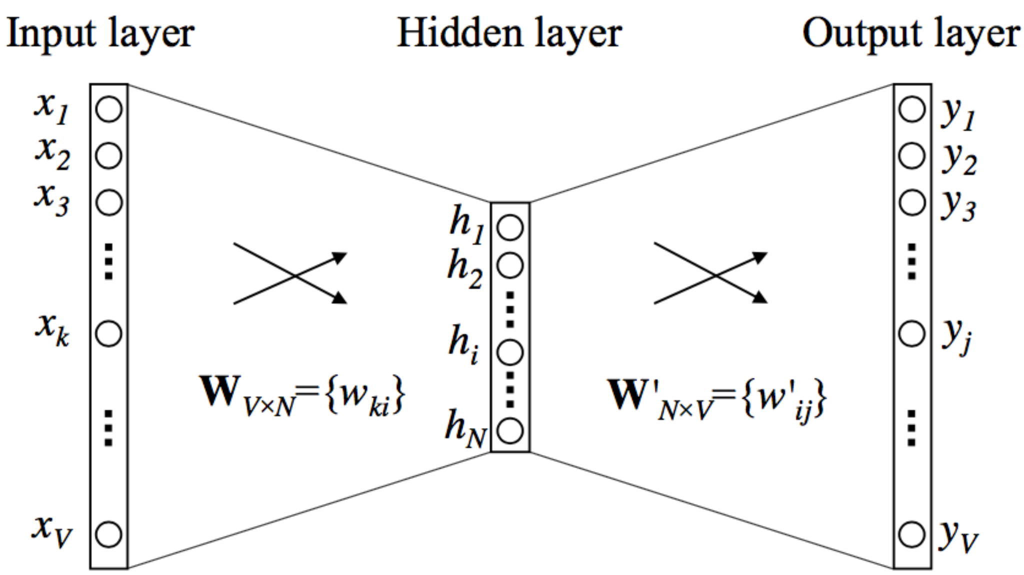
3.1 CBOW 模型

1. `CBOW` 模型（`continuous bag-of-word`）：根据上下文来预测下一个单词。

3.1.1 一个单词上下文

a) 网络结构

1. 在一个单词上下文的 `CBOW` 模型中：
    - 输入是前一个单词，输出是后一个单词。
    - 输入为输出的上下文，由于只有一个单词作为输入，因此称作一个单词的上下文。
2. 一个单词上下文的 `CBOW` 模型如下：



其中：

- N 为隐层的大小，即隐向量 $\vec{\mathbf{h}} \in \mathbb{R}^N$ 。

- 网络输入 $\vec{\mathbf{x}}=(x_1,x_2,\cdots,x_V)^T\in\mathbb{R}^V$ ，它是输入单词（即上下文单词）的 `one-hote` 编码，其中只有一位为 1，其他位都为 0 。
- 网络输出 $\vec{\mathbf{y}}=(y_1,y_2,\cdots,y_V)^T\in\mathbb{R}^V$，它是输出单词为词汇表各单词的概率。
- 相邻层之间为全连接：
  - 输入层和隐层之间的权重矩阵为 $\mathbf{W}$，其尺寸为 $V\times N$ 。
  - 隐层和输出层之间的权重矩阵为 $\mathbf{W}'$，其尺寸为 $N\times V$ 。

3. 假设没有激活函数，没有偏置项。给定输入 $\vec{\mathbf{x}}\in\mathbb{R}^V$，则其对应的隐向量 $\vec{\mathbf{h}}\in\mathbb{R}^N$ 为： $\vec{\mathbf{h}}=\mathbf{W}^T\vec{\mathbf{x}}$ 。

   令：

$$\mathbf{W}=\begin{bmatrix}\vec{\mathbf{w}}_1^T\\\vec{\mathbf{w}}_2^T\\\vdots\\\vec{\mathbf{w}}_V^T\end{bmatrix}$$

   由于 $\vec{\mathbf{x}}$ 是个 `one-hot` 编码，假设它为词表 $\mathbb{V}$ 中第 k 个单词 word_k，即：

$$x_1=0,x_2=0,\cdots,x_{k-1}=0,x_k=1,x_{k+1}=0,\cdots,x_V=0$$

   则有：$\vec{\mathbf{h}}=\vec{\mathbf{w}}_k$。

   即：$\mathbf{W}$ 的第 k 行 $\vec{\mathbf{w}}_k$ 就是词表 $\mathbb{V}$ 中第 k 个单词 word_k 的表达，称作单词 word_k 的输入向量。

4. 给定隐向量 $\vec{\mathbf{h}}$，其对应的输出向量 $\vec{\mathbf{u}}\in\mathbb{R}^V$ 为： $\vec{\mathbf{u}}=\mathbf{W}'^T\vec{\mathbf{h}}$

   令：

$$\mathbf{W}'=[\vec{\mathbf{w}}'_1,\vec{\mathbf{w}}'_2,\cdots,\vec{\mathbf{w}}'_V]$$

   则有：$u_j=\vec{\mathbf{w}}'_j\cdot\vec{\mathbf{h}}$，表示词表 $\mathbb{V}$ 中，第 j 个单词 word_j 的得分。

   $\vec{\mathbf{w}}'_j$ 为矩阵 $\mathbf{W}'$ 的第 j 列，称作单词 word_j 的输出向量。

5. $\vec{\mathbf{u}}$ 之后接入一层 `softmax` 层，则有：

$$y_j=p(\text{word}_j\mid\vec{\mathbf{x}})=\frac{\exp(u_j)}{\sum_{j'=1}^V\exp(u_{j'})},\quad j=1,2,\cdots,V$$

   即 y_j 表示词汇表 $\mathbb{V}$ 中第 j 个单词 word_j 为真实输出单词的概率。

6. 假设给定一个单词 word_I （它称作上下文），观测到它的下一个单词为 word_O 。

   假设 word_O 对应的网络输出编号为 j^* ，则网络的优化目标是：

$$\begin{aligned}\max_{\mathbf{W},\mathbf{W}'}p(\text{word}_O\mid\text{word}_I)&=\max_{\mathbf{W},\mathbf{W}'}y_{j^*}=\max_{\mathbf{W},\mathbf{W}'}\log\frac{\exp(\vec{\mathbf{w}}'_{j^*}\cdot\vec{\mathbf{w}}_I)}{\sum_{i=1}^V\exp(\vec{\mathbf{w}}'_i\cdot\vec{\mathbf{w}}_I)}\\&=\max_{\mathbf{W},\mathbf{W}'}\left[\vec{\mathbf{w}}'_{j^*}\cdot\vec{\mathbf{w}}_I-\log\sum_{i=1}^V\exp(\vec{\mathbf{w}}'_i\cdot\vec{\mathbf{w}}_I)\right]\end{aligned}$$

   其中 $\vec{\mathbf{w}}_I$ 为输入单词 word_I 的输入向量。

7. 考虑到 $u_j=\vec{\mathbf{w}}'_j\cdot\vec{\mathbf{w}}_I$ ，定义：

$$E = -\log p(\text{word}_O \mid \text{word}_I) = -\left[\vec{\mathbf{w}}'_{j^*} \cdot \vec{\mathbf{w}}_I - \log \sum_{i=1}^{V} \exp(\vec{\mathbf{w}}'_i \cdot \vec{\mathbf{w}}_I)\right]$$
$$= -\left[u_{j^*} - \log \sum_{i=1}^{V} \exp(u_i)\right]$$

则优化目标为：$\min E$ 。

b) 参数更新

1. 定义 $t_j = \mathbb{I}(j = j^*)$ ，即第 j 个输出单元对应于真实的输出单词 word_O 时，它为1；否则为0。

   定义：

   $$e_j = \frac{\partial E}{\partial u_j} = y_j - t_j$$

   它刻画了每个输出单元的预测误差：

   - 当 $j = j^*$ 时： $e_j = y_j - 1$，它刻画了输出概率 (y_j) 与真实概率 1 之间的差距。
   - 当 $j \neq j^*$ 时：$e_j = y_j$，它刻画了输出概率 (y_j) 与真实概率 0 之间的差距。
2. 根据：

   $$u_j = \vec{\mathbf{w}}'_j \cdot \vec{\mathbf{h}} \quad \rightarrow \quad \frac{\partial u_j}{\partial \vec{\mathbf{w}}'_j} = \vec{\mathbf{h}}$$

   则有：

   $$\frac{\partial E}{\partial \vec{\mathbf{w}}'_j} = \frac{\partial E}{\partial u_j} \times \frac{\partial u_j}{\partial \vec{\mathbf{w}}'_j} = e_j \vec{\mathbf{h}}$$

   则 $\vec{\mathbf{w}}'_j$ 更新规则为：

   $$\vec{\mathbf{w}}'^{(new)}_j = \vec{\mathbf{w}}'^{(old)}_j - \eta e_j \vec{\mathbf{h}}$$

