慧科 只为赋能人才 | 开课吧 kaikeba.com

文本分类

jwzheng

第二节课课程计划

慧科 只为赋能人才 | 开课吧 kaikeba.com

- 什么是词向量
- onehot词向量
- 分布式词向量
- word2vec模型
- 实践：onehot编码的实现
- 实践：使用gensim训练word2vec
- 实践：使用keras实现cbow模型
- 实践：使用fasttext做新闻数据集分类（待定）

词向量

- 什么是词向量
- 词的表示
  - one-hot representation：one hot编码
  - distributed representation：word2vec

one-hot representation

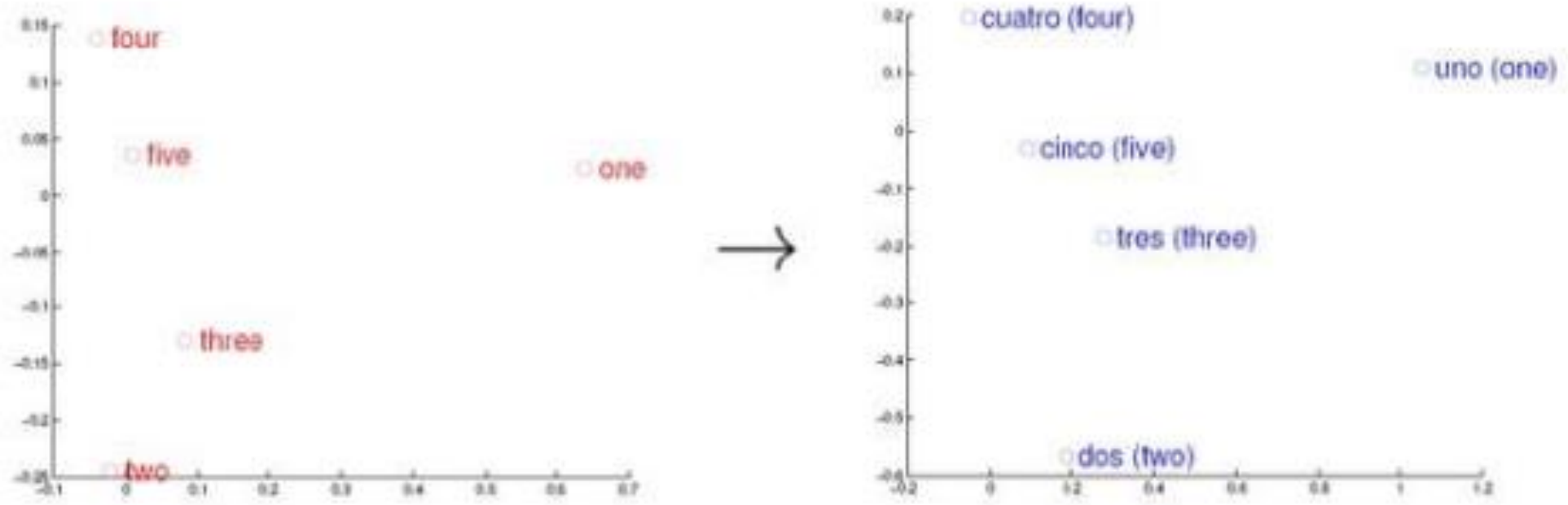
	西瓜	梨子	篮球	足球	猫	狗
西瓜	1	0	0	0	0	0
梨子	0	1	0	0	0	0
篮球	0	0	1	0	0	0
足球	0	0	0	1	0	0
猫	0	0	0	0	1	0
狗	0	0	0	0	0	1

- 最常见的词向量表示方式
- 一个实例：
  - 假设 我们现在有西瓜，梨子，篮球，足球，猫，狗六个词，如何使用one-hot词向量来表示各个词呢？
  - 西瓜：[1,0,0,0,0,0] 梨子：[0,1,0,0,0,0]
  - 篮球：[0,0,1,0,0,0] 足球：[0,0,0,1,0,0]
  - 猫：[0,0,0,0,1,0] 狗：[0,0,0,0,0,1]
- 假设语料库中的词的个数为N，则用一个N维向量来表示该词。这个词出现的位置为1，其他位置为0.
- 缺点：
  - 维度太高
  - 不能体现不同词之间的相关性
  - example：我们去吃饭 我们：[1,0,0,0...0] 去：[0,0,..1,0,0,...,0] 吃饭 :[0,0..,0,1]

