

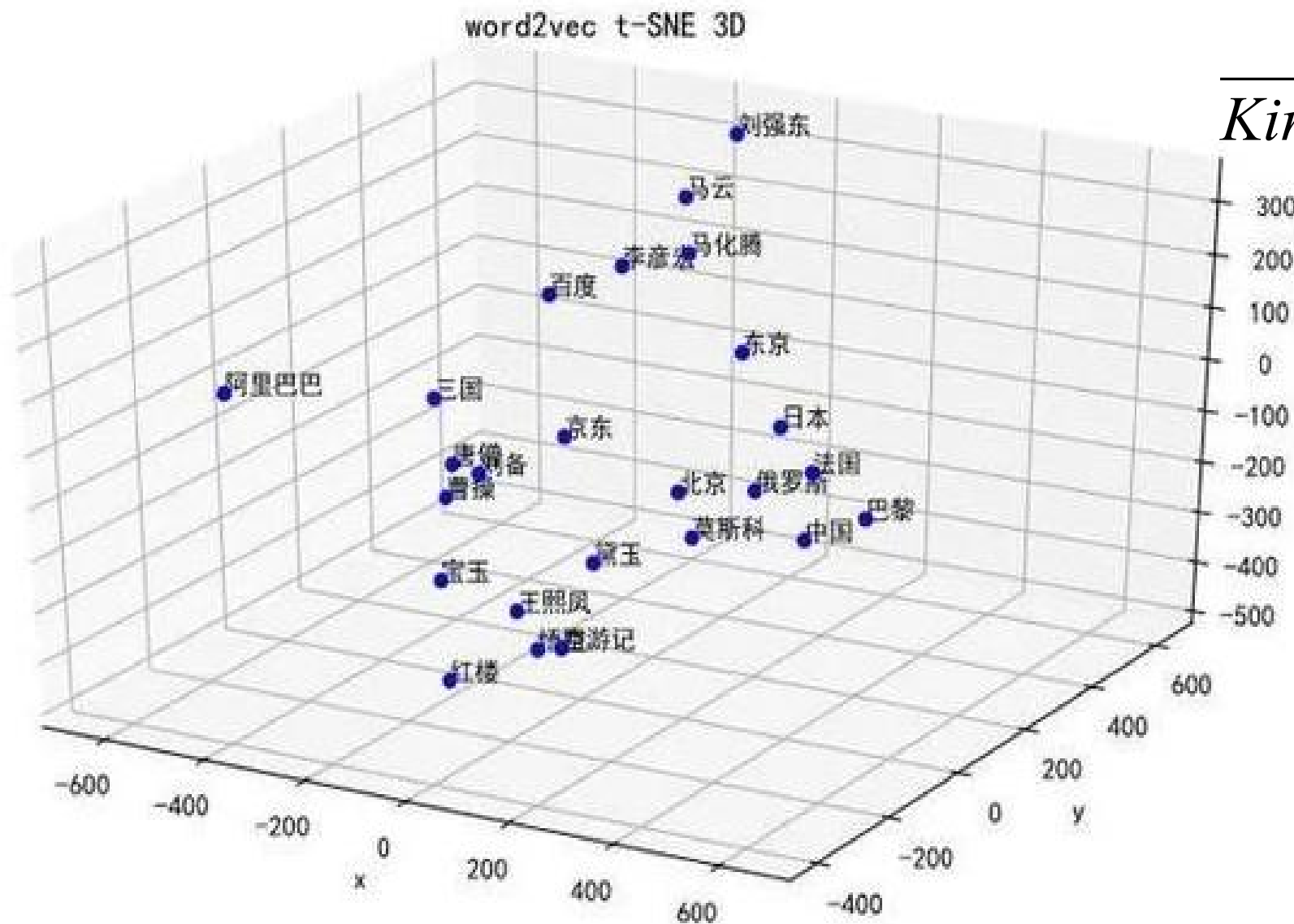
人工智能之机器学习

Word2Vector

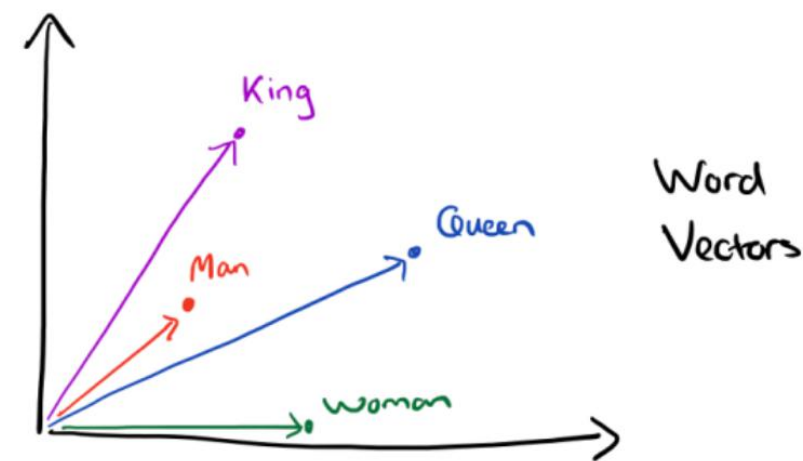
上海育创网络科技有限公司

主讲人：赵翌臣

Word2Vec直观理解



$$\overrightarrow{King} - \overrightarrow{Man} = \overrightarrow{Queen} - \overrightarrow{Woman}$$



Word2Vec的演进历史

- 统计语言模型
- n-gram模型
- 基于神经网络框架的CBOW模型和Skip-gram模型
- 基于Hierarchical Softmax框架的CBOW模型和Skip-gram模型
- 基于Negative Sampling框架的CBOW模型和Skip-gram模型

统计语言模型

- 随着大数据时代的来临，我们对自然语言处理技术（Nature Language Processing）的重视程度越来越大，其中统计语言模型（Statistical Language Model）是很重要的一环，它是所有NLP的基础，被广泛应用于语音识别，机器翻译，词性标注，句法分析等任务
- 思想：统计语言模型是用来计算一个句子的概率的概率模型，它是基于语料库来构建的模型。

统计语言模型

- 我/爱/北京/天安门, 天安门/上/太阳/升。
- $P(\text{我})$
- $P(\text{爱}|\text{我})$
- $P(\text{北京}|\text{我}, \text{爱})$
- $P(\text{天安门}|\text{我}, \text{爱}, \text{北京})$
- $P(\text{天安门}|\text{我}, \text{爱}, \text{北京}, \text{天安门})$
- $P(\text{上}|\text{我}, \text{爱}, \text{北京}, \text{天安门}, \text{天安门})$
- $P(\text{太阳}|\text{我}, \text{爱}, \text{北京}, \text{天安门}, \text{天安门}, \text{上})$
- $P(\text{升}|\text{我}, \text{爱}, \text{北京}, \text{天安门}, \text{天安门}, \text{上}, \text{太阳})$

