

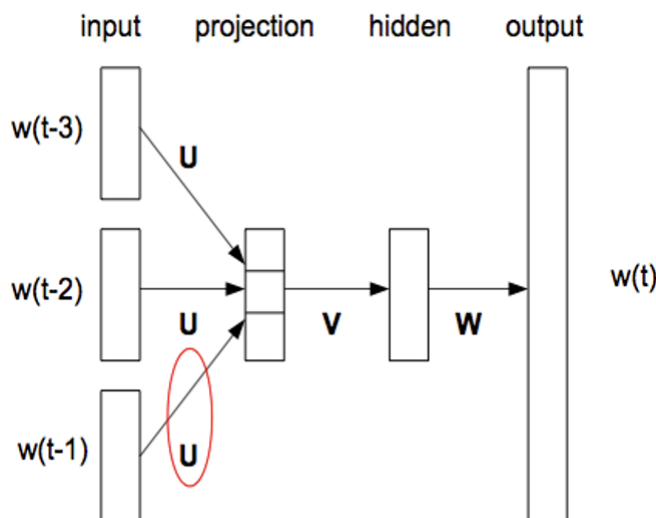
# 1 模型介绍

## 1.1 模型概述

Word2Vec是Google在2013年提出的一个NLP工具，它通过一个浅层的双层神经网络，高效率、高质量地将海量单词向量化。训练得到的词向量满足：

- 相似单词的词向量彼此接近。例如 $\text{dis}(\vec{V}(\text{man}), \vec{V}(\text{woman})) \ll \text{dis}(\vec{V}(\text{man}), \vec{V}(\text{computer}))$
- 保留单词间的线性规则性。例如 $\vec{V}(\text{king}) - \vec{V}(\text{man}) + \vec{V}(\text{woman}) \approx \vec{V}(\text{queen})$

Word2Vec模型的灵感来源于Bengio在2003年提出的NNLM模型（Neural Network Language Model），该模型使用一个三层前馈神经网络 $f(w_k, w_{k-1}, w_{k-2}, \dots, w_{k-n+1}; \theta)$ 来拟合一个词序列的条件概率 $P(w_k | w_{k-1}, w_{k-2}, \dots, w_1)$ 。第一层是映射层，通过一个共享矩阵，将One-Hot向量转化为词向量，第二层是一个激活函数为tanh的隐含层，第三层是Softmax输出层，将向量映射到 $[0, 1]$ 概率空间中。根据条件概率公式与大数定律，使用词频 $\frac{\text{Count}(w_k, w_{k-1}, w_{k-2}, \dots, w_{k-n+1})}{\text{Count}(w_{k-1}, w_{k-2}, \dots, w_{k-n+1})}$ 来近似地估计真实的条件概率。



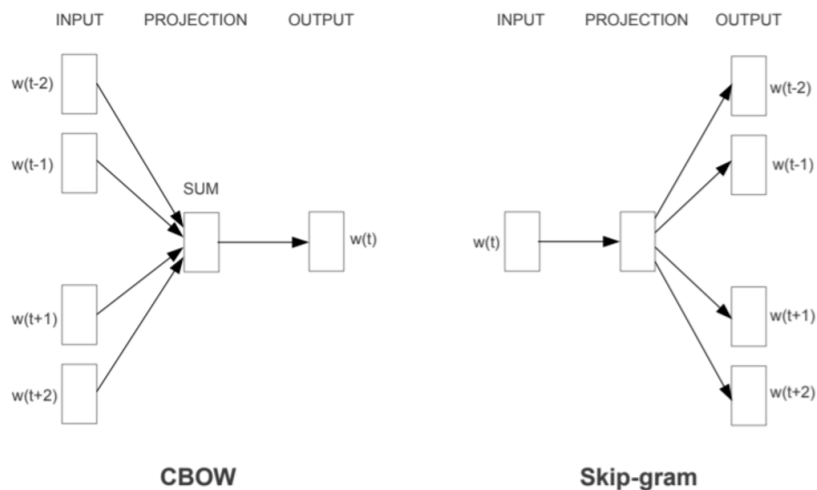
Bengio发现，我们可以使用映射层的权值作为词向量表征。但是，由于参数空间非常庞大，NNLM模型的训练速度非常慢，在百万级的数据集上需要耗时数周才能得到相对不错的结果，而在千万级甚至更大的数据集上，几乎无法得到结果。