distributed representation

- 可以解决one hot词向量的缺点
- 一个实例：
  - 假设 我们现在有西瓜，梨子，篮球，足球，猫，狗六个词，如何使用one-hot词向量来表示各个词呢？
  - 西瓜：[0.3,0.02,0.01] 梨子：[0.2,0.01,0.01]
  - 篮球：[0.01,0.7,0.01] 足球：[0.02,0.4,0.01]
  - 猫：[0.01,0.04,0.2] 狗：[0.03,0.01,0.1]
  - 每两个词向量之间可以计算相似度
- 如何得到分布式词向量
  - word2vec 通过训练语言模型得到词向量的表示

distributed representation



神经网络语言模型

$$\mathbf{y}_w = (y_{w,1}, y_{w,2}, \cdots, y_{w,N})^\top$$

- 神经网络语言模型
  - Context(w)表示w前面的n-1个词
  - 样本：（ Context(w),w ）
  - 概率：softmax归一化求得概率

$$p(w|Context(w)) = \frac{e^{y_{w,i_w}}}{\sum_{i=1}^{N} e^{y_{w,i}}},$$

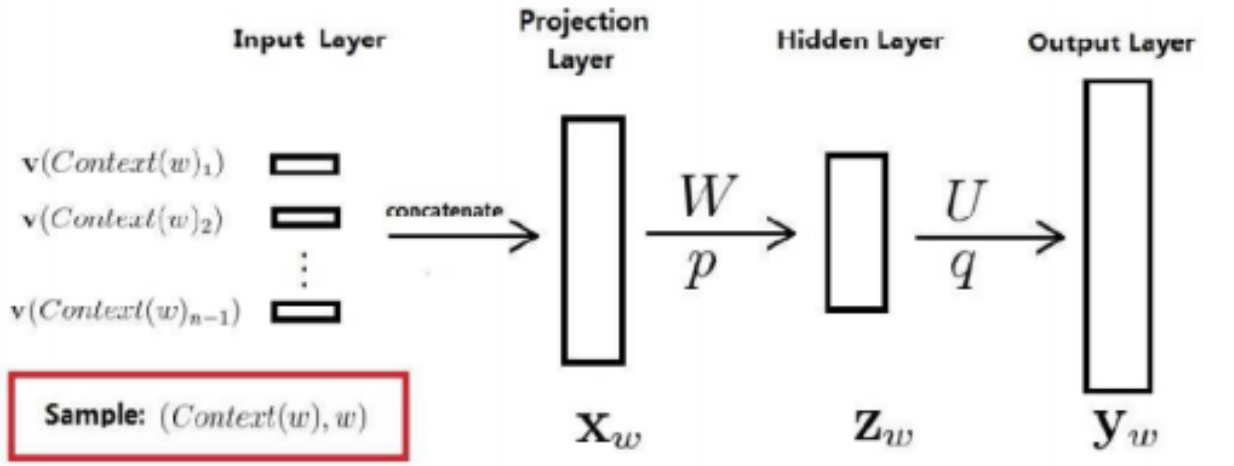


图 4 神经网络结构示意图

- 通过上面的方式来计算

$$P(S) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2)...P(w_n \mid w_{n-1}) \qquad p(w_i \mid w_{i-1}) = \frac{p(w_{i-1}, w_i)}{p(w_{i-1})}$$

$$p(w_{i-1}, w_i) = \frac{N(w_{i-1}, w_i)}{N}$$

神经网络语言模型

- 神经网络语言模型优化方向
  - 分层softmax
  - 目的：降低计算量
- 神经网络语言模型和n-gram语言模型
  - s1 = ' A dog is running in the room' 1000次
  - s2 = ' A cat is running in the room' 10次
  - s3 = ' A cat is walking in the room' 1次
  - P(s1),P(s2),P(S3)
- 概率平滑

