

Word2Vec算法原理

任昭春

- 2013年，Google的Tomas Mikolov在《Efficient Estimation of Word Representation in Vector Space》提出 Word2Vec
- 词向量：
one hot representation，假设词典长度为N，则用一个长度为N的向量（一个位置为1，其他为0）
- {0,0,1,.....,0,0} 表示一个单词，维度灾难+无法刻画词与词的相似度
- Distributed Representation，固定长度的Dense Vector,如50、100、300维 {0.012,-0.009,0.113,...,0.032}
- Word2vec即为求出每个word的分布式表示，同时表征相似相关关系，与语义关系
与France相近的词

Word Cosine distance

spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
ruussia	0.571507
germany	0.563291
catalonia	0.534176

Interesting properties:

**vector('Paris') - vector('France') +
vector('Italy') \sim vector('Rome')**

**vector('king') - vector('man') +
vector('woman') \sim vector('queen')**

目录

- 预备知识：逻辑回归，softmax，霍夫曼树
- 基本模型：CBOW, Skip-gram
- 优化算法：Hierarchical Softmax，Negative Sampling
- Tricks （可以优化效率/性能，无严格证明）
- Vs fasttext
- 应用

逻辑回归 LR

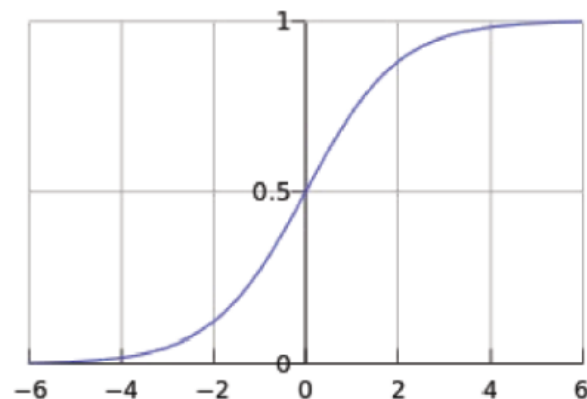
- 逻辑回归（二分类）

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

对 σ 求导, $\sigma' = \sigma(1 - \sigma)$

- Sigmoid函数
- 二分类问题: $y=1$ 为正类, $y=0$ 为负类
- Sigmoid函数的值 = 属于正类的概率
- 对于任意样本 $x = (x_1, x_2, \dots, x_n)$
- 二分类属于正类的概率为

$$h_{\theta}(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n),$$



- 即为
- 对于每条样本, 分类正确的概率即为 $P(y | x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y}$

$$L(\theta) = \prod_{i=1}^m P(y_i | x_i; \theta) = \prod_{i=1}^m (h_{\theta}(x_i))^{y_i} (1 - h_{\theta}(x_i))^{1-y_i}$$

- 似然函数为:
- 极大似然估计, 增大所有样本时似然函数的值
- 对数似然函数:

$$l(\theta) = \log L(\theta) = \sum_{i=1}^m (y_i \log h_{\theta}(x_i) + (1 - y_i) \log(1 - h_{\theta}(x_i)))$$

- 对数似然函数可以等价于损失函数（可以取一个负系数）
- 最大化对数似然函数 = 最小化损失函数
- 优化：梯度下降法

$$J(\theta) = -\frac{1}{m}l(\theta)$$

$$l(\theta) = \log L(\theta) = \sum_{i=1}^m (y_i \log h_{\theta}(x_i) + (1 - y_i) \log(1 - h_{\theta}(x_i)))$$

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta_{\theta_j}} J(\theta)$$

$$\frac{\delta}{\delta_{\theta_j}} J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left(y_i \frac{1}{h_{\theta}(x_i)} \frac{\delta}{\delta_{\theta_j}} h_{\theta}(x_i) - (1 - y_i) \frac{1}{1 - h_{\theta}(x_i)} \frac{\delta}{\delta_{\theta_j}} h_{\theta}(x_i) \right) \quad \text{对log求导}$$

$$= -\frac{1}{m} \sum_{i=1}^m \left(y_i \frac{1}{g(\theta^T x_i)} - (1 - y_i) \frac{1}{1 - g(\theta^T x_i)} \right) \frac{\delta}{\delta_{\theta_j}} g(\theta^T x_i)$$

$$= -\frac{1}{m} \sum_{i=1}^m \left(y_i \frac{1}{g(\theta^T x_i)} - (1 - y_i) \frac{1}{1 - g(\theta^T x_i)} \right) g(\theta^T x_i) (1 - g(\theta^T x_i)) \frac{\delta}{\delta_{\theta_j}} \theta^T x_i \quad \text{对}\sigma\text{求导, } \sigma' = \sigma(1 - \sigma)$$

$$= -\frac{1}{m} \sum_{i=1}^m (y_i (1 - g(\theta^T x_i)) - (1 - y_i) g(\theta^T x_i)) x_i^j$$

$$= -\frac{1}{m} \sum_{i=1}^m (y_i - g(\theta^T x_i)) x_i^j$$

$$= \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i) x_i^j$$

外部求导后结果代入括号内
消去， $\theta^T x_i$ 对 θ_j 求导即为
 x_{ij} (第i个样本的第j个特征
分量)

θ 更新过程可以写成：

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i) x_i^j$$

$h(x) - y$ 即为该条样本
预测值与真实值的差

softmax

- LR的多分类形式，设 x 为一条样本，维度为 n ， (x_1, x_2, \dots, x_n)
- 假设分为 $k=10$ 类（如MNIST数据集）

$$h_{\theta}(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1 | x^{(i)}; \theta) \\ p(y^{(i)} = 2 | x^{(i)}; \theta) \\ \vdots \\ p(y^{(i)} = k | x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^k e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix} \quad \theta = \begin{bmatrix} -\theta_1^T \\ -\theta_2^T \\ \vdots \\ -\theta_k^T \end{bmatrix}$$