Mikolov发现，NNLM模型可以被拆分成两个步骤：

- 用一个简单的模型训练出一个连续的词向量（映射层）
- 基于词向量表征，训练出一个N-Gram神经网络模型（隐含层+输出层）

而模型的计算瓶颈主要在第二步，特别是输出层的Sigmoid归一化部分。如果我们只是想得到词向量，可以对第二步的神经网络模型进行简化，从而提高模型的训练效率。因此，Mikolov对NNLM模型进行了以下几个部分的修改：

- 舍弃了隐含层。
- NNLM在利用上文词预测目标词时，对上文词的词向量进行了拼接，Word2Vec模型对其直接进行了求和，从而降低了隐含元的维度。
- NNLM在进行Sigmoid归一化时需要遍历整个词汇表，Word2Vec模型提出了Hierarchical Softmax与Negative Sampling两种策略进行优化。
- 依据分布式假设（上下文环境相似的两个词有着相近的语义），将下文单词也纳入训练环境，并提出了两种训练策略，一种是用上下文预测中心词，称为CBOW，另一种是用中心词预测上下文，称为Skip-Gram。



## 1.2 CBOW模型

假设我们的语料是"NLP is so interesting and challenging"。循环使用每个词作为中心词，来其上下文词来预测中心词。我们通常使用一个指定长度的窗口，根据马尔可夫性质，忽略窗口以外的单词。

中心词	上下文
NLP	is, so
is	NLP, so, interesting
so	NLP, is, interesting, and
interesting	is, so, and, challenging
and	so, interesting, challenging
challenging	interesting, and

我们的目标是通过上下文来预测中心词，也就是给定上下文词，出现该中心词的概率最大。这和完形填空颇有点异曲同工之妙。也即  $\max P(\text{NLP}|\text{is, so}) * P(\text{is}|\text{NLP, so, interesting}) * \dots$

用公式表示如下：

$$\begin{aligned}
 \max_{\theta} L(\theta) &= \prod_{w \in D} p(w|C(w)) \\
 &= \sum_{w \in D} \log p(w|C(w))
 \end{aligned}$$

其中  $w$  指中心词， $C(w)$  指上下文词集， $D$  指语料库，也即所有中心词的词集。

问题的核心变成了如何构造  $\log p(w|C(w))$ 。我们知道，NNLM模型的瓶颈在Sigmoid归一化上，Mikolov提出了两种改进思路来绕过Sigmoid归一化这一操作。一种思想是将输出改为一个霍夫曼树，每一个单词的概率用其路径上的权重乘积来表示，从而减少高频词的搜索时间；另一种思想是将预测每一个单词的概率，概率最高的单词是中心词改为预测该单词是不是正样本，通过负采样减少负样本数量，从而减少训练时间。

## 1.2.1 Hierarchical Softmax

### 1.2.2 Negative Sampling

基于Hierarchical Softmax的模型使用Huffman树代替了传统的线性神经网络，可以提高模型训练的效率。但是，如果训练样本的中心词是一个很生僻的词，那么在Huffman树中仍旧需要进行很复杂的搜索。负采样方法的核心思想是：设计一个分类器，对于我们需要预测的样本，设为正样本；而对于不是我们需要的样本，设置成负样本。在CBOW模型中，我们需要预测中心词 $w$ ，因此正样本只有 $w$ ，也即 $\text{Pos}(w) = \{w\}$ ，而负样本为除了 $w$ 之外的所有词。对负样本进行**随机采样**，得到 $\text{Neg}(w)$ ，大大简化了模型的计算。