$$p(w_{i-1}, w_i) = \frac{N(w_{i-1}, w_i)}{N}$$

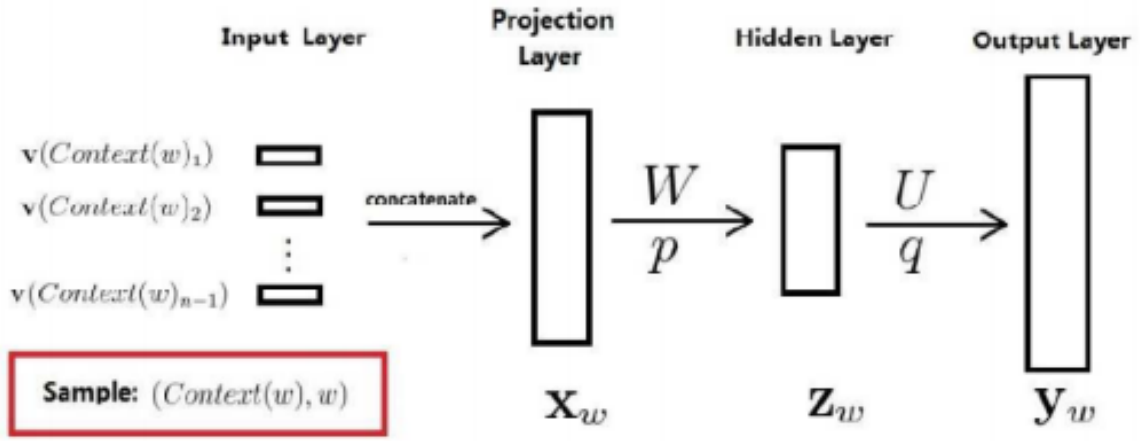


图 4 神经网络结构示意图

接下来, 简要地分析一下上述模型的运算量. 在如图 4 所示的神经网络中, 投影层、隐藏层和输出层的规模分别为 $(n-1)m$, n_h, N, 依次看看其中涉及的参数:

(1) n 是一个词的上下文中包含的词数, 通常不超过 5;

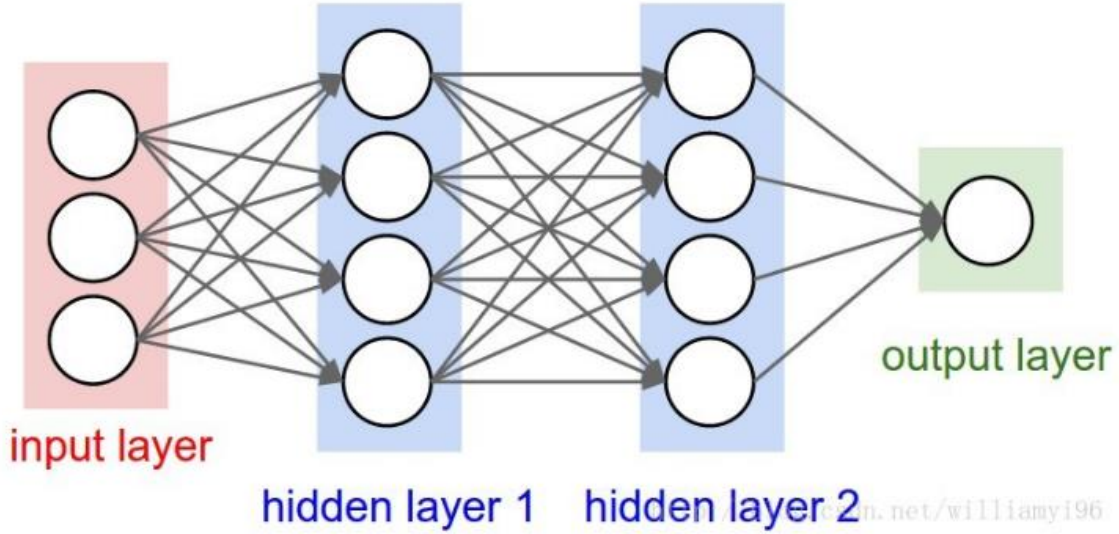
(2) m 是词向量长度, 通常是 $10^1 \sim 10^2$ 量级;

(3) n_h 由用户指定, 通常不需取得太大, 如 10^2 量级;

(4) N 是语料词汇量的大小, 与语料相关, 但通常是 $10^4 \sim 10^5$ 量级.

全连接神经网络回顾

- 全连接神经网络
  - 输入层
  - 隐含层
  - 输出层



word2vec

我们/去/动物园/看/猴子

- 通过神经网络模型进行训练得到每个词的向量表示。
- 常用的两个word2vec模型：
  - CBOW模型
  - Skip-Gram模型
- CBOW模型
  - 输入上下文的词，预测当前词。
- Skip-Gram模型
  - 输入当前词，预测周围的词