- Softmax的参数即为一个 k 行 n 列的矩阵，参数个数为 $n*k$
- 每一行求 $e^{\theta^T x}$ 的值，最后做一个概率的归一

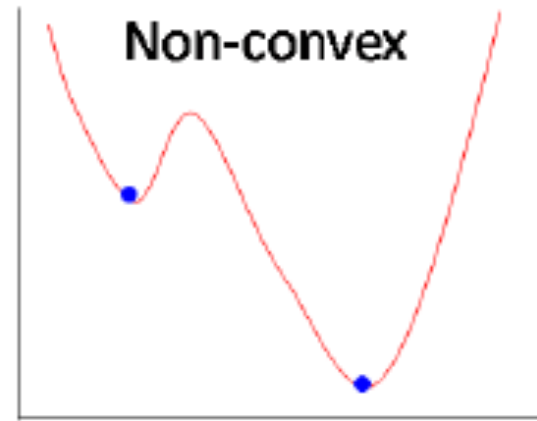
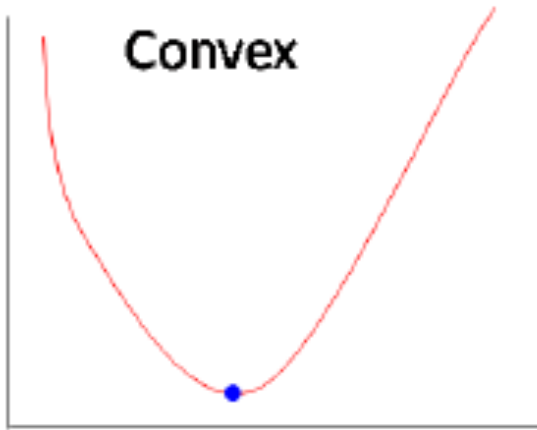
$$p(y^{(i)} = j | x^{(i)}; \theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}}$$

- Softmax每次优化所有 $n*k$ 个参数（加大正确分类的概率，降低错误分类的概率）
- 具体优化方法也是梯度下降

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^m [x^{(i)} (1\{y^{(i)} = j\} - p(y^{(i)} = j | x^{(i)}; \theta))]$$

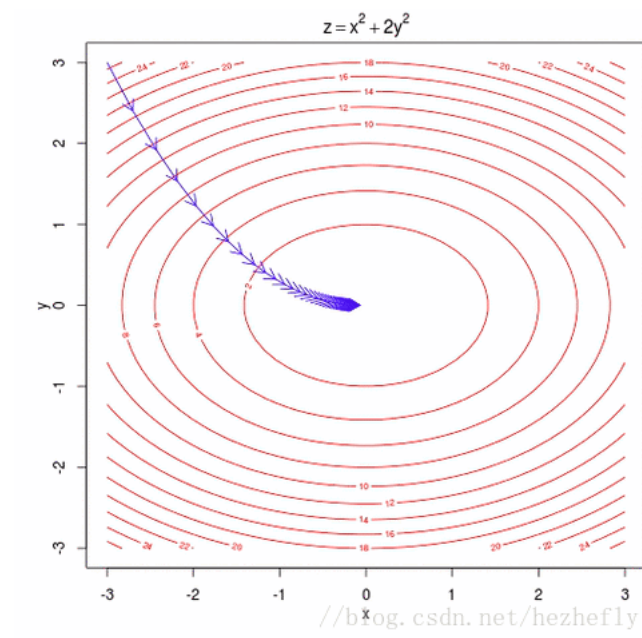
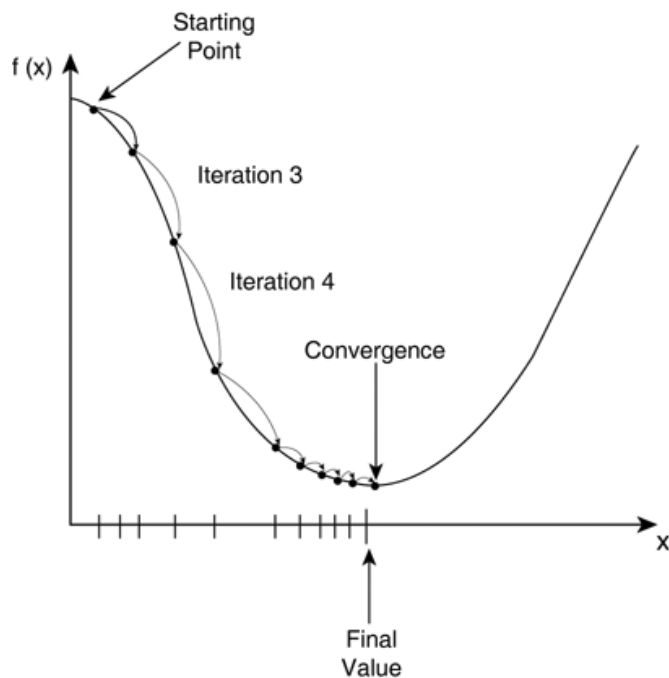
最优化问题

- 机器学习问题转化成为一个最优化问题



$$\min_{\mathbf{x}} f(\mathbf{x})$$

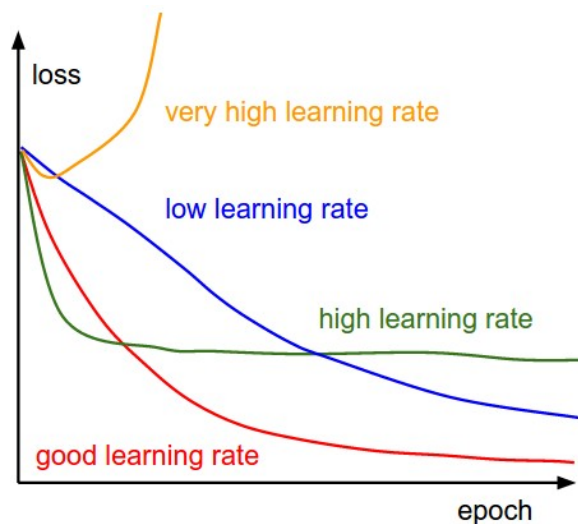
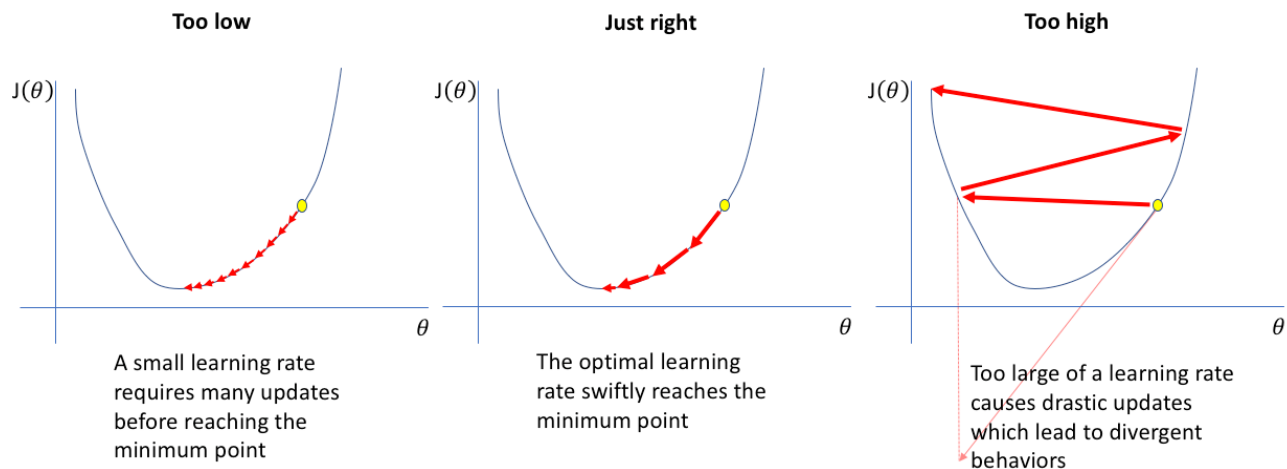
梯度下降法 (Gradient Descent)



$$\begin{aligned}\theta_{t+1} &= \theta_t - \alpha \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t} \\ &= \theta_t - \alpha \frac{1}{N} \sum_{i=1}^N \frac{\partial \mathcal{L}(\theta_t; x^{(i)}, y^{(i)})}{\partial \theta}.\end{aligned}$$