- 缺点:
 - 模型参数太多, 影响求解与储存参数
- 催生出:
 - 近似的平滑n元语法(n-gram)模型

n-gram模型

- 在统计语言模型中，句子中一个词出现的概率与它前面的所有词都相关，如果句子很长这个计算量就会很大。
- n-gram思想：做了一个n-1阶的Markov假设，认为一个词出现的概率只与他前面的n-1个词相关。这样不仅使得单个参数的统计变得简单，参数总数减少
- 缺点：只是一定程度缓解了统计语言模型存在的问题

词向量的提出

- 在NLP任务中，我们将自然语言交给机器处理，但机器无法直接理解人类的语言，因此首先要把语言数学化，如何对自然语言数学化呢？词向量提供了一个很好的方式

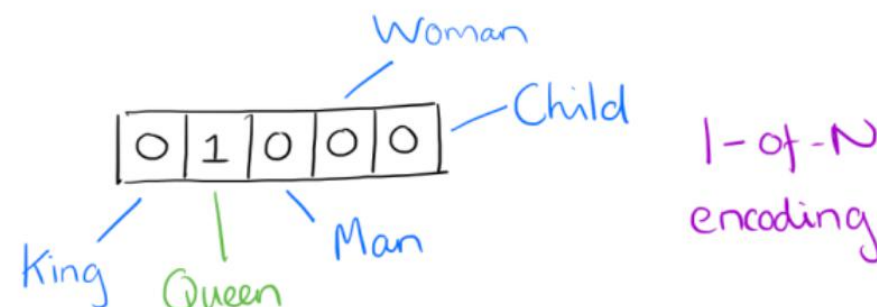
- One hot representation

- 缺点：

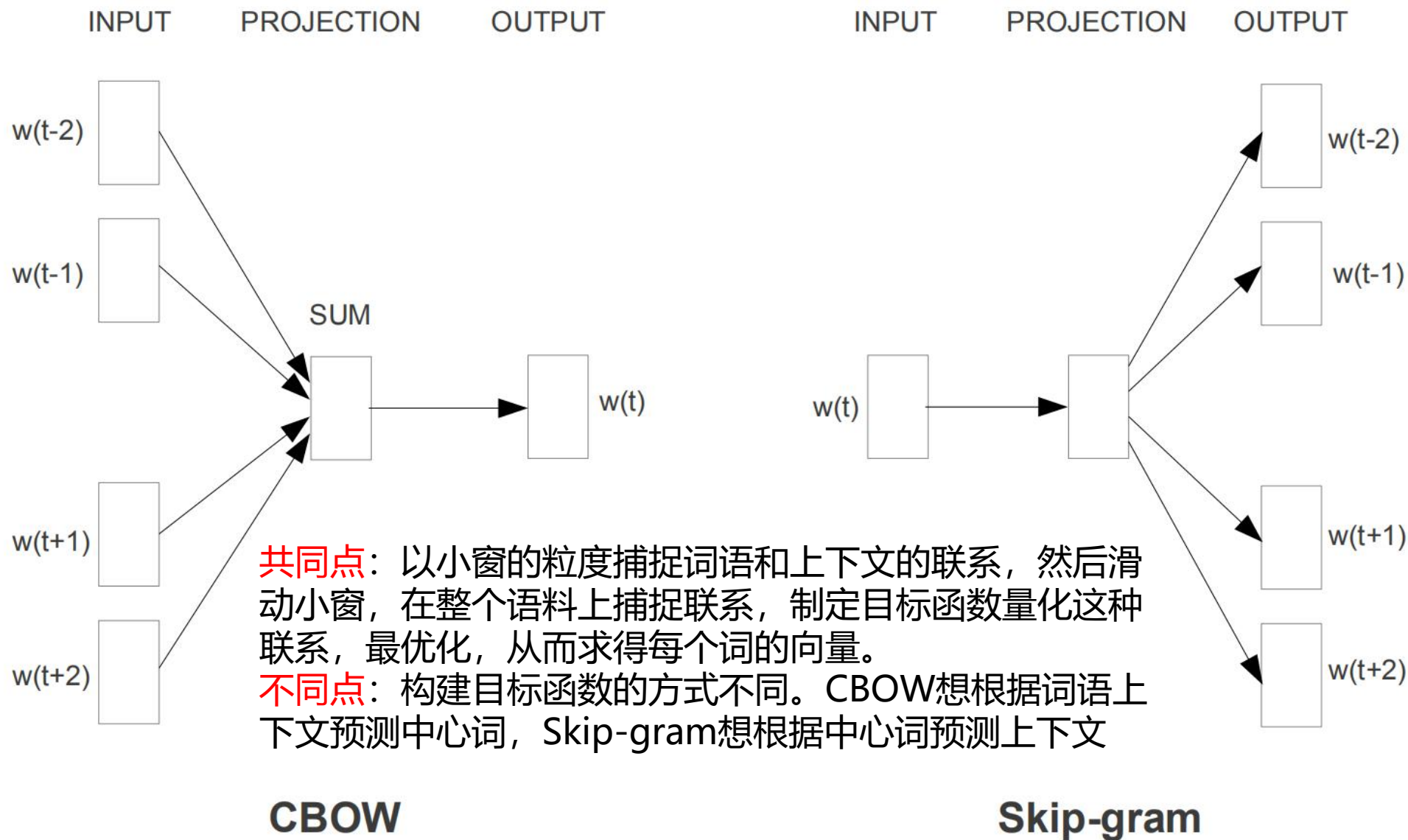
- 纬度高、增加训练难度、难以刻画词语间的联系