   其物理意义为：

   - 当估计过量 $(e_j > 0 \rightarrow y_j > t_j)$ 时， $\vec{\mathbf{w}}'_j$ 会减去一定比例的 $\vec{\mathbf{h}}$ 。

     这发生在第 j 个输出单元不对应于真实的输出单词时。
   - 当估计不足 $(e_j < 0 \rightarrow y_j < t_j)$ 时， $\vec{\mathbf{w}}'_j$ 会加上一定比例的 $\vec{\mathbf{h}}$ 。

     这发生在第 j 个输出单元刚好对应于真实的输出单词时。
   - 当 $y_j \simeq t_j$ 时，更新的幅度将非常微小。
3. 定义：

   $$\overrightarrow{\mathbf{EH}} = \frac{\partial E}{\partial \vec{\mathbf{h}}} = \left(\frac{\partial \vec{\mathbf{u}}}{\partial \vec{\mathbf{h}}}\right)^T \frac{\partial E}{\partial \vec{\mathbf{u}}}$$

   根据：

   $$\vec{\mathbf{u}} = \mathbf{W}'^T \vec{\mathbf{h}} \quad \rightarrow \quad \left(\frac{\partial \vec{\mathbf{u}}}{\partial \vec{\mathbf{h}}}\right)^T = \mathbf{W}'$$

   则有：

$$\overrightarrow{\mathbf{EH}} = \mathbf{W}'\vec{\mathbf{e}} = \sum_{j=1}^{V} e_j \vec{\mathbf{w}}'_j$$

$\overrightarrow{\mathbf{EH}}$ 的物理意义为：词汇表 $\mathbb{V}$ 中所有单词的输出向量的加权和，其权重为 e_j 。

4. 考虑到 $\vec{\mathbf{h}} = \mathbf{W}^T\vec{\mathbf{x}}$ ，则有：

$$\frac{\partial E}{\partial w_{k,i}} = \frac{\partial E}{\partial h_i} \times \frac{\partial h_i}{\partial w_{k,i}} = EH_i \times x_k$$

写成矩阵的形式为：$\frac{\partial E}{\partial \mathbf{W}} = \vec{\mathbf{x}} \otimes \overrightarrow{\mathbf{EH}}$ ，其中 $\otimes$ 为克罗内克积。

由于 $\vec{\mathbf{x}}$ 是 `one-hote` 编码，所以它只有一个分量非零，因此 $\frac{\partial E}{\partial \mathbf{W}}$ 只有一行非零，且该非零行就等于 $\overrightarrow{\mathbf{EH}}$ 。因此得到更新方程：

$$\vec{\mathbf{w}}_I^{(new)} = \vec{\mathbf{w}}_I^{(old)} - \eta\overrightarrow{\mathbf{EH}}$$

其中 $\vec{\mathbf{w}}_I$ 为 $\vec{\mathbf{x}}$ 非零分量对应的 $\mathbf{W}$ 中的行，而 $\mathbf{W}$ 的其它行在本次更新中都保持不变。

5. 考虑更新行的第 k 列，则：

$$w_{I,k}^{(new)} = w_{I,k}^{(old)} - \eta\sum_{j=1}^{V} e_j w'_{j,k}$$

   - 当 $y_j \simeq t_j$ 时，更新的幅度将非常微小。
   - 当 y_j 与 t_j 差距越大，则更新的幅度越大。

6. 当给定许多训练样本（每个样本由两个单词组成），上述更新不断进行，更新的效果在不断积累。
   - 根据单词的共现结果，输出向量与输入向量相互作用并达到平衡。
     - 输出向量 $\vec{\mathbf{w}}'$ 的更新依赖于输入向量 $\vec{\mathbf{w}}_I$ ： $\vec{\mathbf{w}}_j'^{(new)} = \vec{\mathbf{w}}_j'^{(old)} - \eta e_j\vec{\mathbf{h}}$ 。

       这里隐向量 $\vec{\mathbf{h}}$ 等于输入向量 $\vec{\mathbf{w}}_I$ 。
     - 输入向量 $\vec{\mathbf{w}}_I$ 的更新依赖于输出向量 $\vec{\mathbf{w}}'$ ： $\vec{\mathbf{w}}_I^{(new)} = \vec{\mathbf{w}}_I^{(old)} - \eta\overrightarrow{\mathbf{EH}}$ 。

       这里 $\overrightarrow{\mathbf{EH}} = \sum_{j=1}^{V} e_j\vec{\mathbf{w}}'_j$ 为词汇表 $\mathbb{V}$ 中所有单词的输出向量的加权和，其权重为 e_j 。
   - 平衡的速度与效果取决于单词的共现分布，以及学习率。

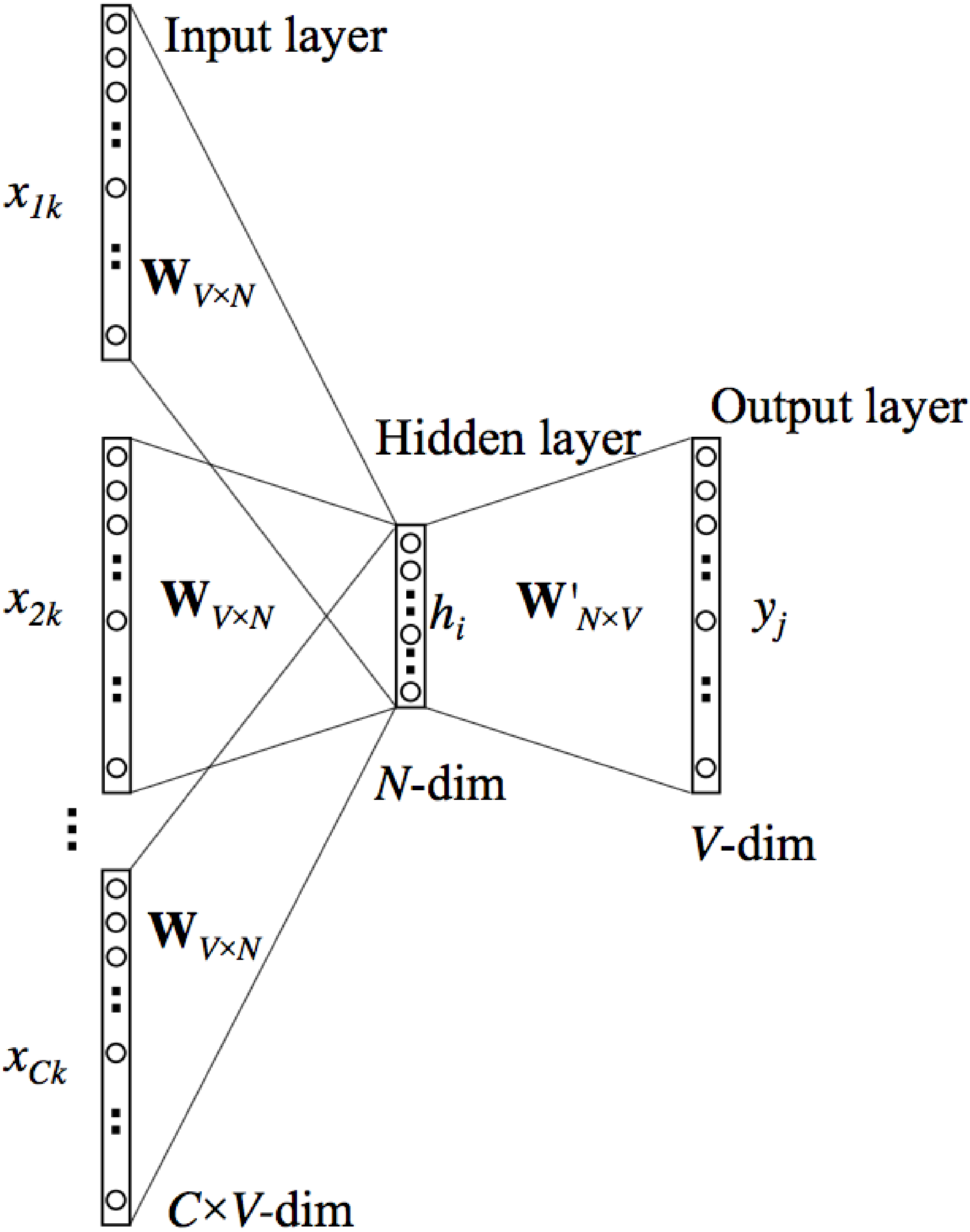
3.1.2 多个单词上下文

1. 考虑输入为多个单词（这些单词作为输出的上下文），输入为 C 个单词：$\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \cdots, \vec{\mathbf{x}}_C$ 。
   - 对于每个输入单词，其权重矩阵都相同，为 $\mathbf{W}$，这称作权重共享。