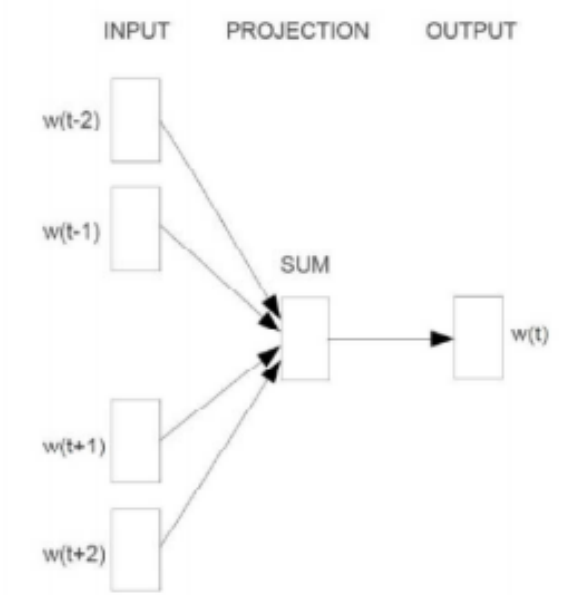


图 8 CBOW 模型

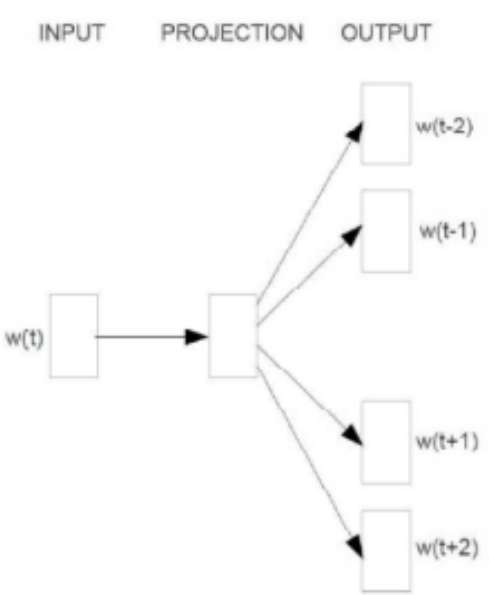


图 9 Skip-gram 模型

- 今天/天气/很/好，我们/一起/去/爬山.

word2vec之CBOW模型

例如：今天/天气/很/好，我们/一起/去/爬山

- 输入层
  - w(t-2) w(t-1) w(t+1) w(t+2)
  - 窗口大小：物理含义和n-gram中相近
- 输出层
  - w(t)
- 投影层
  - SUM操作
- 当窗口大小为4时，生成的训练样本如下：
  - （今天/天气/好/我们，很）
  - （天气/很/我们/一起，好）
  - （很/好/一起/去，我们）
  - （好/我们/去/爬山，一起）

