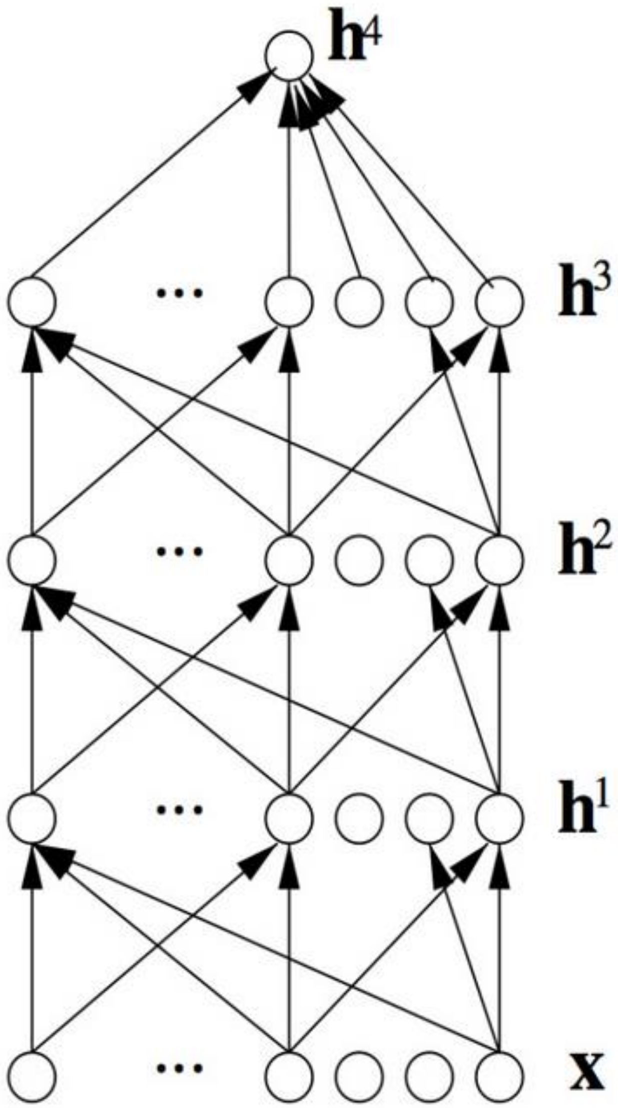
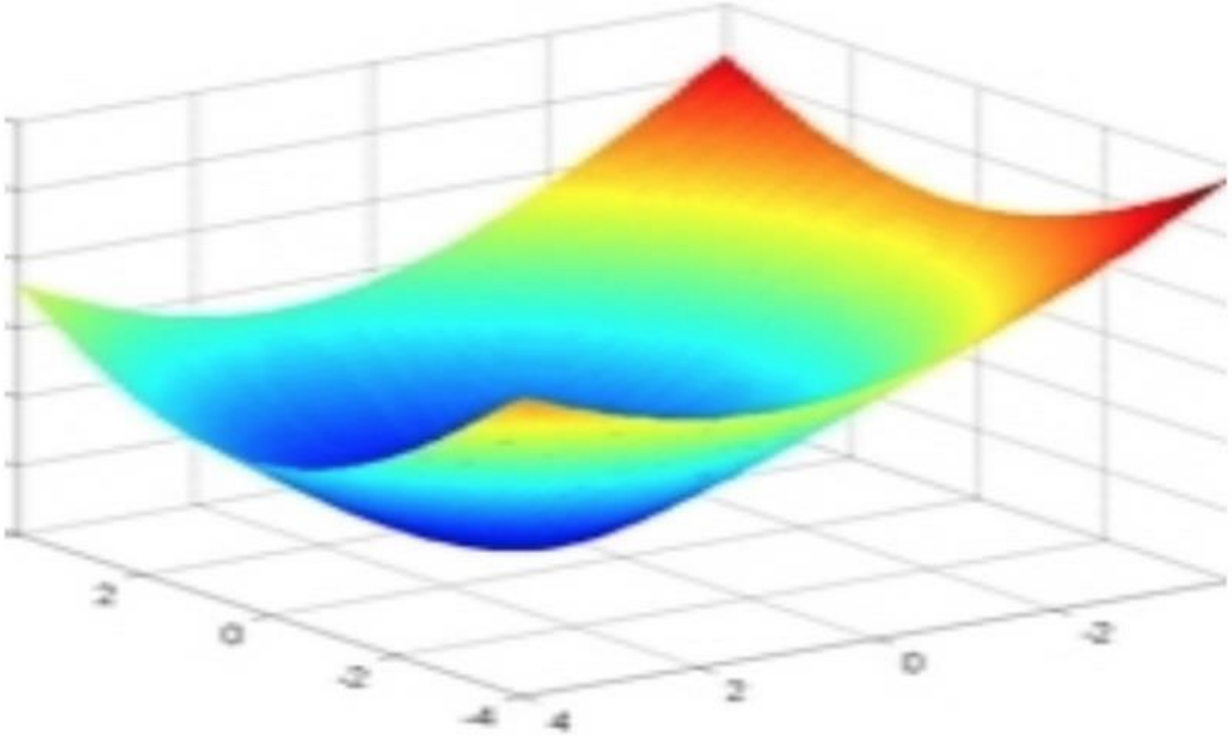
自然语言处理