     这里的权重共享隐含着：每个单词的表达是固定的、唯一的，与它的上下文无关。
   - 隐向量为所有输入单词映射结果的均值：

$$\vec{\mathbf{h}} = \frac{1}{C}\mathbf{W}^T(\vec{\mathbf{x}}_1 + \vec{\mathbf{x}}_2 + \cdots + \vec{\mathbf{x}}_C) = \frac{1}{C}(\vec{\mathbf{w}}_{I_1} + \vec{\mathbf{w}}_{I_2} + \cdots + \vec{\mathbf{w}}_{I_C})$$

     其中：
     - I_i 表示第 i 个输入单词在词汇表 $\mathbb{V}$ 中的编号。
     - $\vec{\mathbf{w}}_j$ 为矩阵 $\mathbf{W}$ 的第 j 行，它是对应输入单词的输入向量。



2. 假设给定一个单词序列 $\text{word}_{I_1},\text{word}_{I_2},\cdots,\text{word}_{I_C}$ （它称作上下文），观测到它的下一个单词为 word_O 。 word_O 对应的网络输出编号为 j^* 。

   定义损失函数为：

$$\begin{aligned}E &= -\log p(\text{word}_O \mid \text{word}_{I_1},\text{word}_{I_2},\cdots,\text{word}_{I_C})\\ &= -u_{j^*} + \log\sum_{i=1}^{V}\exp(u_i)\\ &= -\vec{\mathbf{w}}'_{j^*}\cdot\vec{\mathbf{h}} + \log\sum_{i=1}^{V}\exp(\vec{\mathbf{w}}'_i\cdot\vec{\mathbf{h}})\end{aligned}$$

它的形式与`一个单词上下文`中推导的完全相同，除了这里的 $\vec{\mathbf{h}}$ 不同。

3. 与`一个单词上下文`中推导的结果相同，这里给出参数更新规则：
   - 更新 $\mathbf{W}'$：

$$\vec{\mathbf{w}}_j'^{(new)} = \vec{\mathbf{w}}_j'^{(old)} - \eta e_j \vec{\mathbf{h}}, \quad j = 1, 2, \cdots, V$$

   - 更新 $\mathbf{W}$：

$$\vec{\mathbf{w}}_{I_i}^{(new)} = \vec{\mathbf{w}}_{I_i}^{(old)} - \frac{1}{C} \eta \overrightarrow{\mathbf{EH}}, \quad i = 1, 2, \cdots, C$$

     其中：
     - $\overrightarrow{\mathbf{EH}} = \mathbf{W}'\vec{\mathbf{e}} = \sum_{j=1}^V e_j \vec{\mathbf{w}}_j'$，它是词汇表 $\mathbb{V}$ 中所有单词的输出向量的加权和，其权重为 e_j 。
     - I_i 为第 i 个输入单词在词表 $\mathbb{V}$ 中的编号。
4. 在更新 $\mathbf{W}$ 时，如果有相同的输入单词（如：$\vec{\mathbf{x}}_1 = \vec{\mathbf{x}}_2 \rightarrow \text{word}_{100}$ ），则在参数更新时认为它们是不同的。

   最终的效果就是在 $\vec{\mathbf{w}}_{I_i}$ 中多次减去同一个增量 $\frac{1}{C}\eta\overrightarrow{\mathbf{EH}}$ 。

   你也可以直接减去 $\frac{n_i}{C}\eta\overrightarrow{\mathbf{EH}}$，其中 n_i 为词汇表中单词 word_{I_i} 在输入中出现的次数。

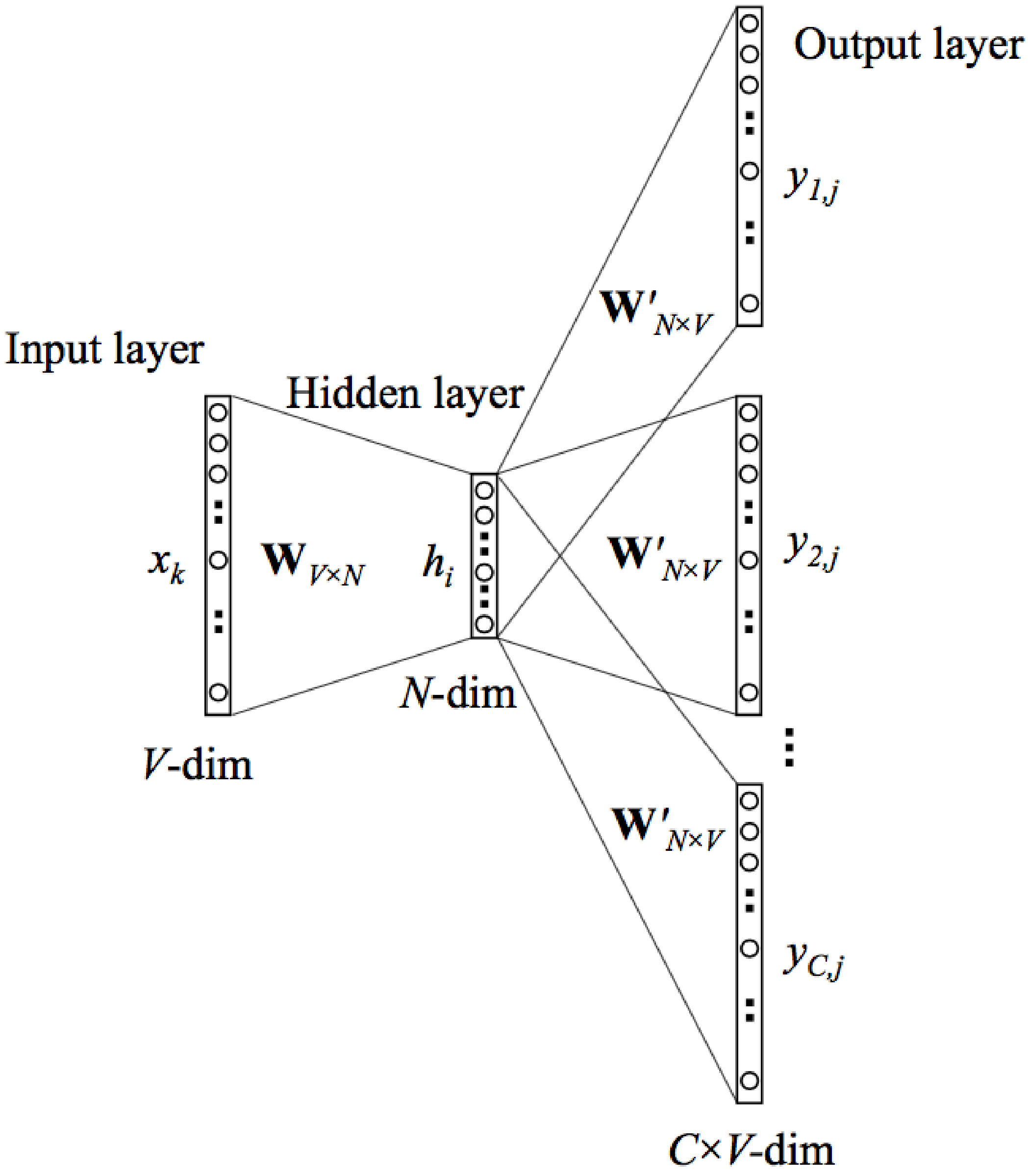
3.2 Skip-Gram

1. `CBOW` 模型是根据前几个单词（即：上下文）来预测下一个单词，而`Skip-Gram` 模型是根据一个单词来预测其前几个单词（即：上下文）。
2. 在`CBOW` 模型中：
   - 同一个单词的表达（即输入向量 $\vec{\mathbf{w}}_I$ ）是相同的，因为参数 $\mathbf{W}$ 是共享的。
   - 同一个单词的输出向量 $\vec{\mathbf{w}}_O'$ 是不同的，因为输出向量随着上下文不同而不同。
3. 在`Skip-Gram` 模型中：
   - 同一个单词的表达（即输出向量 $\vec{\mathbf{w}}_O'$ ）是相同的，因为参数 $\mathbf{W}'$ 是共享的。
   - 同一个单词的输入向量 $\vec{\mathbf{w}}_I$ 是不同的，因为输入向量随着上下文不同而不同。

3.2.1 网络结构

1. `Skip-Gram` 网络模型如下。其中：
   - 网络输入 $\vec{\mathbf{x}} = (x_1, x_2, \cdots, x_V)^T \in \mathbb{R}^V$，它是输入单词的 `one-hote` 编码，其中只有一位为 1，其他位都为 0 。
   - 网络输出 $\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \cdots, \vec{\mathbf{y}}_C$，其中 $\vec{\mathbf{y}}_c = (y_1^c, y_2^c, \cdots, y_V^c)^T \in \mathbb{R}^V$是第 c 个输出单词为词汇表各单词的概率。
   - 对于网络的每个输出 $\vec{\mathbf{y}}_c$，其权重矩阵都相同，为 $\mathbf{W}'$。这称作权重共享。

