

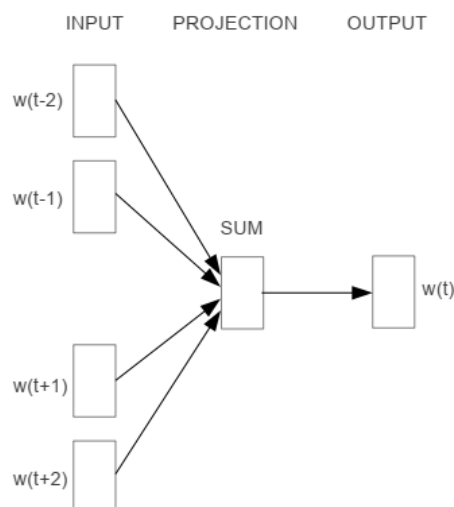
# Word2Vec

Word2Vec是Tomas Mikolov于2013年提出的文本向量化方法，可以在大量文本语料上通过无监督训练学到词的分布式向量表示。该方法提出CBOW、Skip-Gram两种模型架构，二者都基于窗口内的词共现信息来训练模型，还提出Hierarchical Softmax、Negative Sampling两种优化算法来提高训练效率。和之前的词向量学习模型相比，Word2Vec是一个划时代的作品。

关键词：词表征, 词向量

## 1 CBOW

CBOW（Continuous Bag-of-Words）根据窗口内的上下文词来预测中心词，其模型结构如下图所示，从左往右依次是输入层、映射层和输出层。



1. 输入层：输入one-hot形式的上下文词  $(w_{t-c}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+c})$ ，其中  $w_t$  表示中心词， $c$  为窗口尺寸；
2. 映射层：映射层的参数矩阵  $\mathcal{V} \in \mathbb{R}^{d \times |V|}$  包含了各个词对应的input embedding，其中d为词向量的维度， $|V|$  表示词典的大小。通过映射可以得到各个上下文词的词向量  $(v_{t-c}, \dots, v_{t-1}, v_{t+1}, \dots, v_{t+c})$ ，求均值：

$$\hat{v} = \frac{v_{t-c} + \dots + v_{t-1} + v_{t+1} + \dots + v_{t+c}}{2c}$$

3. 输出层：输出层的参数矩阵  $\mathcal{U} \in \mathbb{R}^{|V| \times d}$  包含了各个词对应的output embedding，通过相乘可以得到每个词的分值，然后对输出进行softmax计算：

$$\hat{y} = \text{softmax}(\mathcal{U}\hat{v})$$

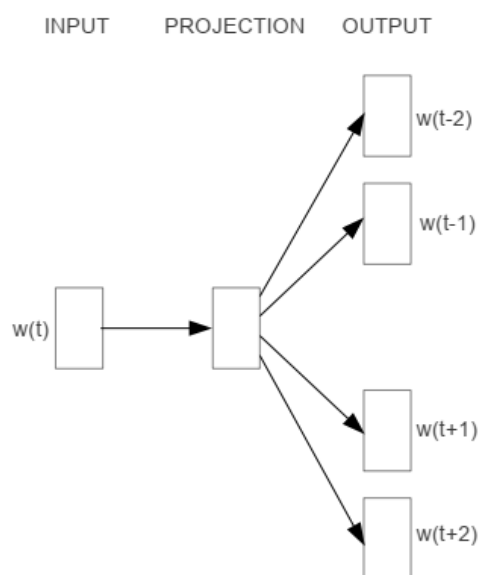
目标函数为：

$$\begin{aligned}
\text{minimize } J &= -\log P(w_t | w_{t-c}, \dots, w_{t-1}, w_{t+1}, w_{t+c}) \\
&= -\log P(u_t | \hat{v}) \\
&= -\log \frac{\exp(u_t^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\
&= u_t^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})
\end{aligned}$$

CBOW采用随机梯度下降来对参数进行优化。

## 2 Skip-Gram

和CBOW相反，Skip-Gram是根据中心词来预测上下文词，其模型架构如下图所示：



模型同样是三层结构：

1. 输入层：输入中心词  $w_t$  ；
2. 映射层：通过  $\mathcal{V}$  得到中心词的input embedding  $v_t$  ；
3. 输出层：通过  $\mathcal{U}$  得到分数向量，然后进行归一化处理：

$$\hat{y} = \text{softmax}(\mathcal{U}v_t)$$

由于Skip-Gram预测的是上下文词，所以输出不止一个。Skip-Gram假设给定中心词的前提下，各个上下文词的预测彼此独立，于是目标函数可表示为：

$$\begin{aligned}
\text{minimize } J &= -\log p(w_{t-c}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+c}) \\
&= -\log \prod_{0 < |i| \leq c} p(w_{t+i} | w_t) \\
&= -\log \prod_{0 < |i| \leq c} p(u_{t+i} | v_t) \\
&= -\log \prod_{0 < |i| \leq c} \frac{\exp(u_{t+i}^T)}{\sum_{j=1}^{|V|} \exp(u_j^T v_t)} \\
&= -\sum_{0 < |i| \leq c} u_{t+i}^T v_t + 2m \log \sum_{j=1}^{|V|} \exp(u_j^T v_t)
\end{aligned}$$