- Distributed representation

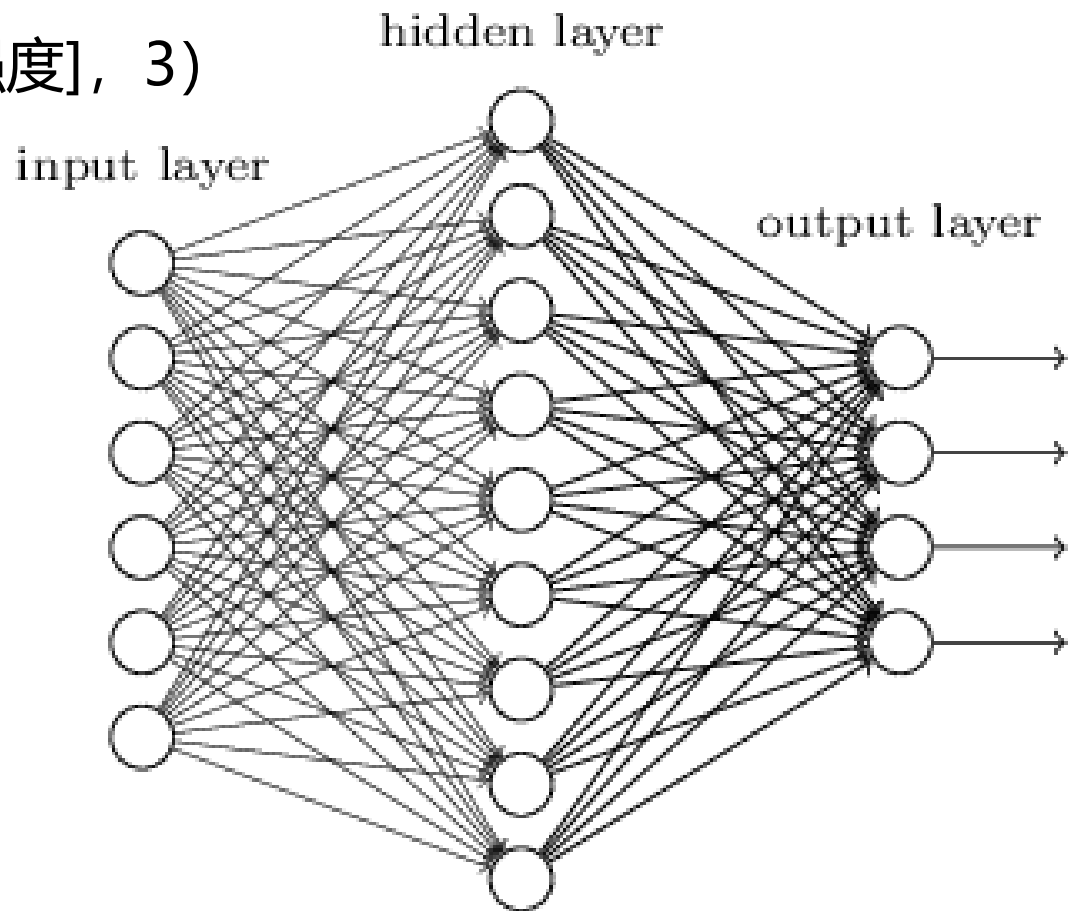
- 对任意词 w ，他的向量为 $v(w) \in R^m$ ， $v(w)$ 就称为 w 的词向量， m 为词向量长度



CBOW VS Skip-gram



- 神经网络结构：输入层、隐藏层、输出层
- sample ([土地类型, 距离水源距离, ..., 光照强度], 3)
- 前向传播算法
 - 从输入计算输出
- 反向传播算法
 - 更新神经网络的参数

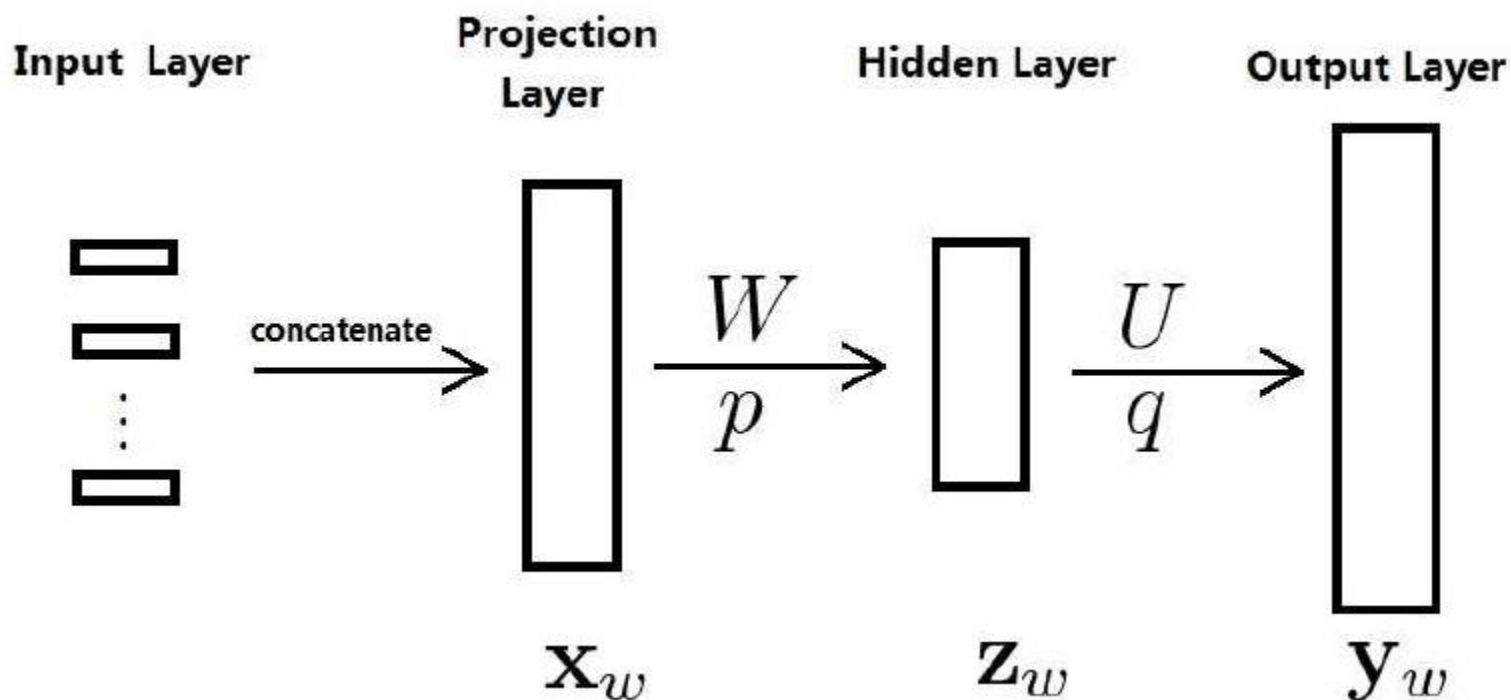


基于神经网络框架的CBOW模型 (2003)

- 输入是8个随机的词向量首尾相连，输出是所有词的softmax概率，训练过程让输出 learning 的概率最大

...an efficient method for learning high quality distributed vector ...