搜索步长 α 中也叫作学习率 (Learning Rate)

学习率是十分重要的超参数！



随机梯度下降法

- 随机梯度下降法 (Stochastic Gradient Descent, SGD) 也叫增量梯度下降, 每个样本都进行更新

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}(\theta_t; x^{(t)}, y^{(t)})}{\partial \theta},$$

- 小批量 (Mini-Batch) 随机梯度下降法

随机梯度下降法

算法 2.1: 随机梯度下降法

输入: 训练集 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$, 验证集 \mathcal{V} , 学习率 α

1 随机初始化 θ ;

2 repeat

3 对训练集 \mathcal{D} 中的样本随机重排序;

4 for $n = 1 \cdots N$ do

5 从训练集 \mathcal{D} 中选取样本 $(\mathbf{x}^{(n)}, y^{(n)})$;

 // 更新参数

6 $\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}(\theta; x^{(n)}, y^{(n)})}{\partial \theta}$;

7 end

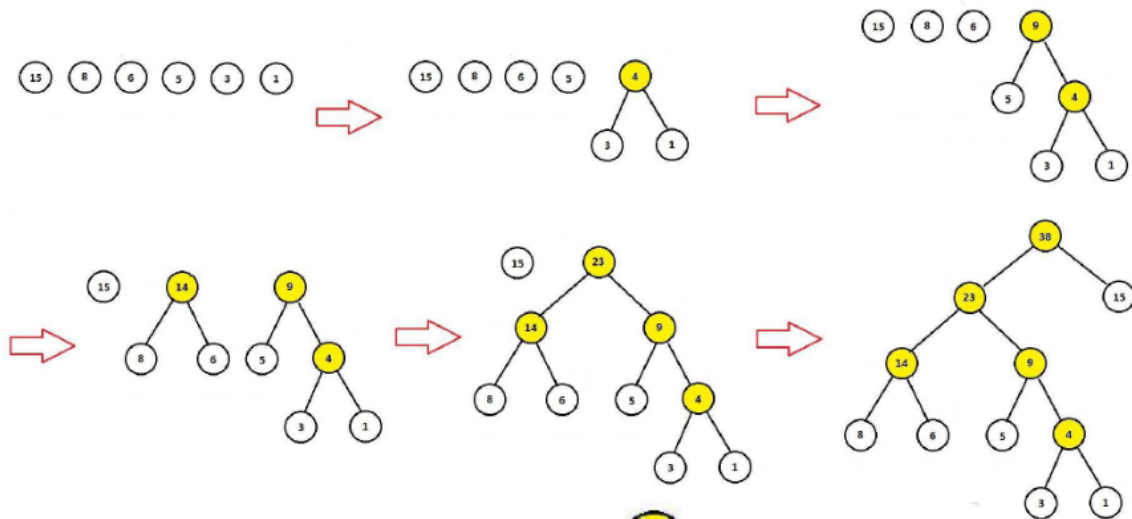
8 until 模型 $f(\mathbf{x}; \theta)$ 在验证集 \mathcal{V} 上的错误率不再下降;

输出: θ



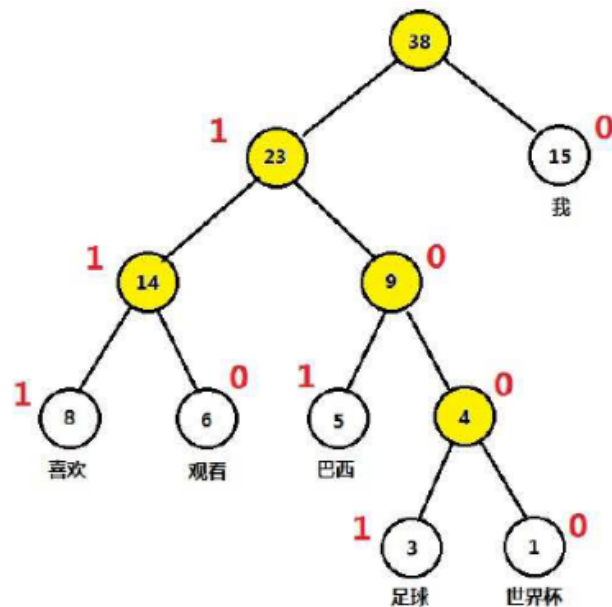
Huffman树

- 二叉树，构造
- 词频越大，
距离根节点越近



- Huffman编码:
- 1. 频率高码长短，
频率低码长高，减少报
- 2. Huffman构造的编码都为前缀
编码（一个字符不可能是另一个
字符的前缀，译码没有歧义）