     这里的权重共享隐含着：每个单词的输出向量是固定的、唯一的，与其他单词的输出无关。



2. `Skip-Gram` 网络模型中，设网络第 c 个输出的第 j 个分量为 $u_j^c = \vec{\mathbf{w}}'_j \cdot \vec{\mathbf{h}}$ ，则有：

$$y_j^c = p(\text{word}_j^c \mid \vec{\mathbf{x}}) = \frac{\exp(u_j^c)}{\sum_{k=1}^{V}\exp(u_k^c)}; \quad c = 1, 2, \cdots, C; \quad j = 1, 2, \cdots, V$$

   y_j^c 表示第 c 个输出中，词汇表 $\mathbb{V}$ 中第 j 个单词 word_j 为真实输出单词的概率。

3. 因为 $\mathbf{W}'$ 在多个单元之间共享，所以对于网络每个输出，其得分分布 $\vec{\mathbf{u}}_c = (u_1^c, u_2^c, \cdots, u_V^c)^T$ 是相同的。但是这并不意味着网络的每个输出都是同一个单词。
   - 并不是网络每个输出中，得分最高的单词为预测单词。因为每个输出中，概率分布都相同，即：$\vec{\mathbf{y}}_1 = \vec{\mathbf{y}}_2 = \cdots = \vec{\mathbf{y}}_C$ 。
   - `Skip-Gram` 网络的目标是：网络的多个输出之间的联合概率最大。
4. 假设输入为单词 word_I ，输出单词序列为 $\text{word}_{O_1}, \text{word}_{O_2}, \cdots, \text{word}_{O_C}$ 。定义损失函数为：

$$E = -\log p(\text{word}_{O_1}, \text{word}_{O_2}, \cdots, \text{word}_{O_C} \mid \text{word}_I) = -\log \prod_{c=1}^{C} \frac{\exp(u_{j_c^*}^c)}{\sum_{k=1}^{V}\exp(u_k^c)}$$

   其中 $j_1^*, j_2^*, \cdots, j_C^*$ 为输出单词序列对应于词典 $\mathbb{V}$ 中的下标序列。

由于网络每个输出的得分分布都相同，令 $u_k = u_k^c = \vec{\mathbf{w}}'_k \cdot \vec{\mathbf{h}}$，则上式化简为：

$$E = -\sum_{c=1}^{C} u_{j_c^*}^c + C \log \sum_{k=1}^{V} \exp(u_k)$$

3.1.2 参数更新

1. 定义 $t_j^c = \mathbb{I}(j_c = j_c^*)$ ，即网络第 c 个输出的第 j 个分量对应于第 c 个真实的输出单词 $\text{word}_{j_c^*}$ 时，它为 1；否则为0。

   定义：

   $$e_j^c = \frac{\partial E}{\partial u_j^c} = y_j^c - t_j^c$$

   它刻画了网络第 c 个输出的第 j 个分量的误差：

   - 当 $j_c = j_c^*$ 时： $e_j^c = y_j^c - 1$，它刻画了输出概率 y_j^c 与真实概率 1 之间的差距
   - 当 $j_c \neq j_c^*$ 时： $e_j^c = y_j^c$，它刻画了输出概率 y_j^c 与真实概率 0 之间的差距

2. 根据：

   $$u_j = \vec{\mathbf{w}}'_j \cdot \vec{\mathbf{h}} \quad \rightarrow \quad \frac{\partial u_j}{\partial \vec{\mathbf{w}}'_j} = \vec{\mathbf{h}}$$

   则有：

   $$\frac{\partial E}{\partial \vec{\mathbf{w}}'_j} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_j^c} \times \frac{\partial u_j^c}{\partial \vec{\mathbf{w}}'_j} = \sum_{c=1}^{C} e_j^c \vec{\mathbf{h}}$$

   定义 $EI_j = \sum_{c=1}^{C} e_j^c$ ，它为网络每个输出的第 j 个分量的误差之和。于是有：

   $$\frac{\partial E}{\partial \vec{\mathbf{w}}'_j} = EI_j \times \vec{\mathbf{h}}$$

   则有更新方程：

   $$\vec{\mathbf{w}}_j'^{(new)} = \vec{\mathbf{w}}_j'^{(old)} - \eta \times EI_j \times \vec{\mathbf{h}}, \qquad j = 1, 2, \cdots, V$$

3. 定义：

$$\overrightarrow{\mathbf{EH}} = \frac{\partial E}{\partial \vec{\mathbf{h}}} = \sum_{c=1}^{C} \left(\frac{\partial \vec{\mathbf{u}}^c}{\partial \vec{\mathbf{h}}}\right)^T \frac{\partial E}{\partial \vec{\mathbf{u}}^c}$$

根据：

$$\vec{\mathbf{u}}^c = \mathbf{W}'^T \vec{\mathbf{h}} \quad \to \quad \left(\frac{\partial \vec{\mathbf{u}}^c}{\partial \vec{\mathbf{h}}}\right)^T = \mathbf{W}'$$

则有：

$$\overrightarrow{\mathbf{EH}} = \sum_{c=1}^{C} \mathbf{W}' \vec{\mathbf{e}}^c = \sum_{j=1}^{V} EI_j \vec{\mathbf{w}}'_j$$

$\overrightarrow{\mathbf{EH}}$ 的物理意义为：词汇表 $\mathbb{V}$ 中所有单词的输出向量的加权和，其权重为 EI_j 。

4. 考虑到 $\vec{\mathbf{h}} = \mathbf{W}^T \vec{\mathbf{x}}$ ，则有：

$$\frac{\partial E}{\partial w_{k,i}} = \frac{\partial E}{\partial h_i} \times \frac{\partial h_i}{\partial w_{k,i}} = EH_i \times x_k$$

写成矩阵的形式为：$\frac{\partial E}{\partial \mathbf{W}} = \vec{\mathbf{x}} \otimes \overrightarrow{\mathbf{EH}}$ ，其中 $\otimes$ 为克罗内克积。

由于 $\vec{\mathbf{x}}$ 是 `one-hote` 编码，所以它只有一个分量非零，因此 $\frac{\partial E}{\partial \mathbf{W}}$ 只有一行非零，且该非零行就等于 $\overrightarrow{\mathbf{EH}}$ 。因此得到更新方程：

$$\vec{\mathbf{w}}_I^{(new)} = \vec{\mathbf{w}}_I^{(old)} - \eta \overrightarrow{\mathbf{EH}}$$

其中 $\vec{\mathbf{w}}_I$ 为 $\vec{\mathbf{x}}$ 非零分量对应的 $\mathbf{W}$ 中的行，而 $\mathbf{W}$ 的其它行在本次更新中都保持不变。

3.3 优化

1. 原始的 `CBOW` 模型和 `Skip-Gram` 模型的计算量太大，非常难以计算。
   - 模型在计算网络输出的时候，需要计算误差 。
     - 需要计算 V 个误差（词汇表的大小），或者 $C \times V$ 个误差（`Skip-Gram` 模型）。
     - 误差的计算需要用到 `softmax` 函数。
   - 每次迭代都需要计算网络输出。

   如果词汇表有 `100万` 单词，模型迭代 `10000` 次，则计算量超过 `100` 亿次。
2. 虽然输入向量的维度也很高，但是由于输入向量只有一位为 1，其它位均为 0，因此输入的总体计算复杂度较小。
3. `word2vec` 优化的主要思想是：限制输出单元的数量。

   事实上在上百万的输出单元中，仅有少量的输出单元对于参数更新比较重要，大部分的输出单元对于参数更新没有贡献。
4. 有两种优化策略：
   - 通过分层 `softmax` 来高效计算 `softmax` 函数。
   - 通过负采样来缩减输出单元的数量。