- 拼写检查、关键词检索……
- 文本挖掘（产品价格、日期、时间、地点、人名、公司名）
- 文本分类
- 机器翻译
- 客服系统
- 复杂对话系统

深度学习





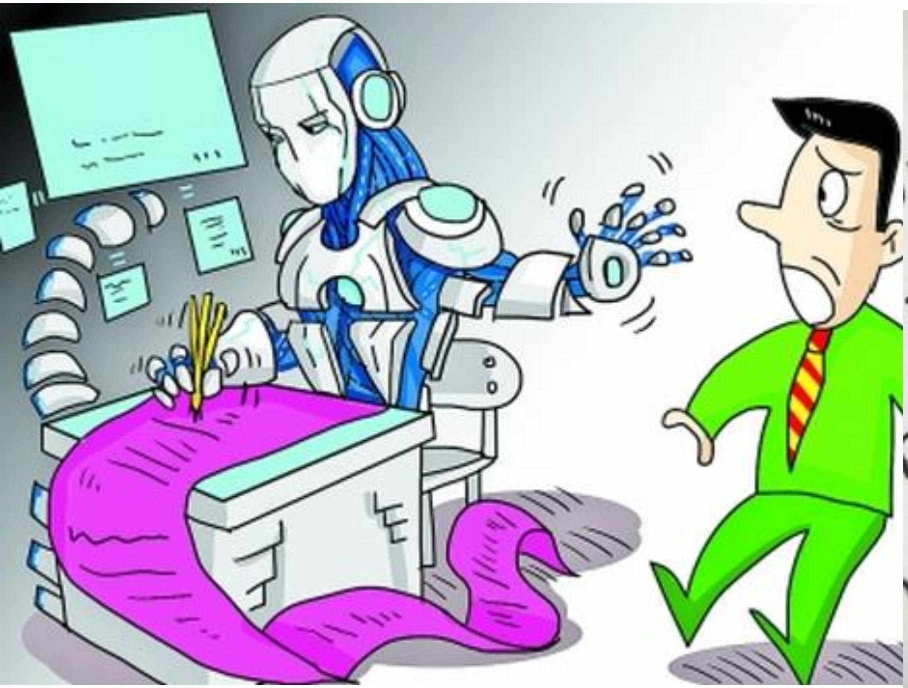
深度学习

为什么需要研究深度学习

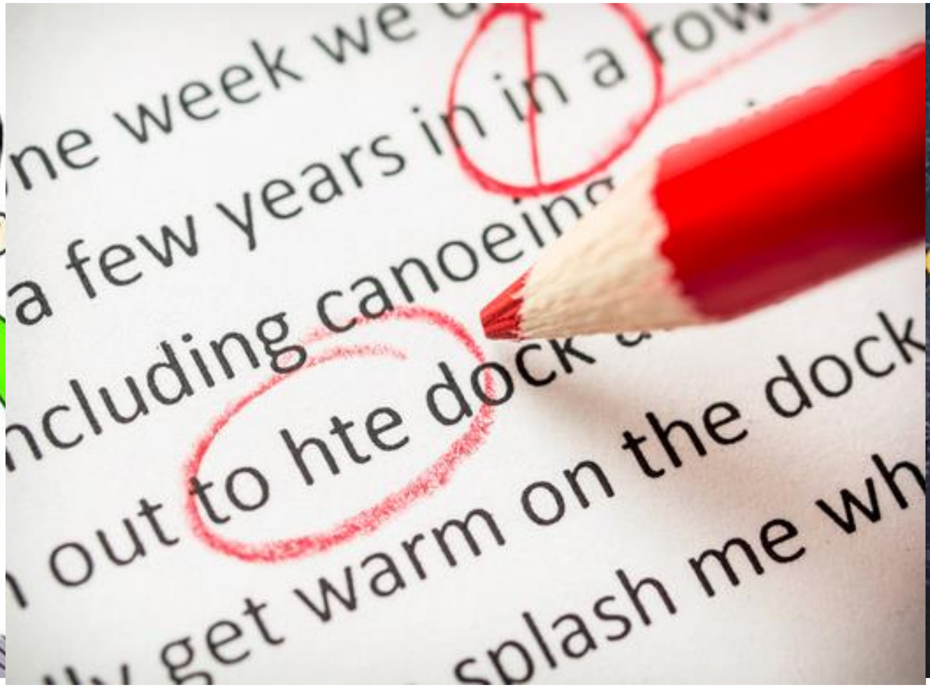
- 手工特征耗时耗力，还不易拓展
- 自动特征学习快，方便拓展
- 深度学习提供了一种通用的学习框架，可用来表示世界、视觉和语言学信息
- 深度学习既可以无监督学习，也可以监督学习