- 我爱巴西足球世界杯
- 0,111,110,101,1001,1000



- Word2vec的两个基本模型，CBOW与skip-gram，都为输入/投影/输出三层

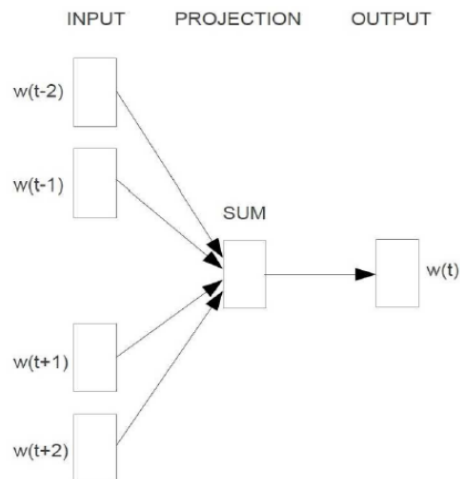


图 8 CBOW 模型

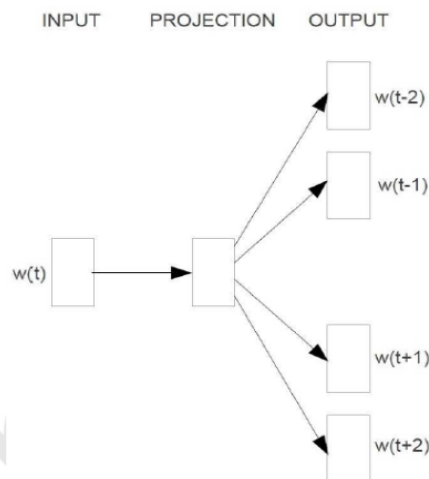
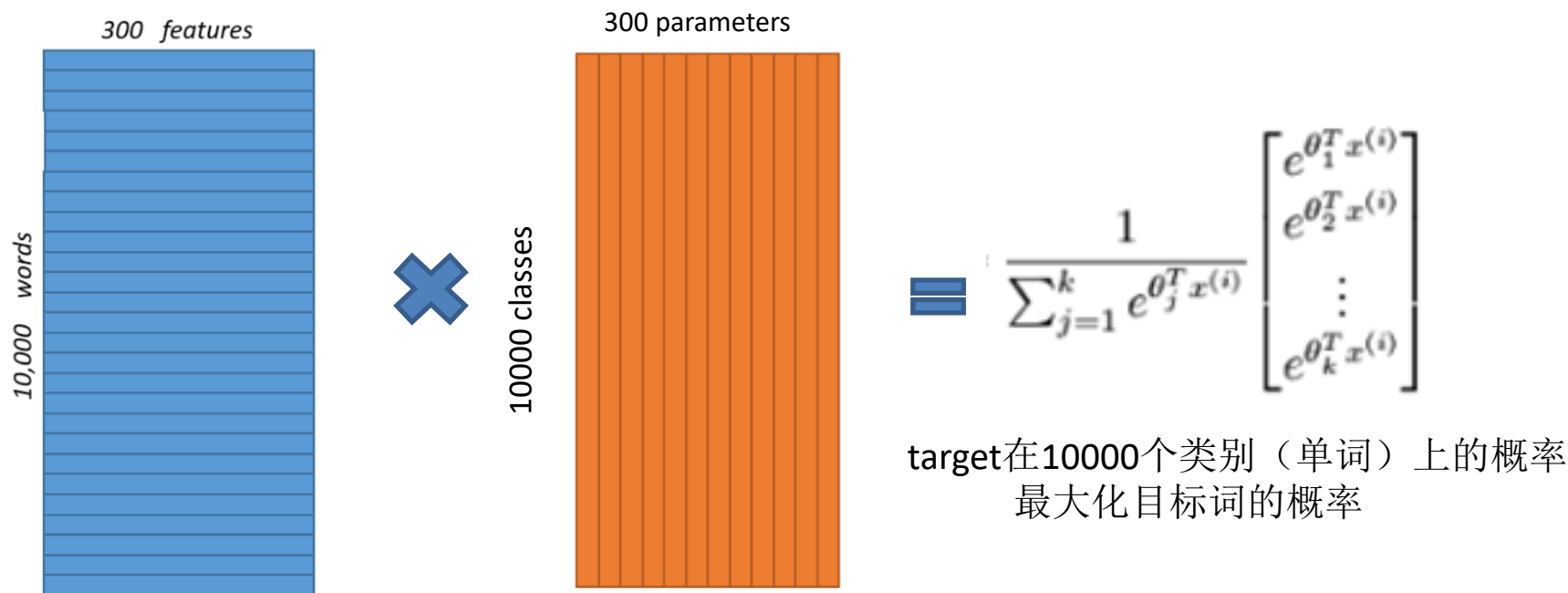


图 9 Skip-gram 模型

- A quick brown fox jump over the lazy dog
- Cbow 用上下文词的**和**来预测中心词 (quick,brown,jump,over) \xrightarrow{sum} fox
- Skip-gram用每个中心词来预测上下文的**每个**词 (fox,quick),(fox,brown),(fox,jump),(fox,over)
- Word2vec的两个基本优化算法：
- HS(Hierarchical Softmax) 利用Huffman树进行层次softmax
- Negative Sampling 负采样构建正负样本

Word2vec真实在做什么？

- 拿skip-gram举例： (fox,quick),(fox,brown),(fox,jump),(fox,over)
- 10000个词，10000分类，softmax
- 拿fox的样本（即fox的词向量）做一个10000分类预测quick



每一行，300维 词向量（特征） * 300 维对应分类H矩阵参数 => sigmoid归一(softmax)
每一个pair对即为一条训练样本，使用softmax每次需更新 $10000 * 300 = 3M$ 个参数！

极难训练，层次softmax (一种softmax的近似算法) 和负采样算法被用于word2vec

基于Hierarchical Softmax +CBOW的算法

- 根据语料的词频生成一颗huffman树，如我爱观看巴西足球世界杯

- 我：15，喜欢：8，观看：6
- 由这颗huffman数，如何定义条件概率函数 $p(w|\text{context}(w))$
- Huffman树提前建立，编码0,1，
对于一个特定target叶子节点，根据其huffman编码，有一条唯一的路径到达

- 从根节点到足球，经过4次分支
相当于4次二分类

- 左:正类 右: 负类

- 正类 $\sigma(\mathbf{x}_w^\top \theta) = \frac{1}{1 + e^{-\mathbf{x}_w^\top \theta}},$

- 负类 $1 - \sigma(\mathbf{x}_w^\top \theta),$

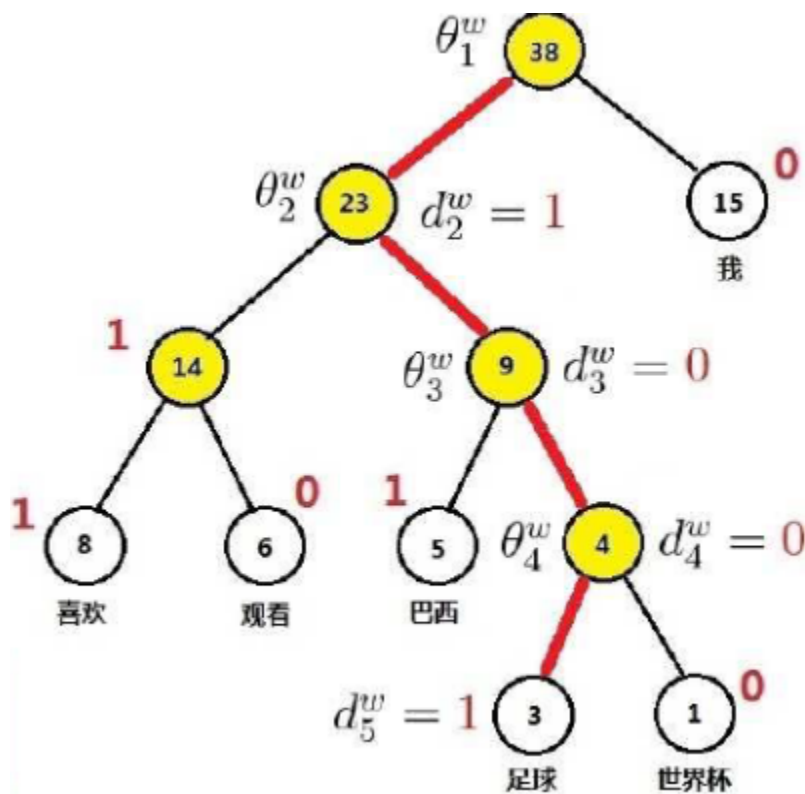
1. 第 1 次: $p(d_2^w | \mathbf{x}_w, \theta_1^w) = 1 - \sigma(\mathbf{x}_w^\top \theta_1^w);$