3.3.1 分层 softmax

1. 分层 `softmax` 是一种高效计算 `softmax` 函数的算法。

2. 经过分层 `softmax` 之后：计算 `softmax` 函数的算法复杂度从 $O(V)$ 降低到 $O(\log V)$ 。

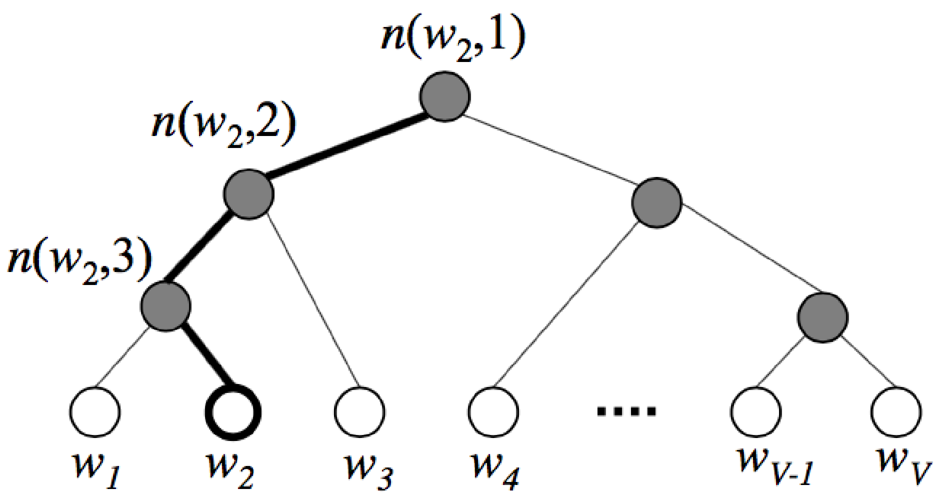
   但是仍然要计算 $V-1$ 个内部节点的向量表达 。

a) 网络结构

1. 在分层 `softmax` 中，字典 $\mathbb{V}$ 中的 V 个单词被组织成二叉树。
    - 叶子结点值为某个具体单词的概率（如下图中的百色结点）
    - 中间节点值也代表一个概率（如下图中的灰色结点）
        - 它的值等于：直系子节点的值之和。
        - 它的值也等于：后继的叶子结点值之和。
        - 它的值也等于：从根节点到当前节点的路径的权重的乘积。

        > 之所以有这些性质，是由于结点值、权重都是概率，满足和为1的性质

    - 根据定义，根节点的值等于所有叶子结点的值之和，即为 1.0 。



2. 二叉树的每条边代表分裂：
    - 向左的边：表示选择左子节点，边的权重为选择左子节点的概率
    - 向右的边：表示选择右子节点，边的权重为选择右子节点的概率
3. 对于任意一个中间节点 t， 假设其向量表达为 $\vec{\mathbf{v}}_t^{\prime}$ ， 它是待求的参数。
    - 选择左子节点的概率为：

$$p(t,left)=\sigma\left(\vec{\mathbf{v}}_t^{\prime}\cdot\vec{\mathbf{h}}\right)$$

    - 选择右子节点的概率为：

$$p(t,right)=1-\sigma\left(\vec{\mathbf{v}}_t^{\prime}\cdot\vec{\mathbf{h}}\right)=\sigma\left(-\vec{\mathbf{v}}_t^{\prime}\cdot\vec{\mathbf{h}}\right)$$

    - 如果求得所有中间节点的向量表达，则根据每个中间节点的分裂概率，可以很容易的求得每个叶节点的值。
4. 在分层 `softmax` 中，算法并不直接求解输出向量 $\{\vec{\mathbf{w}}_1^{\prime},\vec{\mathbf{w}}_2^{\prime},\cdots,\vec{\mathbf{w}}_V^{\prime}\}$，而是求解二叉树的 $V-1$ 个中间节点的向量表达 。
5. 当需要计算某个单词的概率时，只需要记录从根节点到该单词叶子结点的路径。给定单词 w ：

- 定义 $n(w,j)$ 为从根节点到单词 w 的路径的第 j 个节点（从 1 计数）。
- 定义 $L(w)$ 为从根节点到单词 w 的路径的长度。
- 定义 $ch(t)$ 表示节点 t 的左子节点。

输出为单词 w 的概率为：

$$p(w)=\prod_{j=1}^{L(w)-1}\sigma\left(g(n(w,j+1)=ch(n(w,j)))\times\vec{\mathbf{v}}'_{n(w,j)}\cdot\vec{\mathbf{h}}\right)$$

其中：

- $n(w,j+1)=ch(n(w,j))$ 表示：从根节点到单词 w 的路径上，第 $j+1$ 个节点是第 j 个节点的左子节点。
- $g(x)$ 是一个函数。当 x 是个事实时，其值为 1；当 x 不成立时，其值为 -1。

$$g(x)=\begin{cases}1, & \text{if } x \text{ is true}\\ -1, & \text{if } x \text{ is false}\end{cases}$$

- $g(n(w,j+1)=ch(n(w,j)))$ 表示：从根节点到单词 w 的路径上：
  - 当第 $j+1$ 个节点是第 j 个节点的左子节点时，函数值为 1
  - 当第 $j+1$ 个节点是第 j 个节点的右子节点时，函数值为 -1
- $\vec{\mathbf{v}}'_{n(w,j)}$ 表示：从根节点到单词 w 的路径上，第 j 个节点的向量表达
- 对于从根节点到单词 w 的路径上，从第 j 个节点到第 $j+1$ 个节点的概率为：

$$p(j,j+1)=\begin{cases}\sigma\left(\vec{\mathbf{v}}'_{n(w,j)}\cdot\vec{\mathbf{h}}\right), & \text{if } j+1 \text{ is left child of } j\\ \sigma\left(-\vec{\mathbf{v}}'_{n(w,j)}\cdot\vec{\mathbf{h}}\right), & \text{if } j+1 \text{ is right child of } j\end{cases}$$

  因此 $p(w)$ 刻画了：从根节点到单词 w 的路径上，每条边的权重（也就是分裂概率）的乘积。

6. 对于所有的叶子节点，有：

$$\sum_{i=1}^{V}p(w_i)=1$$

利用数学归纳法，可以证明：左子节点的值+右子节点的值=父节点的值。上式最终证明等于根节点的值，也就是 1.0

b) 内部节点更新

1. 为了便于讨论，这里使用`CBOW` 的 `一个单词上下文`模型。

> `CBOW` 的 `多单词上下文`、`Skip-Gram` 模型也类似推导

2. 记 $g(n(w,j+1)=ch(n(w,j)))$ 为 $g_{n(w,j)}$。

定义损失函数对数似然：

$$E=-\log p(w\mid\vec{\mathbf{x}})=-\sum_{j=1}^{L(w)-1}\log\sigma(g_{n(w,j)}\vec{\mathbf{v}}'_{n(w,j)}\cdot\vec{\mathbf{h}})$$

则有：

$$\frac{\partial E}{\partial (\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}})} = \left(\sigma(g_{n(w,j)} \vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}}) - 1\right) g_{n(w,j)} = \begin{cases} \sigma(\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}}) - 1 & \text{if } g_{n(w,j)} = 1 \\ \sigma(\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}}) & \text{if } g_{n(w,j)} = -1 \end{cases}$$
$$= \sigma(\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}}) - t_{n(w,j)}$$

其中：

$$t_{n(w,j)} = \begin{cases} 1, & \text{if node } j+1 \text{ at path , root} \to w \text{ , is left child of node } j \\ 0, & \text{if node } j+1 \text{ at path , root} \to w \text{ , is left child of node } j \end{cases}$$

3. 定义：

$$e_{n(w,j)} = \frac{\partial E}{\partial (\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}})} = \sigma(\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}}) - t_{n(w,j)}$$

   - $\sigma(\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}})$ 为预测选择 j 的左子节点的概率。
   - $e_{n(w,j)}$ 的物理意义为：从根节点到单词 w 的路径上，第 j 个节点的选择误差：
     - 如果下一个节点选择第 j 个节点的左子节点，则 $t_{n(w,j)} = 1$。 此时 $e_{n(w,j)}$ 表示预测的不足
     - 如果下一个节点选择第 j 个节点的右子节点，则 $t_{n(w,j)} = 0$。 此时 $e_{n(w,j)}$ 表示预测的过量

4. 考虑内部节点 $n(w,j)$，其向量表达为 $\vec{\mathbf{v}}'_{n(w,j)}$。则有：

$$\frac{\partial E}{\partial \vec{\mathbf{v}}'_{n(w,j)}} = \frac{\partial E}{\partial (\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}})} \times \frac{\partial (\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}})}{\partial \vec{\mathbf{v}}'_{n(w,j)}} = e_{n(w,j)} \times \vec{\mathbf{h}}$$

   得到向量表达为 $\vec{\mathbf{v}}'_{n(w,j)}$ 的更新方程：

$$\vec{\mathbf{v}}'^{(new)}_{n(w,j)} = \vec{\mathbf{v}}'^{(old)}_{n(w,j)} - \eta \times e_{n(w,j)} \times \vec{\mathbf{h}}; \quad j = 1, 2, \cdots, L(w) - 1$$

   - 对于每个单词 w ，由于它是叶节点，因此可以更新 $L(w) - 1$ 个内部节点的向量表达。
   - 当模型的预测概率较准确，即 $\sigma(\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}}) \simeq t_{n(w,j)}$ 时，则 $e_{n(w,j)}$ 接近0 。此时梯度非常小，$\vec{\mathbf{v}}'_{n(w,j)}$ 的更新幅度也会非常小。