语言模型

机器翻译



拼写纠错

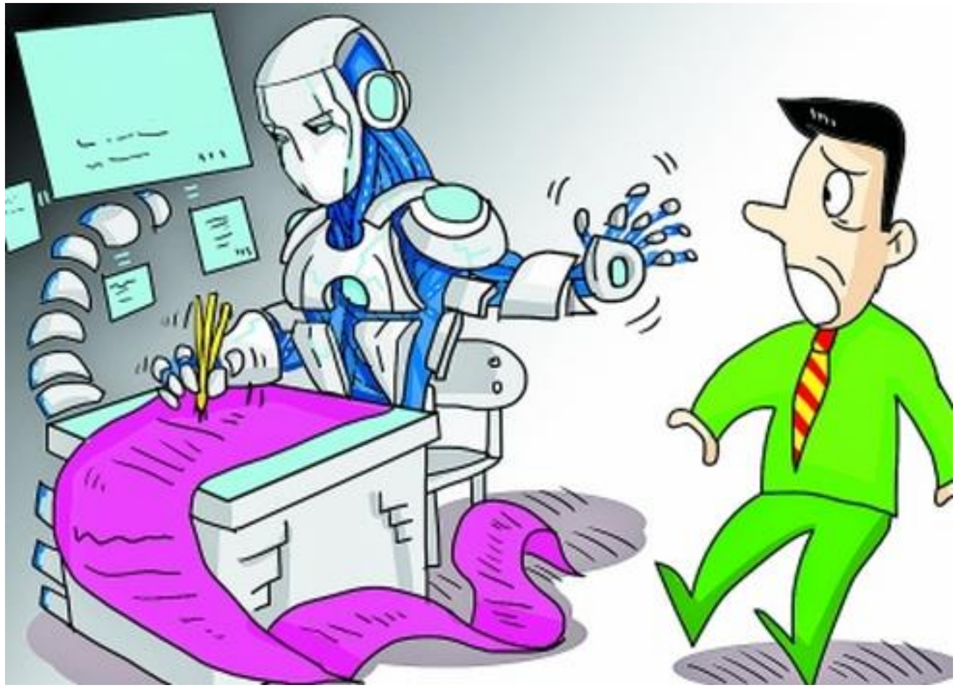


智能问答

语言模型

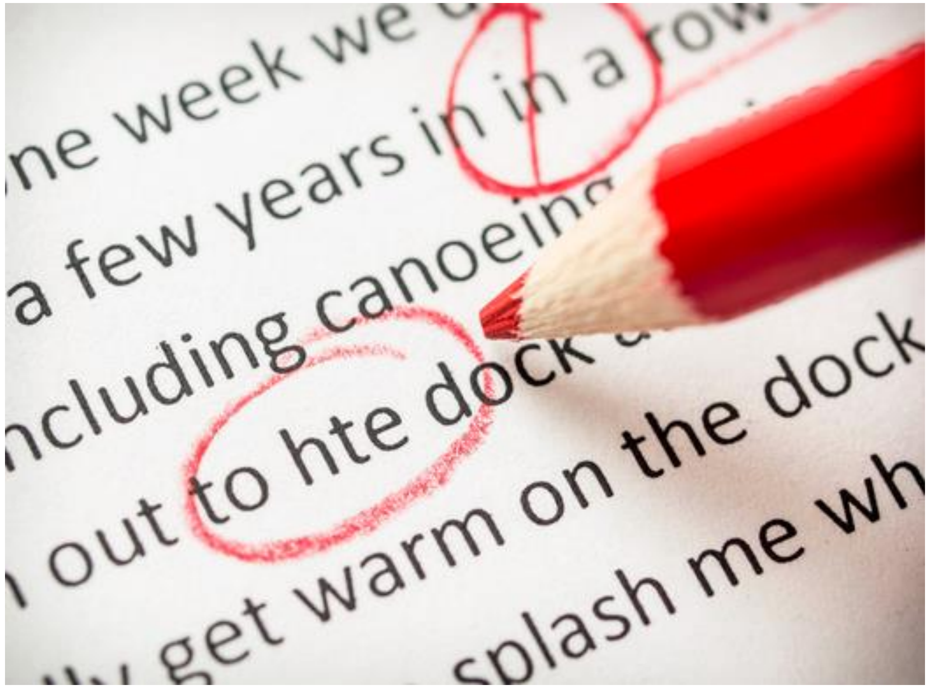
机器翻译

P(**high** price) > P(**large** price)



拼写纠错

P(about fifteen **minutes** from) > P(about fifteen**minuets** from)



语言模型

我 今天 下午 打 篮球

$$p(S)=p(w1,w2,w3,w4,w5,\ldots,wn)$$
$$=p(w1)p(w2|w1)p(w3|w1,w2)\ldots p(wn|w1,w2,\ldots,wn-1)$$

p(S)被称为语言模型，即用来计算一个句子概率的模型

语言模型

p(wi|w1,w2,...,wi-1) = p(w1,w2,...,wi-1,wi) / p(w1,w2,...,wi-1)

1.数据过于稀疏

2.参数空间太大

语言模型

假设下一个词的出现依赖它前面的一个词：