context focus word context

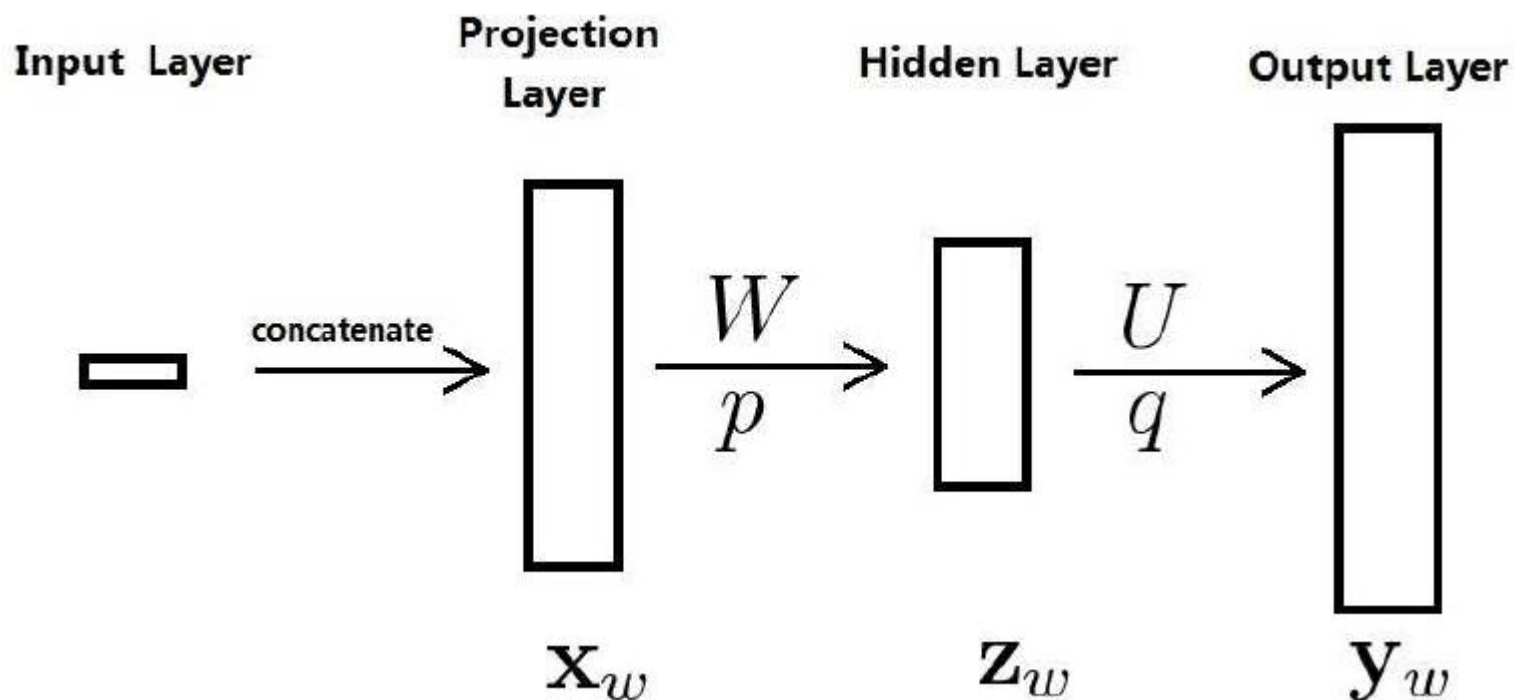


基于神经网络框架的Skip-gram模型 (2003)

- 输入是1个随机的词向量，输出是所有词的softmax概率，训练过程让排名前8的尽可能
是context word

...an efficient method for learning high quality distributed vector ...

context focus word context



基于神经网络框架的缺点

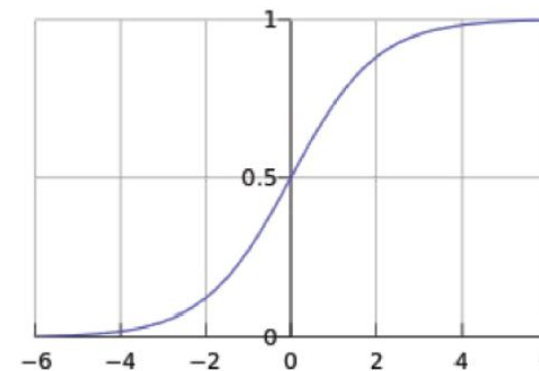
- word2vec神经网络型的原因
 - 神经网络模型的处理过程非常耗时，当词汇表在百万级别以上，使用softmax计算各个词的输出概率的计算量很大
- 催生出：
 - 基于Hierarchical Softmax框架的CBOW模型和Skip-gram模型
 - 基于Negative Sampling框架的CBOW模型和Skip-gram模型

Hierarchical Softmax基础

- Sigmoid函数是常用的激活函数之一，其定义为

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- 该函数的定义域为 $(-\infty, +\infty)$ ，值域为 $(0,1)$ ，函数图像为
- Sigmoid函数的导函数具有如下性质



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

$$\log[\sigma(x)]' = 1 - \sigma(x), \quad \log[(1 - \sigma(x))]' = -\sigma(x)$$

Hierarchical Softmax基础

• 树的概念

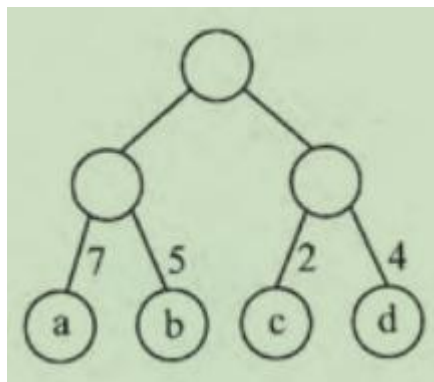
- 在计算机科学中，树是一种重要的非线性数据结构，它是数据元素按分支关系组织起来的结构。若干棵互不相交的树所构成的集合称为森林

• 路径和路径长度

- 在一棵树中，从一个结点往下可以达到的孩子或孙子结点之间的通路，称为路径。通路中分支的数据称为路径长度。若规定根节点的层号为1，则从根节点到第L层结点的路径长度为L-1