     当模型的预测概率较不准，则 $e_{n(w,j)}$ 较大。此时梯度会较大，$\vec{\mathbf{v}}'_{n(w,j)}$ 的更新幅度也会较大。

5. 对于内部结点的向量表达 $\vec{\mathbf{v}}'_{n(w,j)}$ 的更新方程适用于 `CBOW` 模型和 `Skip-Gram` 模型。但是在 `Skip-Gram` 模型中，需要对 C 个输出的每一个单词进行更新。

c) CBOW 输入向量参数更新

1. 对于 `CBOW` 模型，定义：

$$\overrightarrow{\mathbf{EH}} = \frac{\partial E}{\partial \vec{\mathbf{h}}} = \sum_{j=1}^{L(w)-1} \frac{\partial E}{\partial (\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}})} \times \frac{\partial (\vec{\mathbf{v}}'_{n(w,j)} \cdot \vec{\mathbf{h}})}{\partial \vec{\mathbf{h}}} = \sum_{j=1}^{L(w)-1} e_{n(w,j)} \vec{\mathbf{v}}'_{n(w,j)}$$

   $\overrightarrow{\mathbf{EH}}$ 的物理意义为：二叉树中所有内部节点向量表达的加权和，其权重为 $e_{n(w,j)}$ 。

2. 考虑到 $\vec{\mathbf{h}} = \frac{1}{C} \mathbf{W}^T (\vec{\mathbf{x}}_1 + \vec{\mathbf{x}}_2 + \cdots + \vec{\mathbf{x}}_C)$ ，则有：

$$\frac{\partial E}{\partial w_{k,i}} = \frac{\partial E}{\partial h_i} \times \frac{\partial h_i}{\partial w_{k,i}} = \frac{1}{C} EH_i \times (x_{(1,k)} + x_{(2,k)} + \cdots, +x_{(C,k)})$$

写成矩阵的形式为：$\frac{\partial E}{\partial \mathbf{W}}=\frac{1}{C}\sum_{c=1}^{C}\vec{\mathbf{x}}_c\otimes\overrightarrow{\mathbf{EH}}$，其中 $\otimes$ 为克罗内克积。

将 $\mathbf{W}$ 的更新分解为 C 次，每次对应于一个输入 $\vec{\mathbf{x}}_c$ 。因此得到 $\mathbf{W}$ 的更新方程：

$$\vec{\mathbf{w}}_{I_i}^{(new)}=\vec{\mathbf{w}}_{I_i}^{(old)}-\frac{1}{C}\eta\overrightarrow{\mathbf{EH}},\quad i=1,2,\cdots,C$$

其中I_i 为第 i 个输入单词在词表 $\mathbb{V}$ 中的编号。

d) Skip-Gram 输入向量参数更新

1. 对于 `Skip-Gram` 模型，定义：

$$\begin{aligned}\overrightarrow{\mathbf{EH}}=\frac{\partial E}{\partial\vec{\mathbf{h}}}&=\sum_{c=1}^{C}\sum_{j=1}^{L(w_c)-1}\frac{\partial E}{\partial(\vec{\mathbf{v}}_{n(w_c,j)}\cdot\vec{\mathbf{h}})}\times\frac{\partial(\vec{\mathbf{v}}_{n(w_c,j)}\cdot\vec{\mathbf{h}})}{\partial\vec{\mathbf{h}}}\\&=\sum_{c=1}^{C}\sum_{j=1}^{L(w_c)-1}e_{n(w_c,j)}\times\vec{\mathbf{v}}_{n(w_c,j)}\end{aligned}$$

   其中：w_c 表示网络第 c 个输出的输出单词。

2. 注意：由于引入分层`softmax`，导致更新路径因为输出单词的不同而不同。

   因此 $\sum_{j=1}^{L(w_c)-1}$ 会因为 c 的不同而不同。

   因此 $\sum_{c=1}^{C}$ 和 $\sum_{j=1}^{L(w_c)-1}$ 无法互换。

3. 与`Skip-Gram` 中推导相同， $\mathbf{W}$ 的更新方程为：

$$\vec{\mathbf{w}}_{I}^{(new)}=\vec{\mathbf{w}}_{I}^{(old)}-\eta\overrightarrow{\mathbf{EH}}$$

   其中 $\vec{\mathbf{w}}_I$ 为 $\vec{\mathbf{x}}$ 非零分量对应的 $\mathbf{W}$ 中的行，而 $\mathbf{W}$ 的其它行在本次更新中都保持不变。

3.3.2 负采样

a) 原理

1. 在网络的输出层，真实的输出单词对应的输出单元毋庸置疑必须输出（它作为正向单元）。其它所有单词对应的输出单元为负向单元。
   - 如果计算所有负向单元的输出概率，则计算量非常庞大。
   - 可以从所有负向单元中随机采样一批负向单元，仅仅利用这批负向单元来更新。这称作负采样。
2. 负采样的核心思想是：在参数的每一轮更新中，实际上只需要用到一部分单词的输出概率；大部分单词的输出概率为 0 。
3. 负向单元采样的概率分布称作`noise` 分布，记做 $P_n(w)$
   - 它可以为任意的概率分布（通常需要根据经验来选择）
   - 谷歌给出的建议是挑选 5~10 个负向单元，根据下面公式来采样：

$$P_n(w)=\frac{freq(w)^{3/4}}{\sum_{w\ne w_O}freq(w)^{3/4}}$$

     其中：
     - $freq(w)$ 为单词在语料库中出现的概率。
     - 分母对所有的负样本单词累加。

- 其背后的物理意义为：单词在语料库中出现的概率越大，则越可能被挑中。

b) 输出向量参数更新

1. 假设输出的单词分类两类：
   - 正类：只有一个，即真实的输出单词 w_O
   - 负类：从 $P_n(w)$ 采样得到的 K 个单词 $\mathcal{W}_{neg}=\{w_1,w_2,\cdots,w_K\}$

   论文作者指出：下面的训练目标能够得到更好的结果：

$$E=-\log\sigma(\vec{\mathbf{w}}'_{w_O}\cdot\vec{\mathbf{h}})-\sum_{w_j\in\mathcal{W}_{neg}}\log\sigma(-\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})$$

   其中：
   - $\vec{\mathbf{w}}'_{w_O}$ 为真实的输出单词对应的输出向量
   - $\vec{\mathbf{w}}'_{w_j}$ 为负采样的单词得到的输出向量。
2. 负采样的目标函数是一个经验公式，而不是理论上的后验概率 $P(w_O\mid w_I)$ 的负对数似然：

   $-\log\sigma(\vec{\mathbf{w}}'_{w_O}\cdot\vec{\mathbf{h}})$ 。
   - $\sigma(\vec{\mathbf{w}}'_{w_O}\cdot\vec{\mathbf{h}})$ ：在单词 w_O 上输出为正类的概率
   - $\sigma(-\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})$：在单词 w_j 上输出为负类的概率

   其物理意义为：在正类单词上取正类、负类单词上取负类的负对数似然。
3. 根据 E 的定义，有：

$$\frac{\partial E}{\partial(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})}=\begin{cases}\sigma(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})-1, & \text{if } w_j=w_O\\ \sigma(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}}), & \text{if } w_j\in\mathcal{W}_{neg}\end{cases}$$
$$=\sigma(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})-t_j$$

   其中 t_j 标记了单词 w_j 的标签：

$$t_j=\begin{cases}1, & \text{if } w_j=w_O\\ 0, & \text{esle}\end{cases}$$

4. 令 $e_{w_j}=\sigma(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})-t_j$ ，它刻画了网络在正类单词和负类单词上的预测误差。
   - 当 $w_j=w_O$ 时，e_{w_j} 表示对正类单词预测概率的不足。
   - 当 $w_j\in\mathcal{W}_{neg}$ 时，e_{w_j} 表示对负类单词预测概率的过量。
5. 根据：

$$\frac{\partial E}{\partial\vec{\mathbf{w}}'_{w_j}}=\frac{\partial E}{\partial(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})}\times\frac{\partial(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})}{\partial\vec{\mathbf{w}}'_{w_j}}=e_{w_j}\times\vec{\mathbf{h}}$$

   则有输出向量的更新方程：

$$\vec{\mathbf{w}}'^{(new)}_{w_j}=\vec{\mathbf{w}}'^{(old)}_{w_j}-\eta\times e_{w_j}\times\vec{\mathbf{h}}$$

6. 给定一个样本，在更新输出向量时，只有 $K+1$ 个输出向量（ 1 个输出单词 w_O 、K 个负采样单词对应的输出向量）得到更新，其中 K 通常数量很小。其它所有单词对应的输出向量未能得到更新。