我们首先将 $C(w)$ 输入映射层并求和得到隐含表征 $h_w = \sum_{u \in C(w)} \vec{v}(u)$

从而，

$$\begin{aligned} p(u|C(w)) &= \begin{cases} \sigma(h_w^T \theta_u), & \mathcal{D}(w, u) = 1 \\ 1 - \sigma(h_w^T \theta_u), & \mathcal{D}(w, u) = 0 \end{cases} \\ &= [\sigma(h_w^T \theta_u)]^{\mathcal{D}(w, u)} \cdot [1 - \sigma(h_w^T \theta_u)]^{1 - \mathcal{D}(w, u)} \end{aligned}$$

从而，

$$\begin{aligned} \max_{\theta} L(\theta) &= \sum_{w \in D} \log p(w|C(w)) \\ &= \sum_{w \in D} \log \prod_{u \in D} p(u|C(w)) \\ &\approx \sum_{w \in D} \log \prod_{u \in \text{Pos}(w) \cup \text{Neg}(w)} p(u|C(w)) \\ &= \sum_{w \in D} \log \prod_{u \in \text{Pos}(w) \cup \text{Neg}(w)} [\sigma(h_w^T \theta_u)]^{\mathcal{D}(w, u)} \cdot [1 - \sigma(h_w^T \theta_u)]^{1 - \mathcal{D}(w, u)} \\ &= \sum_{w \in D} \sum_{u \in \text{Pos}(w) \cup \text{Neg}(w)} \mathcal{D}(w, u) \cdot \log \sigma(h_w^T \theta_u) + [1 - \mathcal{D}(w, u)] \cdot \log[1 - \sigma(h_w^T \theta_u)] \\ &= \sum_{w \in D} \left\{ \sum_{u \in \text{Pos}(w)} \log \sigma(h_w^T \theta_u) + \sum_{u \in \text{Neg}(w)} \log[1 - \sigma(h_w^T \theta_u)] \right\} \end{aligned}$$

由于上式是一个最大化问题，因此使用随机梯度上升法对问题进行求解。

$$\text{令 } L(w, u, \theta) = \mathcal{D}(w, u) \cdot \log \sigma(h_w^T \theta_u) + [1 - \mathcal{D}(w, u)] \cdot \log[1 - \sigma(h_w^T \theta_u)]$$

$$\text{则 } \frac{\partial L}{\partial \theta_u} = \mathcal{D}(w, u) \cdot [1 - \sigma(h_w^T \theta_u)] h_w + [1 - \mathcal{D}(w, u)] \cdot \sigma(h_w^T \theta_u) h_w = [\mathcal{D}(w, u) - \sigma(h_w^T \theta_u)] h_w$$

因此 $\theta_u$ 的更新公式为： $\theta_u := \theta_u + \eta [\mathcal{D}(w, u) - \sigma(h_w^T \theta_u)] h_w$

同样地， $\frac{\partial L}{\partial h_w} = [\mathcal{D}(w, u) - \sigma(h_w^T \theta_u)] \theta_u$

上下文词的更新公式为： $v(\tilde{w}) := v(\tilde{w}) + \eta \sum_{u \in \text{Pos}(w) \cup \text{Neg}(w)} [\mathcal{D}(w, u) - \sigma(h_w^T \theta_u)] \theta_u$

## 1.3 Skip-Gram模型

仍旧使用上文的语料库"**NLP is so interesting and challenging**"，这次，我们的目标是通过中心词来预测上下文，也就是给定中心词，出现这些上下文词的概率最大。也即

$$\max P(is|NLP) * P(so|NLP) * P(NLP|is) * P(so|is) * P(interesting|is) * \dots$$