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: $p(d_5^w | \mathbf{x}_w, \theta_4^w) = 1 - \sigma(\mathbf{x}_w^\top \theta_4^w),$

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



- 得到对数似然函数，要优化的损失的函数：

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

- 在每一层，实际是一个已知 \mathbf{x} 对 θ 进行优化的logistics regression，使用梯度下降

首先考虑 $\mathcal{L}(w, j)$ 关于 θ_{j-1}^w 的梯度计算.

$$\begin{aligned}\frac{\partial \mathcal{L}(w, j)}{\partial \theta_{j-1}^w} &= \frac{\partial}{\partial \theta_{j-1}^w} \{ (1 - d_j^w) \cdot \log[\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] + d_j^w \cdot \log[1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] \} \\ &= (1 - d_j^w)[1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]\mathbf{x}_w - d_j^w \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)\mathbf{x}_w \quad (\text{利用 (2.1) 式}) \\ &= \{ (1 - d_j^w)[1 - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] - d_j^w \sigma(\mathbf{x}_w^\top \theta_{j-1}^w) \} \mathbf{x}_w \quad (\text{合并}) \\ &= [1 - d_j^w - \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)] \mathbf{x}_w.\end{aligned}$$

于是, θ_{j-1}^w 的更新公式可写为

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

- 区别：需更新 X_w ，LR里 X_w 是特征，不需要更新
word2vec里 X_w 是词向量，需要更新，可以认为在目前huffman树的参数下，怎么调整输入词的词向量可以更大减少loss

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

- HS+CBOW模型的伪代码
- 每一层更新非叶子节点辅助参数
- 到叶子节点后更新词向量

```

1.  $\mathbf{e} = \mathbf{0}$ .
2.  $\mathbf{x}_w = \sum_{u \in \text{Context}(w)} \mathbf{v}(u)$ .
3. FOR  $j = 2 : l^w$  DO
    {
        3.1  $q = \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)$ 
        3.2  $g = \eta(1 - d_j^w - q)$ 
        3.3  $\mathbf{e} := \mathbf{e} + g\theta_{j-1}^w$ 
        3.4  $\theta_{j-1}^w := \theta_{j-1}^w + g\mathbf{x}_w$ 
    }
4. FOR  $u \in \text{Context}(w)$  DO
    {
         $\mathbf{v}(u) := \mathbf{v}(u) + \mathbf{e}$ 
    }

```

基于Hierarchical Softmax + skip-gram的算法

- 与CBOW的区别
中心词->每个上下文词
- Projection为一个恒等投影

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

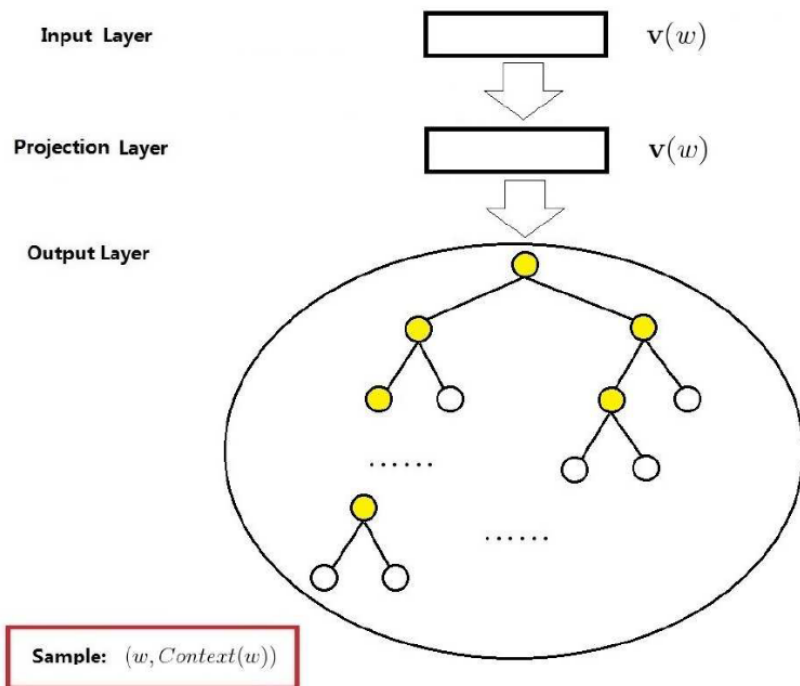
- 按照hs的思想:

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

每个词都走一遍根到叶子节点的N个二分类

- 每一层的logistics概率

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



- 同样地，极大似然估计，loss为：

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

- 之后先对 θ 求导，再对输入中心词向量 $\mathbf{v}(w)$ 求导

论文的伪代码，上下文所有词计算完后再更新词向量


```
e = 0
FOR u ∈ Context(w) DO
{
  FOR j = 2 : lu DO
  {
    1. q = σ(v(w)⊤θj-1u)
    2. g = η(1 - dju - q)
    3. e := e + gθj-1u
    4. θj-1u := θj-1u + g v(w)
  }
}
v(w) := v(w) + e
```



实际实现时的伪代码，每个pair单独算一次loss，更新一次参数

```
FOR u ∈ Context(w) DO
{
  e = 0
  FOR j = 2 : lu DO
  {
    1. q = σ(v(w)⊤θj-1u)
    2. g = η(1 - dju - q)
    3. e := e + gθj-1u
    4. θj-1u := θj-1u + g v(w)
  }
  v(w) := v(w) + e
}
```

➤ 问题：为什么比较相近的词会有相同的词向量？

- 共享上下文（前后购买）：订单session体现了相关
尿不湿，坚果，小米手机，手机贴膜，手机壳（横向关系）

- 同为中心词（上下文相同或相似）：订单session体现了相似
手机壳，贴膜，小米手机，充电宝，数据线（纵向关系）



手机壳，贴膜，华为手机，充电宝，数据线

- 拿skip-gram举例：
- 1中，两词共享了很多上下文，对于每个pair对如（小米手机，手机壳）（手机贴膜，手机壳），在Huffman树中要达到相同的叶子节点，降低loss的方法就是将两个word的词向量变的更为接近。
- 2中，同为中心词。上下文完全一样，每个pair的target也一致，因此也会使两个词向量更加接近。

纵向> 横向

If two different words have very similar “contexts” (that is, what words are likely to appear around them), then our model needs to output very similar results for these two words. And one way for the network to output similar context predictions for these two words is if *the word vectors are similar*. So, if two words have similar contexts, then our network is motivated to learn similar word vectors for these two words! Ta da!