• 结点的权和带权路径长度

- 若为树中的结点赋予一个正数，则这个数值称为该结点的权。结点的带权路径长度是指，从根节点到该结点之间的路径长度与该节点的权的乘积



$$WPL = 7 \times 2 + 5 \times 2 + 2 \times 2 + 4 \times 2 = 36$$

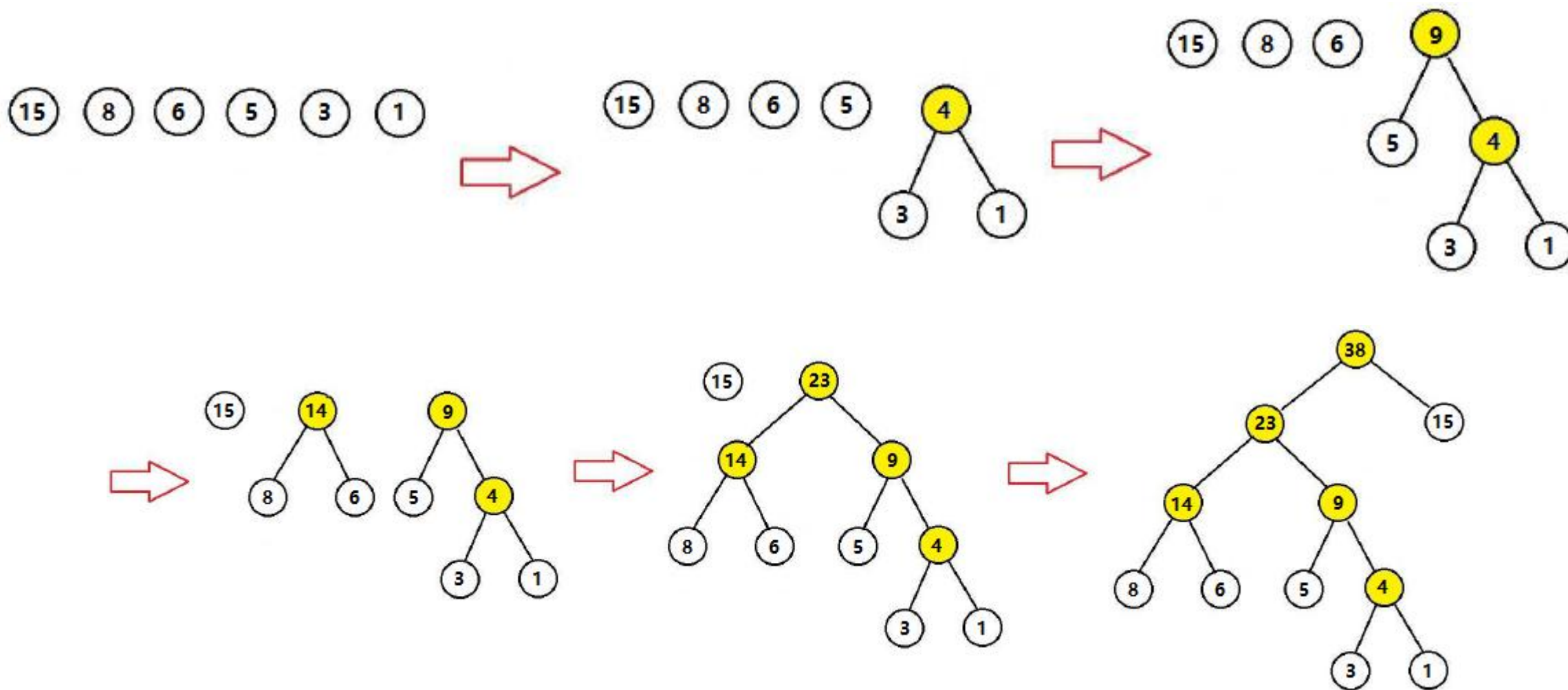
Hierarchical Softmax基础

- 树的带权路径长度
 - 规定为所有叶子节点的带权路径长度之和
- 二叉树
 - 是每个结点最多有两个子树的有序树。两个子树通常被称为“左子树”和“右子树”，定义中的“有序”是指两个子树有左右之分，顺序不能颠倒
- Huffman树
 - 给定 n 个权值作为 n 个叶子结点，构造一棵二叉树，若它的带权路径长度达到最小，则称这样的二叉树为最优二叉树，也称为Huffman树

Hierarchical Softmax基础

- 假设世界杯期间，从新浪微博中抓取了若干条与足球相关的微博，经统计，“我”、“喜欢”、“观看”、“巴西”、“足球”、“世界杯”这六个词出现的次数分别为15，8，6，5，3，1
- 请以这6个词为叶子结点，以相应词频当权值，构造一棵Huffman树