   相比较而言：
   - 原始算法中，给定一个样本，在更新输出向量时，所有输出向量（一共 V 个）都得到更新

- 分层 `softmax` 算法中，给定一个样本，在更新输出向量时，$L(w)-1$ 个内部节点的向量表达得到更新。

7. 输出向量的更新方程可以用于 `CBOW` 模型和 `Skip-Gram` 模型。

   若用于 `Skip-Gram` 模型，则对每个输出依次执行输出向量的更新。

c) CBOW 输入向量参数更新

1. 对于 `CBOW` 模型，定义：

$$\overrightarrow{\mathbf{EH}}=\frac{\partial E}{\partial \vec{\mathbf{h}}}=\sum_{w_j\in\{w_O\}\bigcup\mathcal{W}_{neg}}\frac{\partial E}{\partial(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})}\times\frac{\partial(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})}{\partial\vec{\mathbf{h}}}=\sum_{w_j\in\{w_O\}\bigcup\mathcal{W}_{neg}}e_{w_j}\times\vec{\mathbf{w}}'_{w_j}$$

   $\overrightarrow{\mathbf{EH}}$ 的物理意义为：负采样单词、输出单词对应输出向量的加权和，其权重为 e_{w_j} 。

2. 与 `分层softmax: CBOW 输入向量参数更新` 中的推导相同，$\mathbf{W}$ 的更新方程为：

$$\vec{\mathbf{w}}_{I_i}^{(new)}=\vec{\mathbf{w}}_{I_i}^{(old)}-\frac{1}{C}\eta\overrightarrow{\mathbf{EH}},\quad i=1,2,\cdots,C$$

   其中I_i 为第 i 个输入单词在词表 $\mathbb{V}$ 中的编号。

d) Skip-Gram 输入向量参数更新

1. 对于 `CBOW` 模型，定义：

$$\begin{aligned}\overrightarrow{\mathbf{EH}}=\frac{\partial E}{\partial\vec{\mathbf{h}}}&=\sum_{c=1}^{C}\sum_{w_j\in\{w_O^c\}\bigcup\mathcal{W}_{neg}^c}\frac{\partial E}{\partial(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})}\times\frac{\partial(\vec{\mathbf{w}}'_{w_j}\cdot\vec{\mathbf{h}})}{\partial\vec{\mathbf{h}}}\\&=\sum_{c=1}^{C}\sum_{w_j\in\{w_O^c\}\bigcup\mathcal{W}_{neg}^c}e_{w_j}\times\vec{\mathbf{w}}'_{w_j}\end{aligned}$$

   其中：

   - w_O^c 表示网络第 c 个输出的输出单词。
   - $\mathcal{W}_{neg}^c$ 表示网络第 c 个输出的负采样单词集。

2. 注意：由于引入负采样，导致网络每个输出中，对应的输出单词有所不同，负采样单词也有所不同。

   因此 $\{w_O^c\}\bigcup\mathcal{W}_{neg}^c$ 会因为 c 的不同而不同。

   因此 $\sum_{c=1}^C$ 和 $\sum_{w_j\in\{w_O^c\}\bigcup\mathcal{W}_{neg}^c}$ 无法互换。

3. 与 `Skip-Gram` 中推导相同，$\mathbf{W}$ 的更新方程为：

$$\vec{\mathbf{w}}_I^{(new)}=\vec{\mathbf{w}}_I^{(old)}-\eta\overrightarrow{\mathbf{EH}}$$

   其中 $\vec{\mathbf{w}}_I$ 为 $\vec{\mathbf{x}}$ 非零分量对应的 $\mathbf{W}$ 中的行，而 $\mathbf{W}$ 的其它行在本次更新中都保持不变。

3.4 使用

1. 模型、语料库、超参数这三个方面都会影响词向量的训练，其中语料库对训练结果的好坏影响最大。

3.4.1 模型选择

1. 模型方面，所有的词向量都是基于分布式分布假说：拥有相似上下文的单词，其词义相似。

2. 根据目标词和上下文的关系，模型可以分为两类：
   - 通过上下文来预测目标词。

     这类模型更能够捕获单词之间的可替代关系。
   - 通过目标词来预测上下文。
3. 通过实验发现：简单的模型（`Skip-Gram`）在小语料库下表现较好。复杂的模型在大语料库下略有优势。

3.4.2 语料库

1. 实际上语料库并不是越大越好，语料库的领域更重要。
   - 选择了合适的领域，可能只需要 `1/10` 甚至 `1/100` 的语料就能够得到一个大的、泛领域语料库的效果。
   - 如果选择不合适的领域，甚至会导致负面效果，比随机词向量效果还差。

3.4.3 超参数

1. 词向量的维度：
   - 做词向量语义分析任务时，一般维度越大，效果越好
   - 做具体`NLP` 任务时（用作输入特征、或者网络初始化），`50` 维之后效果提升就比较少了。
2. 迭代次数：由于训练词向量的目标是尽可能精确地预测目标词，这个优化目标和实际任务并不一致。

   因此最好的做法是：直接用实际任务的验证集来做终止条件。

   如果实际任务非常耗时，则可以随机挑选某个任务（如：情感分类）及其验证集来做终止条件。

四、GloVe

1. 学习词向量的所有无监督方法最终都是基于语料库的单词共现统计，因此这些模型之间存在共性。
2. 词向量学习算法有两个主要的模型族：
   - 基于全局矩阵分解的方法，如：`latent semantic analysis:LSA` 。
     - 优点：能够有效的利用全局的统计信息。
     - 缺点：在单词类比任务（如：`国王 vs 王后` 类比于`男人 vs 女人`）中表现相对较差。
   - 基于局部上下文窗口的方法，如：`word2vec`。
     - 优点：在 单词类比任务中表现较好。
     - 缺点：因为`word2vec` 在独立的局部上下文窗口上训练，因此难以利用单词的全局统计信息。
3. `Global Vectors for Word Representation:GloVe` 结合了`LSA` 算法和`Word2Vec` 算法的优点，既考虑了全局统计信息，又利用了局部上下文。

4.1 原理

1. 设单词`单词-单词` 共现矩阵为 $\mathbf{X}$ 。其中 $X_{i,j}$ 表示在整个语料库中，单词 word_j 在单词 word_i 上下文中出现的次数。 令：
   - $X_i=\sum_{k=1}^V X_{i,k}$ ，它表示：单词 word_i 上下文中出现的所有单词的总数。
   - $P_{i,j}=P(\text{word}_j\mid\text{word}_i)=\frac{X_{i,j}}{X_i}$ ，它表示：单词 word_j 出现在单词 word_i 的上下文中的概率。
   - $Ratio_{i,j}^k=\frac{P_{i,k}}{P_{j,k}}$ ，它表示：单词 word_k 出现在单词 word_i 的上下文中的概率，相对于单词 word_k 出现在单词 word_j 的上下文中的概率的比值。
2. 从经验中可以发现以下规律：

	单词 word_k 和单词 word_i 相关	**单词 word_k 和单词 word_i 不相关**
单词 word_k 和单词 word_j 相关	$Ratio_{i,j}^k$ 趋近于 1	$Ratio_{i,j}^k$ 比较小
单词 word_k 和单词 word_j 不相关	$Ratio_{i,j}^k$ 比较大	$Ratio_{i,j}^k$ 趋近于 1

因此 $Ratio_{i,j}^k$ 能够反映单词之间的相关性。

3. 假设单词 $\text{word}_i,\text{word}_j,\text{word}_k$ 的词向量分别为 $\vec{\mathbf{w}}_i,\vec{\mathbf{w}}_j,\vec{\mathbf{w}}_k$ 。

   `GloVe` 认为：这三个单词的词向量经过某个函数的映射之后等于 $Ratio_{i,j}^k$ 。即：词向量中包含了共现矩阵的信息。

   假设这个映射函数为 F ，则有：

   $$F(\vec{\mathbf{w}}_i,\vec{\mathbf{w}}_j,\vec{\mathbf{w}}_k)=Ratio_{i,j}^k=\frac{P_{i,k}}{P_{j,k}}$$

   现在的问题是 $F(\cdot)$ 未知，词向量 $\vec{\mathbf{w}}_i,\vec{\mathbf{w}}_j,\vec{\mathbf{w}}_k$ 也是未知。如果能够确定 $F(\cdot)$ ，则可以求解词向量。

4. 由于 $F(\cdot)$ 映射的是向量空间，而向量空间是一个线性空间。因此从右侧的除法 $\frac{P_{i,k}}{P_{j,k}}$ 可以联想到对 $\vec{\mathbf{w}}_i$ 和 $\vec{\mathbf{w}}_j$ 做减法。即 $F(\cdot)$ 的形式为：

   $$F(\vec{\mathbf{w}}_i-\vec{\mathbf{w}}_j,\vec{\mathbf{w}}_k)=\frac{P_{i,k}}{P_{j,k}}$$