考虑下面两个句子：

- 爱因斯坦 是 伟大 的 物理学家
- 霍金 是 伟大 的 物理学家
- Word2vec中，更多建模了纵向关系
（爱因斯坦，霍金） 大于 （爱因斯坦，物理学家）（频繁项集， n-gram）

层次softmax VS softmax 复杂度分析

- 对于N个word，词向量为k维，softmax要做N分类
- 每次需更新 $N*k$ 个参数
- 对于N个word，建立一颗有N个叶子节点的Huffman n叉树，有 $(N-1) / (n-1) \approx N$ 个非叶子节点
- 则最多做N次Logistic Regression

Approach	Speed-up factor	During training?	During testing?	Performance (small vocab)	Performance (large vocab)	Proportion of parameters
Softmax	1x	-	-	very good	very poor	100%
Hierarchical Softmax	25x (50-100x)	X	-	very poor	very good	100%
Differentiated Softmax	2x	X	X	very good	very good	< 100%

Negative Sampling优化算法

- 与HS相比，不再使用复杂的Huffman树，而是利用简单的随机负采样，能大幅提高训练速度
- **CBOW**: 对于指定的上下文 $\text{context}(w)$ ，中心词 w 是正样本，除了正样本的其他词为负样本，就需要采样（**Negative Sampling**）

假定现在已经选好了一个关于 w 的负样本子集 $NEG(w) \neq \emptyset$. 且对 $\forall \tilde{w} \in \mathcal{D}$, 定义

$$L^w(\tilde{w}) = \begin{cases} 1, & \tilde{w} = w; \\ 0, & \tilde{w} \neq w, \end{cases}$$

对于一个给定的正样本 $(\text{Context}(w), w)$, 我们希望最大化

$$g(w) = \prod_{u \in \{w\} \cup NEG(w)} p(u | \text{Context}(w)),$$

其中

$$p(u | \text{Context}(w)) = \begin{cases} \sigma(\mathbf{x}_w^\top \theta^u), & L^w(u) = 1; \\ 1 - \sigma(\mathbf{x}_w^\top \theta^u), & L^w(u) = 0, \end{cases}$$

或者写成整体表达式

$$p(u | \text{Context}(w)) = [\sigma(\mathbf{x}_w^\top \theta^u)]^{L^w(u)} \cdot [1 - \sigma(\mathbf{x}_w^\top \theta^u)]^{1-L^w(u)},$$

对每个词来说， θ 是一个辅助向量，待训练参数

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

- 正负样本都有了，要做的就是优化 θ^u 和 \mathbf{x}_w ，本质上和LR没有区别，只是先对 θ^u 求导优化，再对 \mathbf{x}_w 求导优化

$$\begin{aligned} \frac{\partial \mathcal{L}(w, u)}{\partial \theta^u} &= \frac{\partial}{\partial \theta^u} \{L^w(u) \cdot \log [\sigma(\mathbf{x}_w^\top \theta^u)] + [1 - L^w(u)] \cdot \log [1 - \sigma(\mathbf{x}_w^\top \theta^u)]\} \\ &= L^w(u)[1 - \sigma(\mathbf{x}_w^\top \theta^u)]\mathbf{x}_w - [1 - L^w(u)]\sigma(\mathbf{x}_w^\top \theta^u)\mathbf{x}_w \quad (\text{利用 (2.1) 式}) \\ &= \{L^w(u)[1 - \sigma(\mathbf{x}_w^\top \theta^u)] - [1 - L^w(u)]\sigma(\mathbf{x}_w^\top \theta^u)\}\mathbf{x}_w \quad (\text{合并}) \\ &= [L^w(u) - \sigma(\mathbf{x}_w^\top \theta^u)]\mathbf{x}_w. \end{aligned}$$

于是, θ^u 的更新公式可写为

$$\theta^u := \theta^u + \eta [L^w(u) - \sigma(\mathbf{x}_w^\top \theta^u)] \mathbf{x}_w.$$

接下来考虑 $\mathcal{L}(w, u)$ 关于 \mathbf{x}_w 的梯度. 同样利用 $\mathcal{L}(w, u)$ 中 \mathbf{x}_w 和 θ^u 的对称性, 有

$$\frac{\partial \mathcal{L}(w, u)}{\partial \mathbf{x}_w} = [L^w(u) - \sigma(\mathbf{x}_w^\top \theta^u)] \theta^u.$$

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

- NS+CBOW使用随机梯度上升的伪代码

```

1.  $\mathbf{e} = 0.$ 
2.  $\mathbf{x}_w = \sum_{u \in \text{Context}(w)} \mathbf{v}(u).$ 
3. FOR  $u = \{w\} \cup \text{NEG}(w)$  DO
  {
    3.1  $q = \sigma(\mathbf{x}_w^\top \theta^u)$ 
    3.2  $g = \eta(L^w(u) - q)$ 
    3.3  $\mathbf{e} := \mathbf{e} + g\theta^u$ 
    3.4  $\theta^u := \theta^u + g\mathbf{x}_w$ 
  }
4. FOR  $u \in \text{Context}(w)$  DO
  {
     $\mathbf{v}(u) := \mathbf{v}(u) + \mathbf{e}$ 
  }

```

```

FOR  $\tilde{w} = \text{Context}(w)$  DO
{
   $\mathbf{e} = 0.$ 
  FOR  $u = \{w\} \cup \text{NEG}^{\tilde{w}}(w)$  DO
  {
     $q = \sigma(\mathbf{v}(\tilde{w})^\top \theta^u)$ 
     $g = \eta(L^w(u) - q)$ 
     $\mathbf{e} := \mathbf{e} + g\theta^u$ 
     $\theta^u := \theta^u + g\mathbf{v}(\tilde{w})$ 
  }
   $\mathbf{v}(\tilde{w}) := \mathbf{v}(\tilde{w}) + \mathbf{e}$ 
}

```

- Skip-gram算法类似，且在具体实现中将context(w)拆成了一个一个的词单独考虑，采样 $\text{len}(\text{context}(w))$ 个负样本，与 $\text{len}(\text{context}(w))$ 个正样本共同训练一轮，更新每个候选集的辅助向量与中心词的词向量

负采样 VS softmax 复杂度分析

- 对于N个word，词向量为k维，softmax要做N分类
- 每次需更新 $N*k$ 个参数
- 对于N个word，window = 5
- 假设skip-gram算法，则前后共10个词，负采样10个词，问题降级为每次做一个20分类的softmax
- 或降级为每次做20次LR的二分类
- 速度能提升百倍以上

Tricks （提高性能or 提高速度）

1. 负采样的实现（带权采样）：

设词典 \mathcal{D} 中的每一个词 w 对应一个线段 $l(w)$, 长度为

$$len(w) = \frac{\text{counter}(w)}{\sum_{u \in \mathcal{D}} \text{counter}(u)}, \quad (5.16)$$

这里 $\text{counter}(\cdot)$ 表示一个词在语料 \mathcal{C} 中出现的次数 (分母中的求和项用来做归一化). 现在将这些线段首尾相连地拼接在一起, 形成一个长度为 1 的单位线段. 如果随机地往这个单位线段上打点, 则其中长度越长的线段 (对应高频词) 被打中的概率就越大.