同样Skip-Gram也是采用随机梯度下降来进行参数估计。

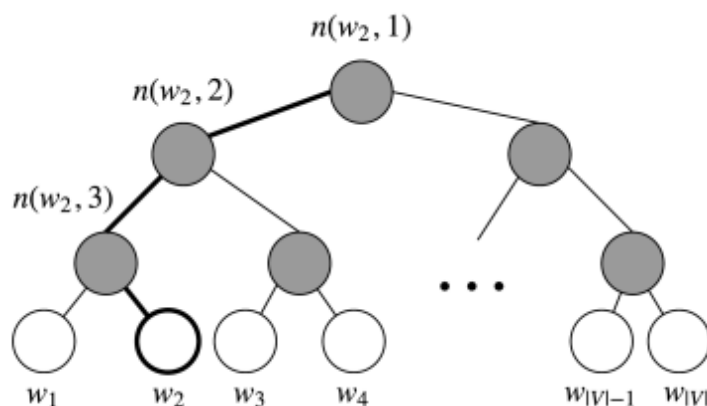
📌 Skip-Gram输出层只包含一个参数矩阵  $\mathcal{U}$ ，在训练时，Word2Vec根据中心词预测各个上下词的过程是互相独立的，也就是会分开计算各个正确上下文词对应的损失值来进行反向传播训练。但是，Word2Vec模型不需要测试，输入嵌入对应的矩阵参数  $\mathcal{V}$  就是训练得到的词向量，直接基于此计算词向量的性能度量即可。

CBOW和Skip-Gram的计算量都很大，因为要计算一个词典大小的分数向量，然后再进行softmax归一化。当词典很大的时候，分数几乎都接近0。所以为了提高计算效率，Word2Vec提出了下面两种优化算法。

### 3 Hierarchical Softmax

Hierarchical Softmax借助哈夫曼树（Huffman Tree）来提高解码的效率，它把词典中所有的词根据词频排成哈夫曼树，使得高频词靠近树根，低频词远离树根。这样依次计算各个词的分数过程变成从树根到任意叶子结点的行走过程，计算次数可从  $|V|$  降至  $\log_2 |V|$ （树的高度）。

以下图为例，每个叶子结点对应一个词，每个内部结点（非叶子结点）都包含一个向量。



假设

- $L(w)$  是从根结点到  $w$  叶子结点路径中所有内部结点的数量，如  $L(w_2) = 3$ ；

- $n(w, i)$  表示该路径上的第  $i$  个结点，对应的向量为  $v_{n(w, i)}$ ，如  $n(w, 1) = root$ ， $n(w, L(w))$  是  $w$  的父结点；
- 对于任意一个内部结点， $ch(n)$  表示  $n$  的固定方向的子结点（要么一直为左子结点，要么一直为右子结点）。

于是采用Hierarchical Softmax的概率计算公式为：

$$P(w|w_i) = \prod_j^{L(w)-1} \sigma([n(w, j+1) = ch(n(w, j))]) \cdot v_{n(w, j)}^T v_{w_i} \quad (*)$$

其中

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

因为有了  $[x]$ ，使得任意一个内部结点上，左右的概率和为1：

$$\sigma(v_n^T v_{w_i}) + \sigma(-v_n^T v_{w_i}) = 1$$

并且也不难证明  $\sum_{w=1}^{|V|} P(w|w_i) = 1$ 。

假如在行走过程中的方向选择分别为左、左、右，那么  $(*)$  式具体为：

$$\begin{aligned} P(w_2|w_i) &= p(n(w_2, 1), left) \cdot p(n(w_2, 2), left) \cdot p(n(w_2, 3), right) \\ &= \sigma(v_{n(w_2, 1)}^T v_{w_i}) \cdot \sigma(v_{n(w_2, 2)}^T v_{w_i}) \cdot \sigma(v_{n(w_2, 3)}^T v_{w_i}) \end{aligned}$$