   由于 $\vec{\mathbf{w}}_i-\vec{\mathbf{w}}_j$ 和 $\vec{\mathbf{w}}_k$ 均为向量，而 $\frac{P_{i,k}}{P_{j,k}}$ 为标量。因此可以联想到向量的内积。即 $F(\cdot)$ 的形式为：

   $$F\left((\vec{\mathbf{w}}_i-\vec{\mathbf{w}}_j)^T\vec{\mathbf{w}}_k\right)=F\left(\vec{\mathbf{w}}_i^T\vec{\mathbf{w}}_k-\vec{\mathbf{w}}_j^T\vec{\mathbf{w}}_k\right)=\frac{P_{i,k}}{P_{j,k}}$$

   上式左边为差的形式，右边为商的形式。因此联想到函数 $\exp(\cdot)$ 。即 $F(\cdot)$ 的形式为：

   $$F(\cdot)=\exp(\cdot)$$
   $$\vec{\mathbf{w}}_i^T\vec{\mathbf{w}}_k-\vec{\mathbf{w}}_j^T\vec{\mathbf{w}}_k=\log P_{i,k}-\log P_{j,k}$$

   要想使得上式成立，只需要令 $\vec{\mathbf{w}}_i^T\vec{\mathbf{w}}_k=\log P_{i,k},\quad \vec{\mathbf{w}}_j^T\vec{\mathbf{w}}_k=\log P_{j,k}$ 即可。

5. 向量的内积具有对称性，即 $\vec{\mathbf{w}}_i^T\vec{\mathbf{w}}_k=\vec{\mathbf{w}}_k^T\vec{\mathbf{w}}_i$ 。而 $\log X_{i,k}-\log X_i\ne\log X_{k,i}-\log X_k$ ，即：$\log P_{i,k}\ne\log P_{k,i}$ 。

   为了解决这个问题，模型引入两个偏置项：

   $$\log X_{i,k}=\vec{\mathbf{w}}_i^T\vec{\mathbf{w}}_k+b_i+b_k$$

6. 上面的公式仅仅是理想状态，实际上只能要求左右两边尽可能相等。于是设计代价函数为：

   $$J=\sum_{i,k}\left(\vec{\mathbf{w}}_i^T\vec{\mathbf{w}}_k+b_i+b_k-\log X_{i,k}\right)^2$$

4.2 权重函数

1. 根据经验，如果两个词共现的次数越多，则这两个词在代价函数中的影响就应该越大。因此可以设计一个权重来对代价函数中的每一项进行加权，权重为共现次数的函数：

$$J=\sum_{i,k} f(X_{i,k})\left(\vec{\mathbf{w}}_i^T\vec{\mathbf{w}}_k+b_i+b_k-\log X_{i,k}\right)^2$$

   其中权重函数应该符合三个条件：

   - $f(0)=0$ 。即：如果两个词没有共现过，则权重为 0 。

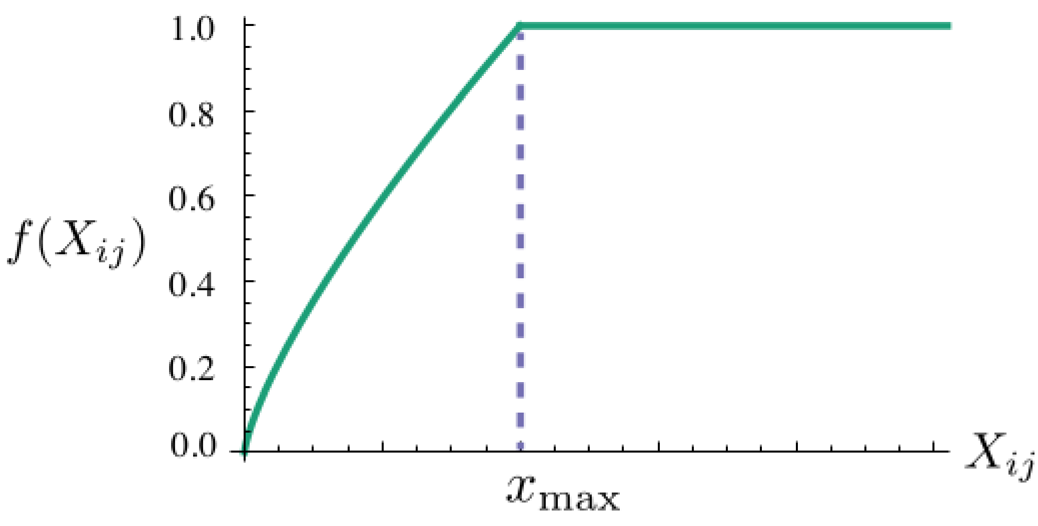
     这是为了确保 $\lim_{x\to 0} f(x)\log^2 x$ 是有限值。
   - $f(\cdot)$ 是非递减的。即：两个词共现次数越大，则权重越大。
   - $f(\cdot)$ 对于较大的 $X_{i,k}$ 不能取太大的值。即：有些单词共现次数非常大（如单词 `的` 与其它词的组合），但是它们的重要性并不是很大。

2. `GloVe` 论文给出的权重函数 $f(\cdot)$ 为：

$$f(x)=\begin{cases}\left(\frac{x}{x_{\max}}\right)^\alpha & \text{if} \quad x<x_{\max}\\ 1, & \text{otherwise}\end{cases}$$

   其中：

   - `GloVe` 论文给出参数 α 和 $x_{\max}$ 的经验值为：$\alpha=\frac{3}{4}, x_{\max}=100$ 。
   - `GloVe` 论文指出： $x_{\max}$ 对模型的性能影响较小。



4.3 性能

1. `GloVe` 模型的算法复杂度取决于共现矩阵 $\mathbf{X}$ 中的非零元素的个数，最坏的情况下为 $O(V^2)$ 。由于词汇表的数量通常很庞大，因此 V^2 会非常大。

   实际上单词共现的次数满足齐普夫定律(`Zipf's Law`)，因此算法复杂度较低，约为 $O(|C|)$ ，其中 C 为语料库的大小。

   > `Zipf's Law`：如果有一个包含 n 个词的文章，将这些词按其出现的频次递减地排序，那么序号 r 和其出现频次 f 之积 $f \times r$，将近似地为一个常数，即 $f \times r = const$

2. `GloVe` 模型评估任务：
   - `semantic` 任务： 语义任务。如：`'雅典'之于'希腊' = '柏林'之于'_'?`
   - `syntactic` 任务：语法任务。如：`'dance'之于'dancing' = 'fly'之于'_'?`
3. `GloVe` 模型性能与语料库大小的关系：
   - 在语法任务中，模型性能随着语料库大小的增长而单调增长。

     这是因为语料库越大，则语法的统计结果越可靠。
   - 在语义任务中，模型性能与语料库绝对大小无关，而与语料库的有效大小有关。

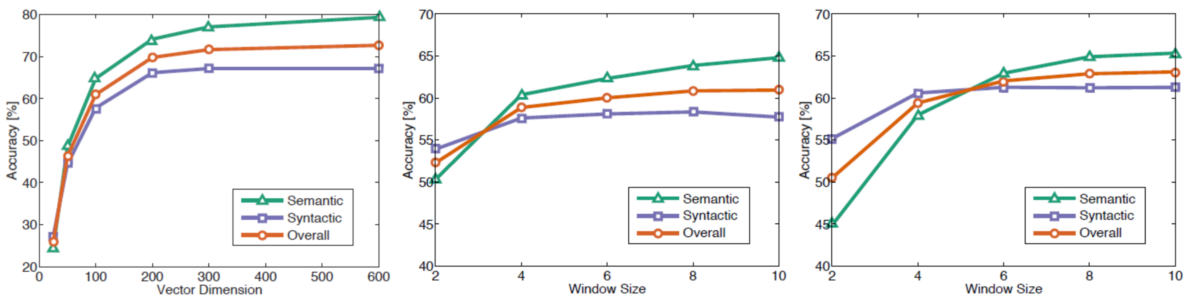
     有效大小指的是语料库中，与目标语义相关的内容的大小。
4. `GloVe` 模型超参数选择：
   - 词向量大小：词向量大小越大，则模型性能越好。但是词向量超过 `200` 维时，维度增加的收益是递减的。
   - 窗口对称性：计算一个单词的上下文时，上下文窗口可以是对称的，也可以是非对称的。
     - 对称窗口：既考虑单词左侧的上下文，又考虑单词右侧的上下文
     - 非对称窗口：只考虑单词左侧的上下文。

       > 因为语言的阅读习惯是从左到右，所以只考虑左侧的上下文，不考虑右侧的上下文。
   - 窗口大小：
     - 在语法任务中，选择小的、非对称的窗口时，模型性能更好。

       因为语法是局部的，所以小窗口即可。

       因为语法是依赖于单词顺序的，所以需要非对称窗口。
     - 对于语义任务，则需要选择更大的窗口。

       因为语义是非局部的。



(a) Symmetric context (b) Symmetric context (c) Asymmetric context