Hierarchical Softmax基础



由图可见词频越大的词离根结点越近

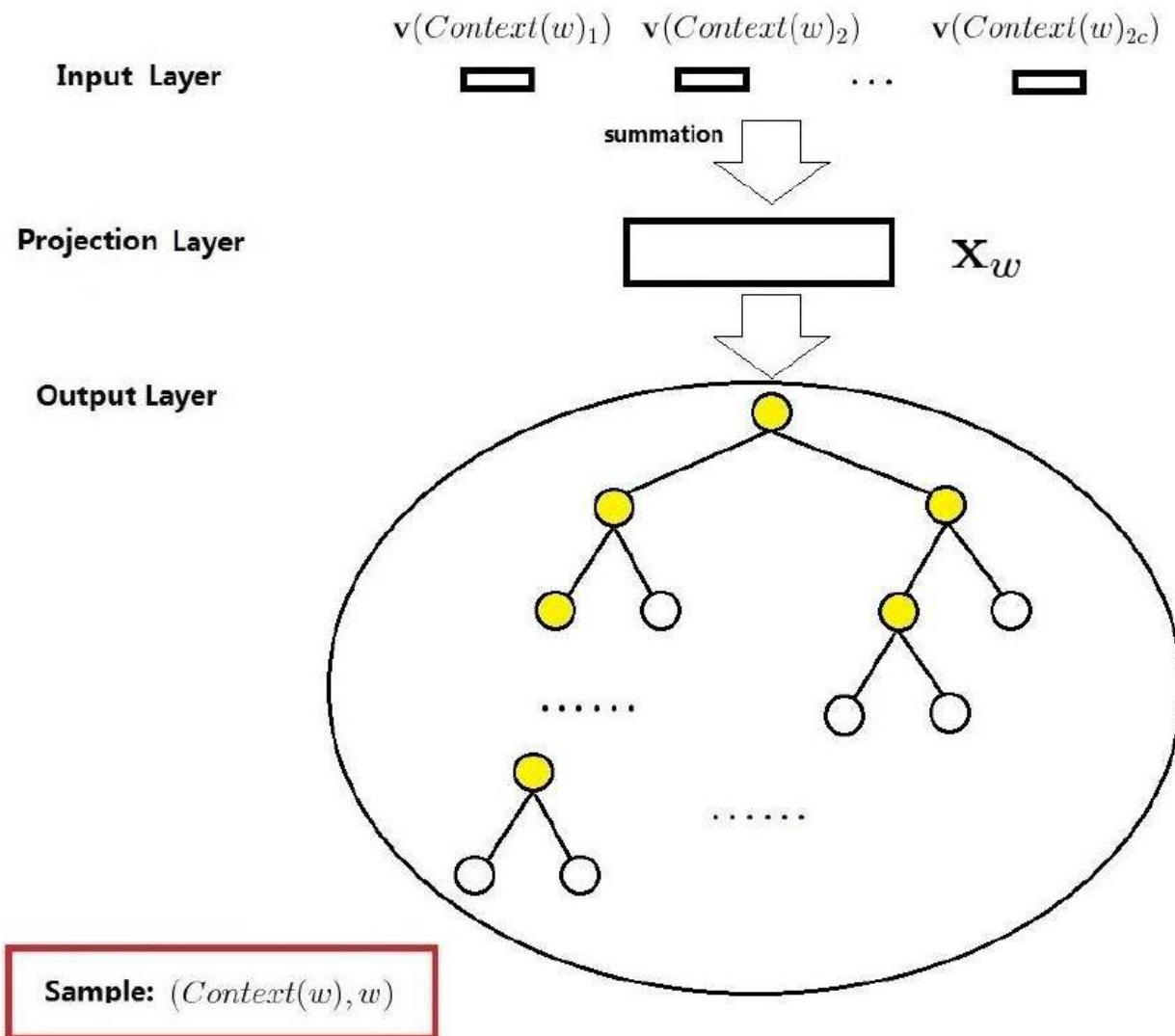
Hierarchical Softmax基础

- Huffman树的构造

- 给定 n 个权值 $\{w_1, w_2, \dots, w_n\}$ 作为二叉树的 n 个叶子结点，可通过以下算法来构造一棵Huffman树
- (1) 将 $\{w_1, w_2, \dots, w_n\}$ 看成是有 n 棵树的森林（每棵树仅有一个结点）
- (2) 在森林中选出两个根结点权值最小的树合并，作为一棵新树的左、右子树，且新树的根结点权值为其左、右子树根结点权值之和
- (3) 从森林中删除选取的两棵树，并将新树加入森林
- (4) 重复(2)、(3)步，直到森林中只剩一棵树为止，该树即为所求的Huffman树

基于Hierarchical Softmax框架的CBOW模型

- 输入层：中心词的前c个词向量与后c个词向量
- 投影层：词向量的叠加 x_w
- 输出层：根据语料库词频建立的霍夫曼树



基于Hierarchical Softmax框架的CBOW模型

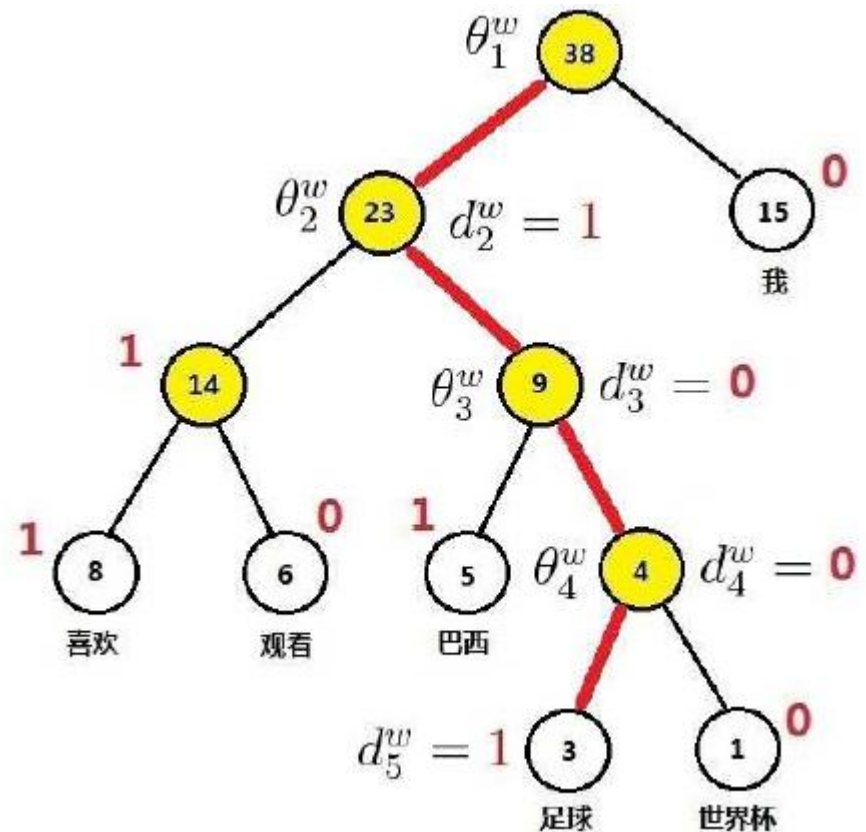
- 最终目的：得到每个词的向量 $v(w)$
- 手段：Define objective, and optimize it!!
- 我们现在的输入是 x_w ，因此每次更新不是词 w ，而是词 w 周边的词
- 为每个非叶子节点添加一个辅助参数 θ ，对于从根节点出发到达“足球”这个结点所经历的4次二分类，每次分类概率为

$$p(d_2^w | x_w, \theta_1^w) = 1 - \sigma(x_w^\top \theta_1^w);$$

$$p(d_3^w | x_w, \theta_2^w) = \sigma(x_w^\top \theta_2^w);$$

$$p(d_4^w | x_w, \theta_3^w) = \sigma(x_w^\top \theta_3^w);$$

$$p(d_5^w | x_w, \theta_4^w) = 1 - \sigma(x_w^\top \theta_4^w),$$



基于Hierarchical Softmax框架的CBOW模型

- 将这四个概率相乘得到：

$$p(\text{足球} | \text{Context}(\text{足球})) = \prod_{j=2}^5 p(d_j^w | \mathbf{x}_w, \theta_{j-1}^w).$$

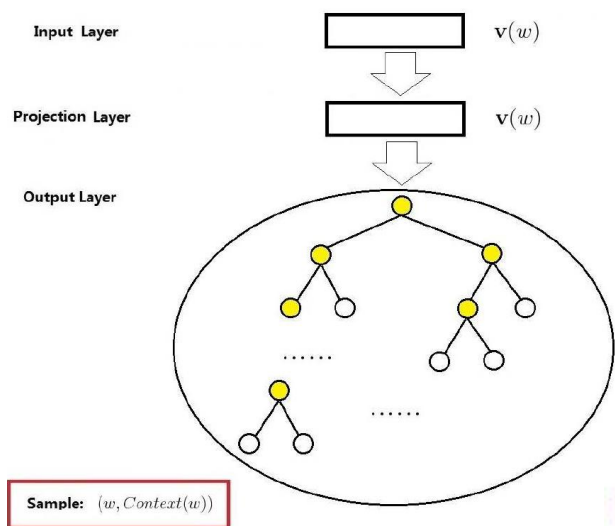
- 也就得到了足球上下文的优化函数，滑动小窗，也就扩展到全部预料：

$$\begin{aligned} \mathcal{L} &= \sum_{w \in \mathcal{C}} \log \prod_{j=2}^{l^w} \{ [\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]^{1-d_j^w} \cdot [1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]^{d_j^w} \} \\ &= \sum_{w \in \mathcal{C}} \sum_{j=2}^{l^w} \{ (1 - d_j^w) \cdot \log[\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] + d_j^w \cdot \log[1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] \}, \end{aligned}$$