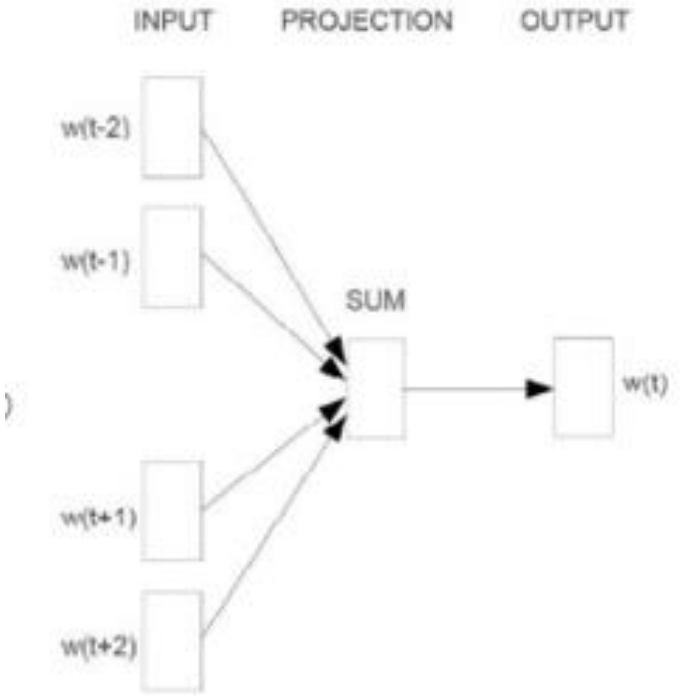


图 8 CBOW 模型

word2vec之CBOW模型

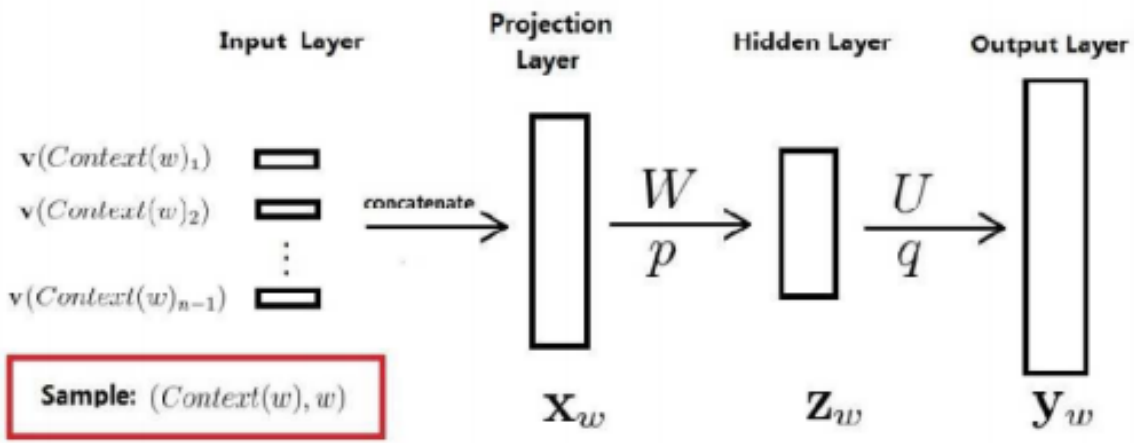
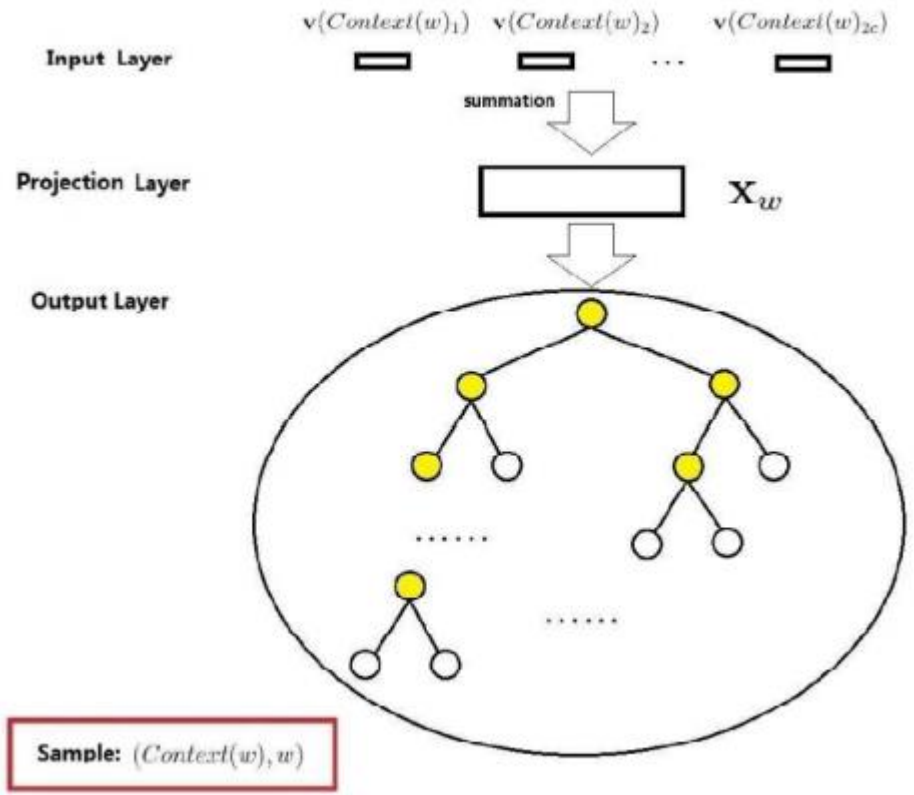


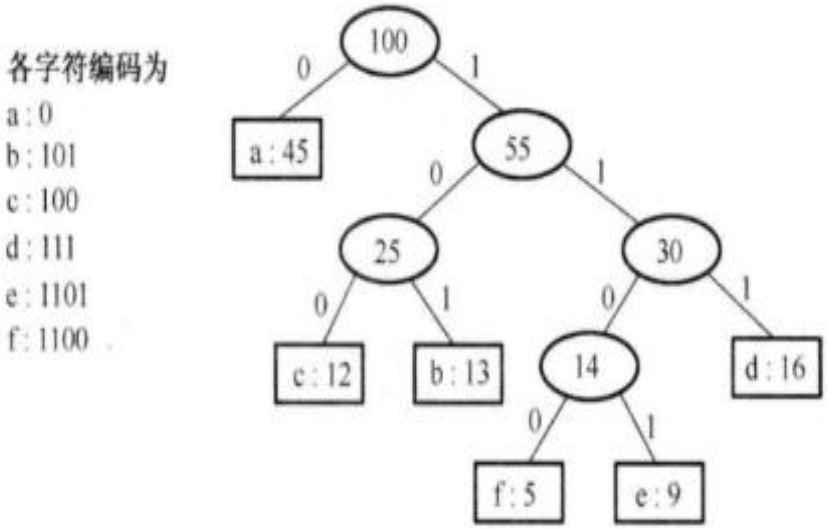
图 4 神经网络结构示意图



1. (从输入层到投影层的操作) 前者是通过拼接, 后者通过**累加求和**.
2. (隐藏层) 前者有隐藏层, 后者**无隐藏层**.
3. (输出层) 前者是线性结构, 后者是**树形结构**.