p(S)=p(w1)p(w2|w1)p(w3|w1,w2)...p(wn|w1,w2,...,wn-1)

=p(w1)p(w2|w1)p(w3|w2)...p(wn|wn-1)

假设下一个词的出现依赖它前面的两个词：

p(S)=p(w1)p(w2|w1)p(w3|w1,w2)...p(wn|w1,w2,...,wn-1)

=p(w1)p(w2|w1)p(w3|w1,w2)...p(wn|wn-1,wn-2) c(wi)如下:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

c(wi-1,wi)如下:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

I want english food

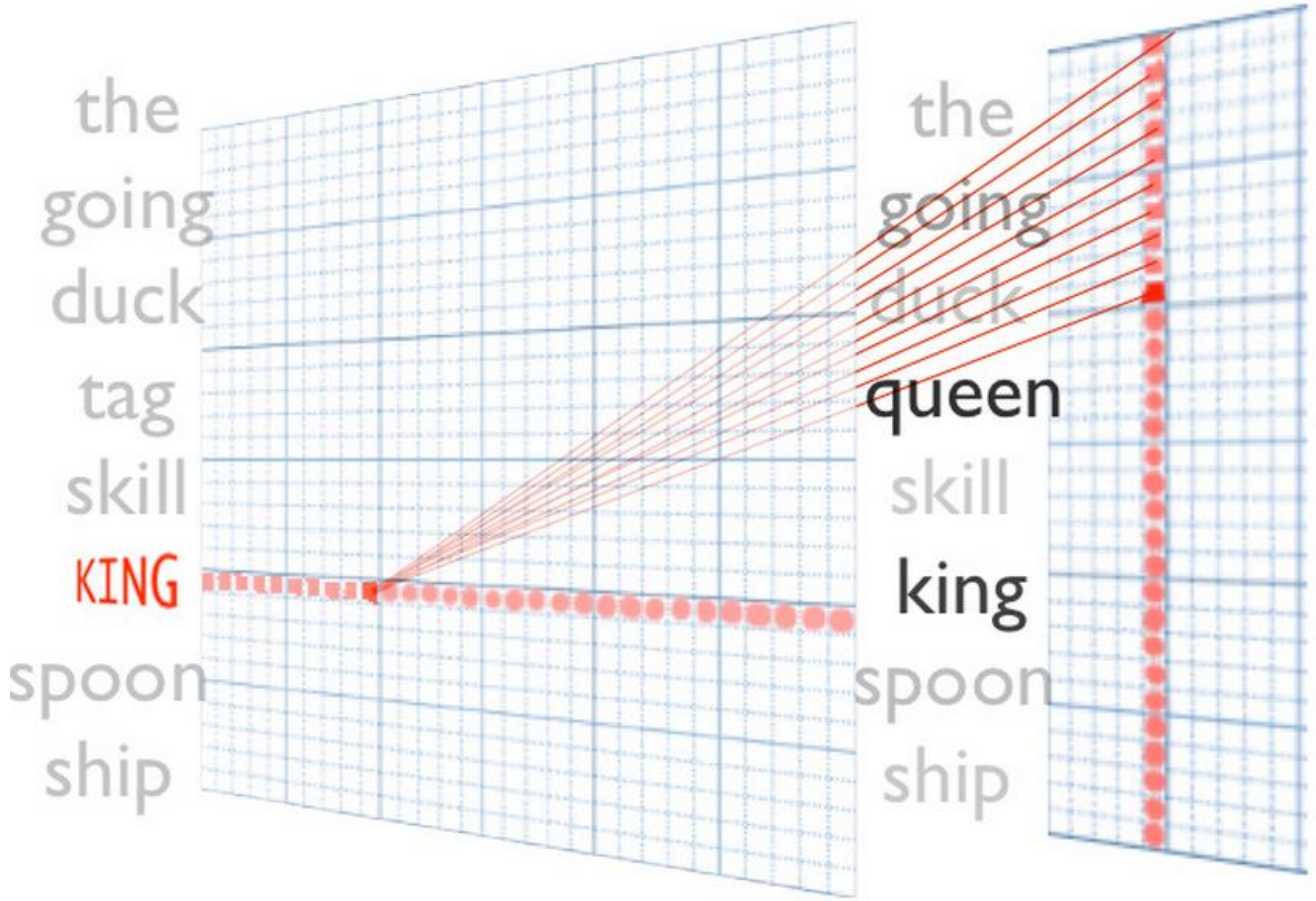
$$p(\text{I want chinese food}) = P(\text{want}|\text{I}) \times P(\text{chinese}|\text{want}) \times P(\text{food}|\text{chinese})$$