用公式表示如下：

$$\begin{aligned}\max_{\theta} L(\theta) &= \prod_{w \in D} \prod_{c \in C(w)} p(c|w) \\ &= \sum_{w \in D} \sum_{c \in C(w)} \log p(c|w)\end{aligned}$$

### 1.3.1 Hierarchical Softmax

### 1.3.2 Negative Sampling

## 2 常见面试问题

**Q1: 介绍一下Word2Vec模型。**

A: 两个模型: CBOW/Skip-Gram

两种加速方案: Hierarchical Softmax/Negative Sampling

**Q2: Word2Vec模型为什么要定义两套词向量?**

A: 因为每个单词承担了两个角色: 中心词和上下文词。通过定义两套词向量, 可以将两种角色分开。cs224n中提到是为了更方便地求梯度。参考见: <https://www.zhihu.com/answer/706466139>

**Q3: Hierarchical Softmax 和 Negative Sampling对比**

A: 基于Huffman树的Hierarchical Softmax 虽然在一定程度上能够提升模型运算效率, 但是, 如果中心词是生僻词, 那么在Huffman树中仍旧需要进行很复杂的搜索( $O(\log N)$ )。而Negative Sampling通过随机负采样来提升运算效率, 其复杂度和设定的负样本数 $K$ 线性相关( $O(K)$ ), 当 $K$ 取较小的常数时, 负采样在每一步的梯度计算开销都较小。

**Q4: HS为什么用霍夫曼树而不用其他二叉树?**

这是因为Huffman树对于高频词会赋予更短的编码, 使得高频词离根节点距离更近, 从而使得训练速度加快。

**Q5: Word2Vec模型为什么要进行负采样?**

A: 因为负样本的数量很庞大, 是 $O(|V|^2)$ 。

**Q6: 负采样为什么要用词频来做采样概率?**

为这样可以让频率高的词先学习, 然后带动其他词的学习。

**Q7: One-hot模型与Word2Vec模型比较?**

A: One-hot模型的缺点

- 稀疏 Sparsity
- 只能表示维度数量的单词 Capacity
- 无法表示单词的语义 Meaning

**Q8: Word2Vec模型在NNLM模型上做了哪些改进?**

A: 相同点: 其本质都可以看作是语言模型;

不同点: 词向量只不过 NNLM 一个产物, Word2vec 虽然其本质也是语言模型, 但是其专注于词向量本身, 因此做了许多优化来提高计算效率:

- 与 NNLM 相比, 词向量直接 sum, 不再拼接, 并舍弃隐层;
- 考虑到 softmax 归一化需要遍历整个词汇表, 采用 hierarchical softmax 和 negative sampling 进行优化, hierarchical softmax 实质上生成一颗带权路径最小的哈夫曼树, 让高

频词搜索路劲变小；negative sampling 更为直接，实质上对每一个样本中每一个词都进行负例采样；

#### Q9: Word2Vec与LSA对比？

A: LSA是基于共现矩阵构建词向量，本质上是基于全局语料进行SVD矩阵分解，计算效率低；而Word2Vec是基于上下文局部语料计算共现概率，计算效率高。

#### Q10: Word2Vec的缺点？

忽略了词语的语序；  
没有考虑一词多义现象

#### Q11: 怎么从语言模型理解词向量？怎么理解分布式假设？

词向量是语言模型的一个副产物，可以理解为，在语言模型训练的过程中，势必在一定程度上理解了每个单词的含义。而这在计算机的表示下就是词向量。

分布式假设指的是相同上下文语境的词有似含义。

#### 参考资料

word2vec 中的数学原理详解 <https://blog.csdn.net/itplus/article/details/37969519>

Word2Vec原理介绍 <https://www.cnblogs.com/pinard/p/7160330.html>

词向量介绍 <https://www.cnblogs.com/sandwichnlp/p/11596848.html>

一些关于词向量的问题 <https://zhuanlan.zhihu.com/p/56382372>

一个在线尝试Word2Vec的小demo <https://ronxin.github.io/wevi/>