word2vec之CBOW模型进阶

- Huffman树
  - n个带有权重的叶子节点，构造一个二叉树，使得树的带权路径长度最小，称为最优二叉树，也叫哈夫曼树(Huffman Tree)
  - 哈夫曼树是带权路径长度最短的树，权值较大的结点离根较近。
  - 路径和路径长度
  - 节点的权重和带权路径
  - 树的带权路径
- Huffman树的构造
  - 原始叶子节点如下：
  - a:45 c:12 b:13 d:16 f:5 e:9
  - step1：45 12 13 16 14
  - step2：45 16 14 25
  - step3：45 25 30
  - step4：45 55
  - step5：100



算法 2.1 (*Huffman* 树构造算法)

(1) 将 $\{w_1, w_2, \cdots, w_n\}$ 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的权值最小的树合并，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.

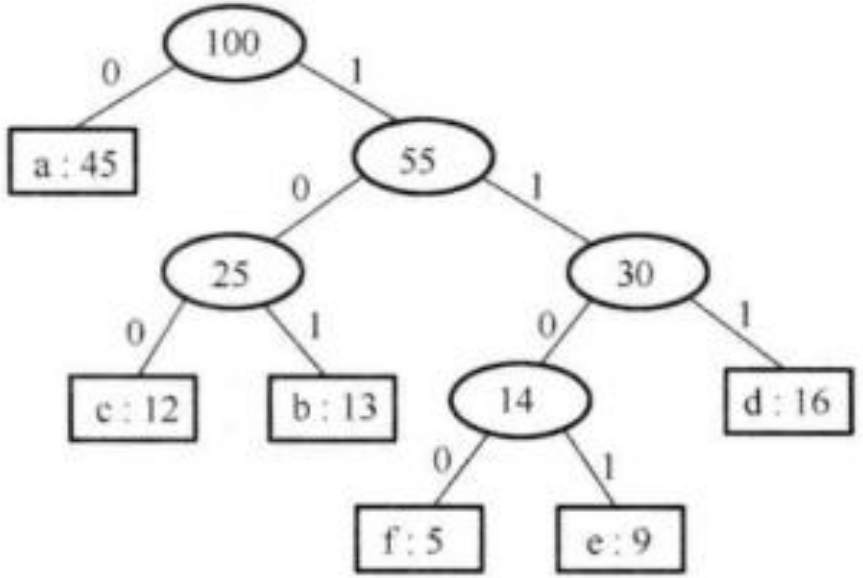
(3) 从森林中删除选取的两棵树，并将新树加入森林.

(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.

word2vec之CBOW模型进阶

- Huffman编码
  - 等长编码
  - 优化编码长度，节省内存
  - 文档的编码最短
- 编码约定
  - 权重小的节点为左节点，大的为右节点
  - 左孩子节点编码为0，右孩子编码为1

各字符编码为
a : 0
b : 101
c : 100
d : 111
e : 1101
f : 1100



- abababababaef

word2vec之CBOW模型进阶

- 分层Softmax
  - 若干个二分类
  - 正类 $\sigma(X_w^T\theta)=\frac{1}{1+e^{-X_w^T\theta}}$
  - 负类 $1-\sigma(X_w^T\theta)$
  - 左节点(1)是负类 右节点(0)是正类