由此可见，原先  $|V|$  个output embedding变成了哈夫曼树所有内部结点的向量，共计  $\frac{1}{2}n(n-1)$  个。

## 4 Negative Sampling

和Hierarchical Softmax不同，Negative Sampling从训练的角度提出优化，对于每个正例数据  $(w, c)$ ， $c$  为中心词， $w$  为上下文词，只引入部分负样例数据来进行对比学习，使得模型具备区分二者的能力。

假设  $(w, c)$  为正、负样例的概率分别为：

$$\begin{aligned} p(y = 1|w, c, \theta) &= \sigma(v_c^T u_w) \\ p(y = 0|w, c, \theta) &= 1 - \sigma(v_c^T u_w) \\ &= \sigma(-v_c^T u_w) \end{aligned}$$

则根据极大似然估计得到损失函数为：

$$\mathcal{L} = \prod_{(w, c) \in \mathcal{D}} p(y = 1|w, c, \theta) \prod_{(w, c) \in \overline{\mathcal{D}}} p(y = 0|w, c, \theta)$$

目标函数为似然函数的负对数形式：


$$\text{minimize } J = - \sum_{(w,c) \in D} \ln \sigma(v_c^T u_w) - \sum_{(w,c) \in \bar{D}} \ln \sigma(-v_c^T u_w)$$

这里的损失函数分两项，对于正例数据，损失值为前一项；对于负例数据，损失值为后一项。

论文2中给出了negative sampling的另一种表达形式：

$$\text{maximize } \log \sigma(v'_{w_O} v_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} [\log \sigma(-v'_{w_i} v_{w_I})]$$

其中  $w_I, w_O$  分别表示输入词和输出词， $v', v$  分别表示output embedding和input embedding。该式子对负样本表达地更加清晰，即负样本服从一个噪声分布  $P(n)$ 。

 原文对于噪声分布的介绍很少，故不展开。

## 高频词下采样

训练集中每个词  $w_i$  被丢弃的概率为：

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

其中  $t$  表示阈值，一般取  $10^{-5}$  左右；该式子使得模型倾向于丢弃频率大于  $t$  的高频词，因为高频词往往是一些无义词。

## 5 深度思考

### 关于input embedding和output embedding的理解

input embedding表示词的词向量，它刻画的是词本身的语义；output embedding表示词的上下文向量，它刻画的是词的上下文语义（或者说是所有上下文词的整体语义）。分析如下：

假设  $a, b$  为一组同义词， $c, d$  为一组同义词，这两组同义词**彼此不同义但在句子中共现的频率非常高**。

如果用  $I_a$  表示  $a$  的input embedding， $O_a$  表示  $a$  的output embedding，经过word2vec训练后，由于同义关系的存在，我们会得到：

$$I_a \approx I_b, \quad I_c \approx I_d$$

同时由于共现关系的存在，我们会得到：

$$I_a \approx O_c, \quad I_a \approx O_d$$

那么结合以上两式，可以得到：

$$\begin{cases} I_a \approx I_b \approx O_c \approx O_d \\ I_c \approx I_d \approx O_a \approx O_b \end{cases}$$

此时，如果尝试以这个结论去反推模型设计的初衷，有如下思考：

- 一个词的input embedding会和所有同义词的input embedding相似，为了提高它的质量，作者也是在word similarity的基础上提出了一个更严格的word analogy任务去评测，所以作者设计input embedding就是为了刻画词本身的语义，而作者本人在论文中也承认了这点；
- 一个词的output embedding会和它所有高频的上下文词的input embedding相似，那么可以大胆推测，output embedding刻画的是词的上下文语义，但是很可惜在论文中并没有找到这个说法，但在tensorflow的官方教程上看到了类似的理解；

正是基于这个理解，我认为这两个矩阵不能一样，即  $I_a \approx O_a$  会带来什么，会带来  $I_a \approx I_c$ ，也就是满足上下文关系的词也会变得同义。

🧐 延展：基于这个理解，也可以思考self-attention中的q,k和multihead-attention中的q,k,v.

## 优缺点

word2vec的优点是：

- 词向量是低维稠密的，解决了维度灾难；

缺点是：

- 无法解决一词多义的问题；

## 实现

word2vec源码是c语言，同时也没有用到gpu，比较流行的做法是采用gensim.

## 参考文献

1. Linguistic Regularities in Continuous Space Word Representations. Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. NAACL 2013.
2. [Efficient Estimation of Word Representations in Vector Space](#). Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. ICLR 2013.
3. Distributed Representations of Words and Phrases and their Compositionality. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. NIPS 2013.