- 使用最优化方法（随机梯度下降）对其极大化，便可得到每个词的辅助参数和词向量。

$$\theta_{j-1}^w := \theta_{j-1}^w + \eta \frac{\partial \mathcal{L}(w, j)}{\partial \theta_{j-1}^w} \quad \mathbf{v}(\tilde{w}) := \mathbf{v}(\tilde{w}) + \eta \sum_{j=2}^{l^w} \frac{\partial \mathcal{L}(w, j)}{\partial \mathbf{x}_w}, \quad \tilde{w} \in \text{Context}(w)$$

基于Hierarchical Softmax框架的Skip-gram模型



$$p(Context(w)|w) = \prod_{u \in Context(w)} p(u|w),$$

$$p(u|w) = \prod_{j=2}^{l^u} p(d_j^u | \mathbf{v}(w), \theta_{j-1}^u),$$

$$p(d_j^u | \mathbf{v}(w), \theta_{j-1}^u) = [\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{d_j^u}.$$

$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log \prod_{u \in Context(w)} \prod_{j=2}^{l^u} \{ [\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)]^{d_j^u} \}$$

$$= \sum_{w \in \mathcal{C}} \sum_{u \in Context(w)} \sum_{j=2}^{l^u} \{ (1 - d_j^u) \cdot \log[\sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] + d_j^u \cdot \log[1 - \sigma(\mathbf{v}(w)^\top \theta_{j-1}^u)] \}$$

使用最优化方法（随机梯度下降）对其极大化，便可得到每个词的辅助参数和词向量。

Hierarchical Softmax总结

- 第一步：为所有的词随机初始化自己的向量（200维）
- 第二步：根据词频构建霍夫曼树，并给每个非叶结点一个辅助变量
- 在基于Hierarchical Softmax的CBOW模型中，对某个词 w 来说，使用它的前 c 和后 c 个词向量的叠加作为词 w 的 $\text{context}(w)$ 记为 x_w ，根据霍夫曼树，可以构建一个从根节点到词 w 的函数（多个概率的累乘），我们目的是让这个函数极大，滑动小窗，扩展到整个语料库，就是让每个词的函数的累乘极大，这便是最终需要最大化的目标函数。
- 在基于Hierarchical Softmax的Skip-gram模型中，对某个词 w 来说，根据霍夫曼树，可以构建 $2c$ 个从根节点到词 w 上下文词语的函数（ $2c$ *多个概率的累乘），我们目的是让这个函数极大，扩展到整个语料库，就是让每个词的函数的累乘极大，这便是最终需要最大化的目标函数。

Hierarchical Softmax框架 VS 神经网络框架

- Hierarchical Softmax拿掉了隐藏层，输出层使用了霍夫曼树，由于使用霍夫曼树是高频的词靠近树根，这样高频词需要更少的时间会被找到

基于Negative Sampling框架的CBOW模型

- 与Hierarchical Softmax框架相比Negative Sampling框架不再使用复杂的霍夫曼树，而是利用简单的随机负采样，**目的是大幅提高训练速度并改善所得词向量的质量**，因而可作为Hierarchical Softmax的一种替代
- 在CBOW模型中，已知词 w 的上下文词向量叠加为 $\text{Context}(w)$ ，因此对于给定的 $\text{Context}(w)$ ，词 w 就是一个正样本，其他词就是负样本。
- 目标：得到每个词的向量
- 手段：Define objective, and optimize it!!

基于Negative Sampling框架的CBOW模型

- 为每个词初始化一个 θ 辅助向量，构建词 w 的目标函数：

$$g(w) = \sigma(\mathbf{x}_w^\top \theta^w) \prod_{u \in NEG(w)} [1 - \sigma(\mathbf{x}_w^\top \theta^u)]$$

- 意义：增大选中正样本的概率同时降低选中负样本的概率，对于整个语料库

$$G = \prod_{w \in \mathcal{C}} g(w)$$

- 目标函数为：

$$\begin{aligned} \mathcal{L} &= \log G = \log \prod_{w \in \mathcal{C}} g(w) = \sum_{w \in \mathcal{C}} \log g(w) \\ &= \sum_{w \in \mathcal{C}} \log \prod_{u \in \{w\} \cup NEG(w)} \left\{ [\sigma(\mathbf{x}_w^\top \theta^u)]^{L^w(u)} \cdot [1 - \sigma(\mathbf{x}_w^\top \theta^u)]^{1-L^w(u)} \right\} \\ &= \sum_{w \in \mathcal{C}} \sum_{u \in \{w\} \cup NEG(w)} \left\{ L^w(u) \cdot \log [\sigma(\mathbf{x}_w^\top \theta^u)] + [1 - L^w(u)] \cdot \log [1 - \sigma(\mathbf{x}_w^\top \theta^u)] \right\} \end{aligned}$$
- 使用最优化方法（随机梯度下降）对其极大化，便可得到每个词的辅助参数和词向量。

$$\theta^u := \theta^u + \eta \frac{\partial \mathcal{L}(w, u)}{\partial \theta^u} \quad \mathbf{v}(\tilde{w}) := \mathbf{v}(\tilde{w}) + \eta \sum_{u \in \{w\} \cup NEG(w)} \frac{\partial \mathcal{L}(w, u)}{\partial \mathbf{x}_w}, \quad \tilde{w} \in Context(w).$$

基于Negative Sampling框架的Skip-gram模型

- 有了基于Negative Sampling框架的CBOW模型的推导经验，Skip-gram可以顺水推舟，
- 将CBOW的目标函数：

$$G = \prod_{w \in \mathcal{C}} g(w)$$

- 改为：

$$G = \prod_{w \in \mathcal{C}} \prod_{u \in \text{Context}(w)} g(u)$$

- 最终目标函数：

$$\mathcal{L} = \log G = \log \prod_{w \in \mathcal{C}} \prod_{u \in \text{Context}(w)} g(u) = \sum_{w \in \mathcal{C}} \sum_{u \in \text{Context}(w)} \log g(u)$$