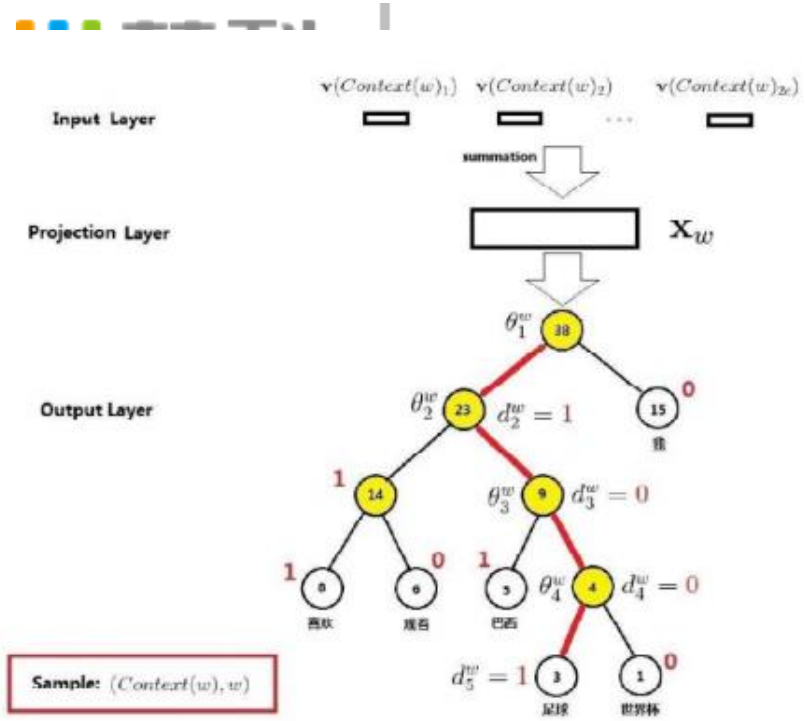


图 11 $w=$“足球”时的相关记号示意图

对于从根结点出发到达“足球”这个叶子节点所经历的 4 次二分类, 将每次分类结果的概率写出来就是

1. 第 1 次: $p(d_2^w|\mathbf{x}_w,\theta_1^w)=1-\sigma(\mathbf{x}_w^\top\theta_1^w)$;
2. 第 2 次: $p(d_3^w|\mathbf{x}_w,\theta_2^w)=\sigma(\mathbf{x}_w^\top\theta_2^w)$;
3. 第 3 次: $p(d_4^w|\mathbf{x}_w,\theta_3^w)=\sigma(\mathbf{x}_w^\top\theta_3^w))$;
4. 第 4 次: $p(d_5^w|\mathbf{x}_w,\theta_4^w)=1-\sigma(\mathbf{x}_w^\top\theta_4^w)$,

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

word2vec之CBOW模型进阶

- 分层Softmax
  - 若干个二分类

至此, 通过 w = “足球” 的小例子, Hierarchical Softmax 的基本思想其实就已经介绍完了. 小结一下: **对于词典 $\mathcal{D}$ 中的任意词 w, Huffman 树中必存在一条从根结点到词 w 对应结点的路径 p^w (且这条路径是唯一的). 路径 p^w 上存在 l^w-1 个分支, 将每个分支看做一次二分类, 每一次分类就产生一个概率, 将这些概率乘起来, 就是所需的 $p(w|Context(w))$.**

条件概率 $p(w|Context(w))$ 的一般公式可写为

$$p(w|Context(w)) - \prod_{j=2}^{l^w} p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w), \tag{4.3}$$

其中

$$p(d_j^w|\mathbf{x}_w, \theta_{j-1}^w) = \begin{cases} \sigma(\mathbf{x}_w^\top \theta_{j-1}^w), & d_j^w = 0; \\ 1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w), & d_j^w = 1, \end{cases}$$

或者写成整体表达式

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

$$\mathcal{L} = \sum_{w\in\mathcal{C}} \log p(w|Context(w)),$$

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

word2vec之CBOW模型进阶

- 如何更新词向量v(w)

x(w)是由v(W)求和得到的，

x(w)=SUM(v(w))

将x(w)的改变量平均到v(w)上

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

$$= \sum_{w\in\mathcal{C}} \sum_{j=2}^{l^w} \left\{ (1-d_j^w) \cdot \log[\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] + d_j^w \cdot \log[1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] \right\},$$

$$\theta_{j-1}^w := \theta_{j-1}^w + \eta \left[1 - d_j^w - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w) \right] \mathbf{x}_w,$$