2. $\sigma(x)$ sigmoid函数的近似计算，类似查表法，预先分段计算好

3. 低频词：默认出现次数<5的词不进入词典中

4. 高频词: subsampling，采用公式决定保留这个word:

➤ $z(w)$ 为一个词在所有词中的比例

➤ 比如 $z(w_i)=0.00746$, $P(w_i)=0.5P(w_i)=0.5$

➤ 50%的概率保留这个pair对

➤ 提高2~10的训练速度，同时提升低频词的向量精度(间接提升权重)

$$P(w_i) = \left(\sqrt{\frac{z(w_i)}{0.001}} + 1 \right) \cdot \frac{0.001}{z(w_i)}$$

Tricks(2)

- 窗口windows大小n: 中心词前后各取n个词（源码中是每次取一个1~n的随机数k，中心词前后各取k个词）
- 自适应学习率，默认初始学习率为0.025，每处理完10000个词，降低一次学习率，最低学习率为1e-4

$$\eta = \eta_0 \left(1 - \frac{\text{word_count_actual}}{\text{train_words} + 1} \right),$$

- 参数初始化与训练:

word2vec都采用随机梯度下降（每个词或每组词就更新一次模型），且只对模型遍历一次

➤ 逻辑回归对应的参数 θ (huffman树的非叶子节点参数/ns中的每个词的辅助向量): 零初始化

➤ 词向量的初始化 $v(x)$: 随机初始化 $\frac{[\text{rand()}/\text{RAND_MAX}] - 0.5}{m}$,

- 多线程or分布式:

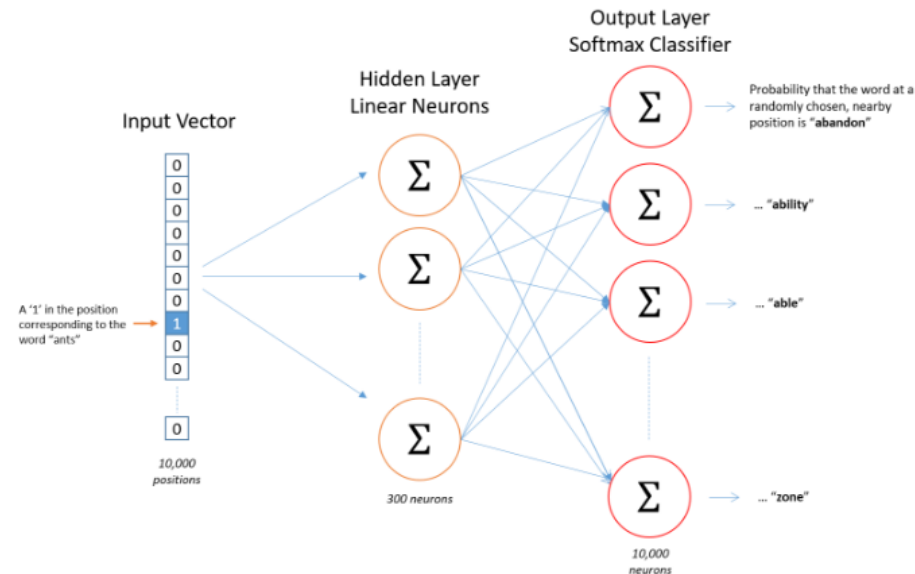
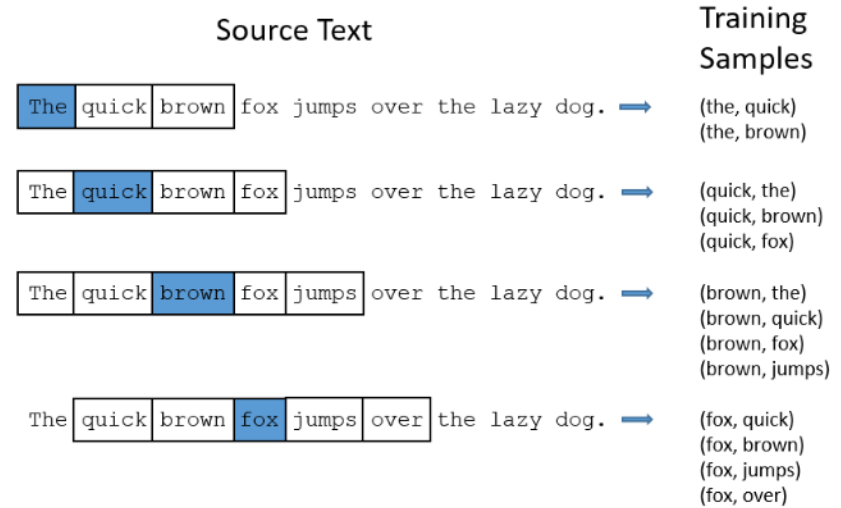
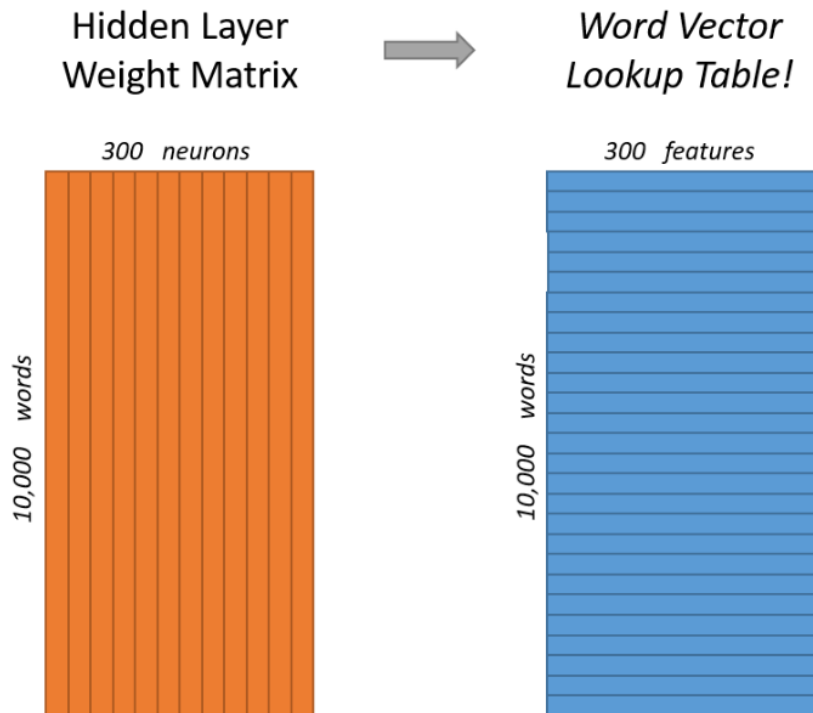
速度很快，多线程足以，在我们堡垒机num_thread=16