- 使用最优化方法（随机梯度下降）对其极大化，便可得到每个词的辅助参数和词向量。

Negative Sampling总结

- 首先为所有的词随机初始化自己的向量（200维）
- 在基于Negative Sampling的CBOW模型中，对某个词 w 来说，我们构建一个函数，这个函数刻画的是：预测出词 w 的概率 * 不预测出其他词的概率，这是我们想要极大的，扩展到整个语料库，就是让每个词的函数的累乘极大，这便是最终需要最大化的目标函数；
- 在基于Negative Sampling的Skip-gram模型中，对某个词 w 来说，我们构建一个函数，这个函数刻画的是：预测出词 w 上下文词语的概率 * 不预测出 w 上下文之外词语的概率，这是我们想要极大的，扩展到整个语料库，就是让每个词的函数的累乘极大，这便是最终需要最大化的目标函数。

安装gensim

- 方式一： 下载gensim离线包
 - cd到whl的目录， 打开控制台执行pip install xxx.whl
- 方式二： pip install gensim
- 官网
 - <https://pypi.org/project/gensim/>

gensim-3.5.0-cp36-cp36m-manylinux1_i686.whl (23.5 MB) 	Wheel	cp36	
gensim-3.5.0-cp36-cp36m-manylinux1_x86_64.whl (23.5 MB) 	Wheel	cp36	
gensim-3.5.0-cp36-cp36m-win32.whl (23.4 MB) 	Wheel	cp36	Jul 6, 2018
gensim-3.5.0-cp36-cp36m-win_amd64.whl (23.5 MB) 	Wheel	cp36	Jul 6, 2018
gensim-3.5.0.tar.gz (22.9 MB) 	Source	None	Jul 6, 2018
gensim-3.5.0.win32-py2.7.exe (23.6 MB) 	Windows Installer	2.7	Jul 6, 2018

```

C:\Windows\system32\cmd.exe
Downloading https://files.pythonhosted.org/packages/b7/31/05c8d001f7487f0f07289a5fc0fc3832e9a57f24bd4d3b0fee70e0d51365a
/jmespath-0.9.3-py2.py3-none-any.whl
Collecting boto3<1.12.0,>=1.11.8 (from boto3->smart-open==1.2.1->gensim==3.5.0)
Downloading https://files.pythonhosted.org/packages/43/96/f5c5fd630f8f8c36800ec5e1daa474c065a75ea1d011f26d1fa934137e18
/boto3-1.11.8-py2.py3-none-any.whl (4.6MB)
100% |#####| 4.7MB 92kB/s
Collecting s3transfer<0.2.0,>=0.1.10 (from boto3->smart-open==1.2.1->gensim==3.5.0)
Downloading https://files.pythonhosted.org/packages/d7/14/2a0004d487464d120c9fb85313a75cd3d71a7506955be458eebfe19a6b1d
/s3transfer-0.1.13-py2.py3-none-any.whl (59kB)
100% |#####| 61kB 1.3MB/s
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in d:\anaconda3\lib\site-packages (from boto3->smart-open==1.2.1->gensim==3.5.0)
Collecting urllib3<1.24,>=1.12.0 (from boto3->smart-open==1.2.1->gensim==3.5.0)
Downloading https://files.pythonhosted.org/packages/bd/c9/6fdd990019071a4a32a5e7cb78a1d92c53851ef4f56f62a3486e6a7d8ffb
/urllib3-1.23-py2.py3-none-any.whl (133kB)
100% |#####| 143kB 1.5MB/s
Requirement already satisfied: docutils<0.10 in d:\anaconda3\lib\site-packages (from boto3->smart-open==1.2.1->gensim==3.5.0)
Building wheels for collected packages: smart-open, bz2file
Running setup.py bdist_wheel for smart-open ... done
Stored in directory: C:\Users\ibf\AppData\Local\pip\Cache\wheels\73\1f\9b\c9f93d4ba073b6f79b1ed9df68ab5ce048d3136d0efc
f90e30
Running setup.py bdist_wheel for bz2file ... done
Stored in directory: C:\Users\ibf\AppData\Local\pip\Cache\wheels\81\75\d6\ea1317bf09f1af5a30bfc2a007869fa6e1f516b8f7c
591cb9
Successfully built smart-open bz2file
Installing collected packages: bz2file, jmespath, urllib3, boto3, s3transfer, boto3, smart-open, gensim
Successfully installed boto3-1.8.8 boto3-1.11.8 bz2file-0.98 gensim-3.5.0 jmespath-0.9.3 s3transfer-0.1.13 smart-open
-1.6.0 urllib3-1.23
You are using pip version 9.0.1, however version 18.0 is available.

```

Word2vec模型介绍

- `most_similar(positive=[u"中国",u"战争"],topn=20)` # 20个最相关的
- # 抗日战争 国人 中日战争 侵略战争 二战 国家 我国 本国 中华民族 鸦片战争 抗战 南京大屠杀 文化大革命 别国 中日 当今世界 美国 国民性 解放战争
- `y2 = model.most_similar(positive=[u"美国",u"战争"],topn=20)` # 20个最相关的
- # 二战 越战 伊战 越南战争 侵略战争 内战 第二次世界大战 南北战争 国家 二次世界大战 二次大战 朝鲜战争 太平洋战争 当今世界 抗日战争 米国 独立战争 殖民主义 霸权主义 强权政治

word2vec 参数

- **sentences**: 要分析的语料，可以是一个列表，或者从文件中遍历读出。
- **size**: 词向量的维度，默认值是**100**。这个维度的取值一般与我们的语料的大小相关，如果是不大的语料，比如小于**100M**的文本语料，则使用默认值一般就可以了。如果是超大的语料，建议增大维度。
- **window**: 即词向量上下文最大距离，**window**越大，则和某一词较远的词也会产生上下文关系。默认值为**5**。在实际使用中，可以根据实际的需求来动态调整这个**window**的大小。如果是小语料则这个值可以设的更小。对于一般的语料这个值推荐在**[5,10]**之间。
- **sg**: 即我们的word2vec两个模型的选择了。如果是**0**，则是CBOW模型，是**1**则是Skip-Gram模型，默认是**0**即CBOW模型。
- **hs**: 即我们的word2vec两个解法的选择了，如果是**0**，则是Negative Sampling，是**1**则是Hierarchical Softmax。默认是**0**。
- **negative**: 即使用Negative Sampling时负采样的个数，默认是**5**。推荐在**[3,10]**之间。

word2vec 参数

- **min_count**: 需要计算词向量的最小词频。这个值可以去掉一些很生僻的低频词，默认是5。如果是小语料，可以调低这个值。
- **iter**: 随机梯度下降法中迭代的最大次数，默认是5。对于大语料，可以增大这个值。
- **alpha**: 在随机梯度下降法中迭代的初始步长。默认是0.025。

基于Word2vec推荐系统简介