- 问题：初始的词向量v(w)如何设定？
- 所有的词向量每次更新的值都一样

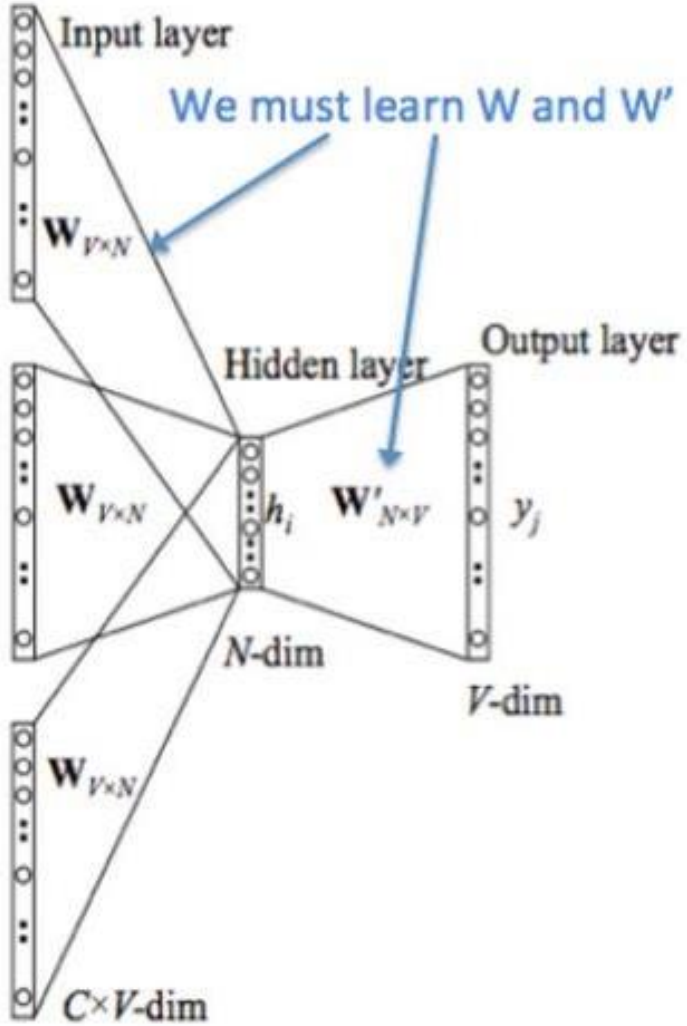
$$\frac{\partial \mathcal{L}(w,j)}{\partial \mathbf{x}_w} = \left[1 - d_j^w - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w) \right] \theta_{j-1}^w.$$

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

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

word2vec之CBOW模型进阶

- 输入的是什么
  - w(t-2) w(t-1) w(t+1) w(t+2) 上下文大小为C
  - 单词onehot编码 每个单词V维度
  - 输入C*V维
- 投影层如何工作
  - 输入层到投影层的权重矩阵W　V*N维度 W随机初始化
  - 所有onehot向量分别乘以共享的输入权重矩阵W.
  - 相加
- 输出层
  - 投影层到输出层的权重矩阵W ‘ N*V维度 W’ 随机初始化
  - 乘以输出权重矩阵W' {N*V} 为什么这里也是v维度
- 误差
  - 目标变量是onehot编码
- 如何得到词向量
  - 输入层的每个单词与矩阵W相乘得到的向量的就是该单词的词向量
- 副产物
  - 通过训练语言模型间接得到词向量，直接产物是神经网络语言模型



word2vec之Skip-Gram模型

- 输入层
  - w(t)
  - 窗口大小：物理含义和n-gram中相近
- 输出层
  - w(t-2) w(t-1) w(t+1) w(t+2)
- 投影层
  - 恒等操作
- 当窗口大小为4时，生成的训练样本如下：
  - （很，今天/天气/好/我们）
  - （好，天气/很/我们/一起）
  - （我们，很/好/一起/去）
  - （一起，好/我们/去/爬山）

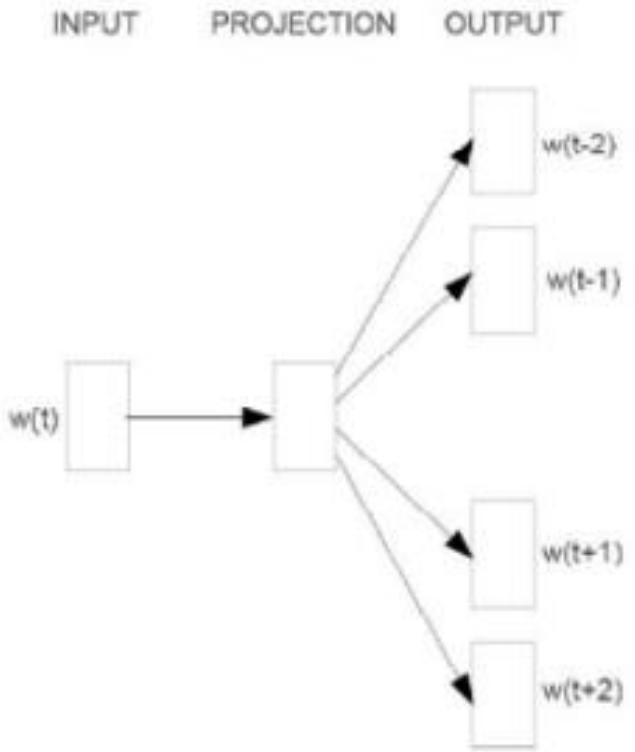


图 9 Skip-gram 模型

实践：onehot编码实现

- 使用sklearn包

实践：word2vec实现

- 使用gensim包训练word2vec
- 使用keras实现word2vec