Parameter tuning

- **Performance**
- 训练速度 The training speed can be significantly improved by using parallel training on multiple-CPU machine (use the switch '-threads N'). The hyper-parameter choice is crucial for performance (both speed and accuracy), however varies for different applications. The main choices to make are:
 - 模型选择 architecture: skip-gram (slower, better for infrequent words) vs CBOW (fast)
 - 训练算法 the training algorithm: hierarchical softmax (better for infrequent words) vs negative sampling (better for frequent words, better with low dimensional vectors)
 - sub-sampling of frequent words: can improve both accuracy and speed for large data sets (useful values are in range $1e-3$ to $1e-5$)
 - dimensionality of the word vectors: usually more is better, but not always
 - context (window) size: for skip-gram usually around 10, for CBOW around 5
- Command line: 做聚类
- `./word2vec -train corpus -output cluster -cbow 0 -size 200 -window 5 -negative 0 -hs 1 -sample $1e-3$ -threads 12 -classes 100`
- `sort cluster -k 2 -n > classes.sorted.txt`

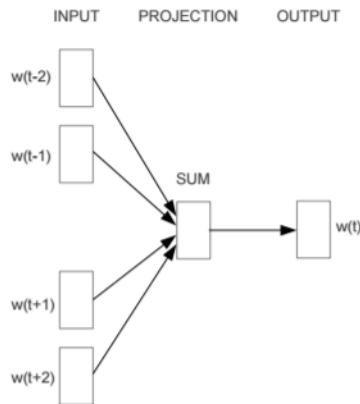
神经网络理解word2vec

- Word2vec , 深度学习?
- 浅层神经网络
- A fake task -> transfer learning
- Suppose a vocab of 10000 uniq words
- One hot vector for input
- One hot vector for output(in fact, a probability distribution)

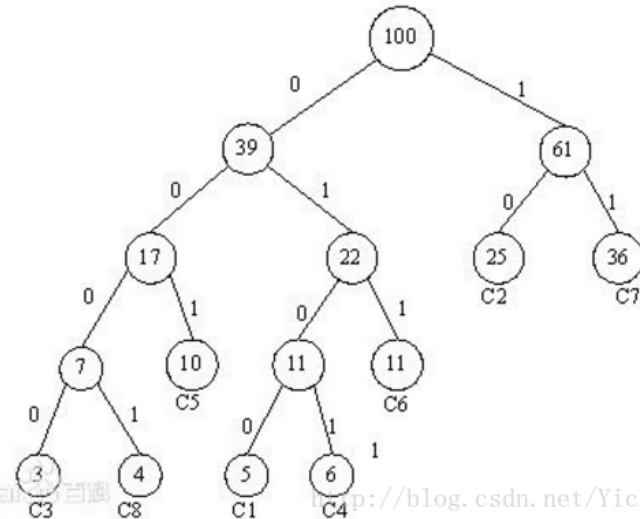


Vs fasttext

- 同作者Tomas Mikolov (Facebook Research) 与2016发表
- Fasttext 可做有监督，多分类
- 语料格式为： word1,word2...word_n __label__pos （比如情感分析）
word1,word2...word_n __label__neg



http://blog.csdn.net/Yick_Liao



- word2vec 上下文预测中心词
- Fasttext 语料里所有word预测label
- 输入：所有word中vector的平均 输出：Huffman树的叶子节点为label类别
- 具体分类时：fasttext遍历所有huffman树所有叶子节点，求出属于每一类的概率

- 适合大型数据+高效的训练速度：
- 在使用标准多核CPU的情况下10分钟内处理超过10亿个词汇”
- 特别是与深度模型对比，fastText能将训练时间由数天缩短到几秒钟。
- 使用一个标准多核 CPU，得到了在10分钟内训练完超过10亿词汇量模型的结果。
- fastText还能在五分钟内将50万个句子分成超过30万个类别

文本分类，情感分析，标签预测公开 数据集

	Yahoo		Amazon full		Amazon polarity	
	Accuracy	Time	Accuracy	Time	Accuracy	Time
char-CNN	71.2	1 day	59.5	5 days	94.5	5 days
VDCNN	73.4	2h	63	7h	95.7	7h
fastText	72.3	5s	60.2	9s	94.6	10s

- 测试了一下拿 商品标题的分词做分类(label = 三级类目) 抽取数据
- 取了20W sku， 20类别， accuracy = 0.993

应用

- 场景化：
比如刚搬新家要买做饭的场景：各种油，米，醋，炒锅，铲子，菜板，电饭煲，微波炉等
订单产品词session语料训练 -> 产品词的向量 -> 向量聚类
- 相似相关，品牌类目：
订单品牌类目session训练 -> 类目/品牌的向量 -> cos距离
- 商品内容相似度冷启召回：
标题分词计算 -> word向量 -> 标题向量（或sent2vec） -> cos距离
- 寻找相似词：如纸尿裤与尿不湿相似（通过标题）
之前使用lucene检索时候用于TF-IDF的降权，增加多样性

宝宝护理大全	宝宝	婴儿	儿童	宝宝护理大全	宝宝^0.166666666667	婴儿
	新生儿	纯棉	夏季	^0.229166666667	儿童^0.211538461538	新生
	肚兜	透气	凉席	儿^0.261904761905	纯棉^0.733333333333	夏季
	湿巾	防水	浴巾	^0.6875	肚兜^0.647058823529	透气
	尿不湿	纸尿裤	小孩	^0.611111111111	凉席^0.578947368421	湿巾
	隔尿垫	超薄	婴幼儿	^0.55	防水^0.52380952381	尿不
	尿布	加大		湿^0.239130434783	纸尿裤^0.229166666667	小孩
				^0.146666666667	隔尿垫^0.211538461538	超薄
				^0.407407407407	婴幼儿^0.0982142857143	尿布
				^0.189655172414	加大^0.366666666667	