语言模型

n	模型参数数量
1 (unigram)	2×10^5
2 (bigram)	4×10^{10}
3 (trigram)	8×10^{15}
4 (4-gram)	16×10^{20}

假设词典的大小是N则模型参数的量级是 $(O(N^n))$

词向量



the
going
duck
tag
skill
KING
spoon
ship
the
going
duck
queen
skill
king
spoon
ship

词向量

$$expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

词向量

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

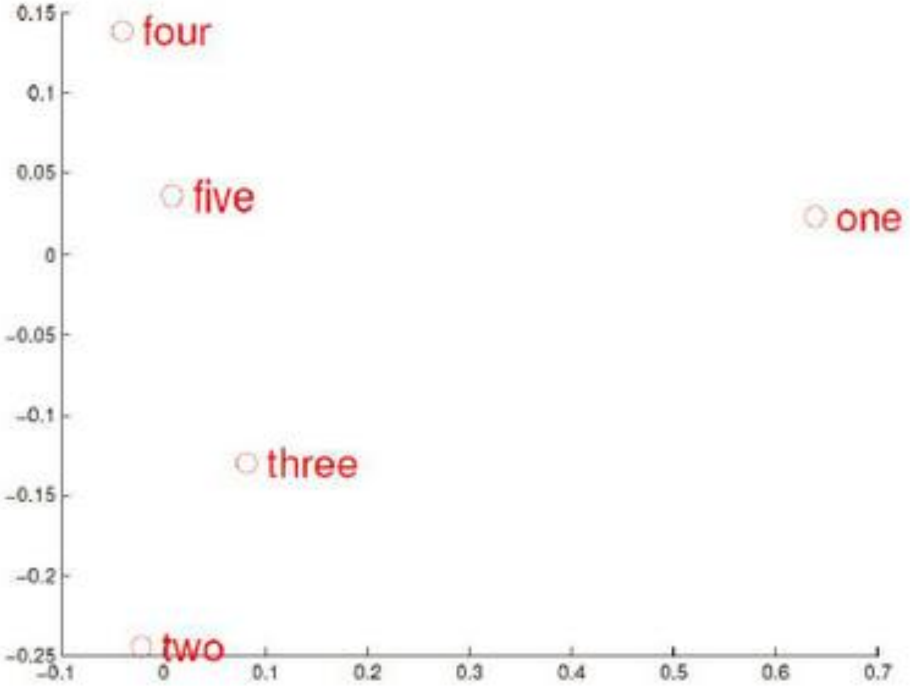

litoria


leptodactylidae

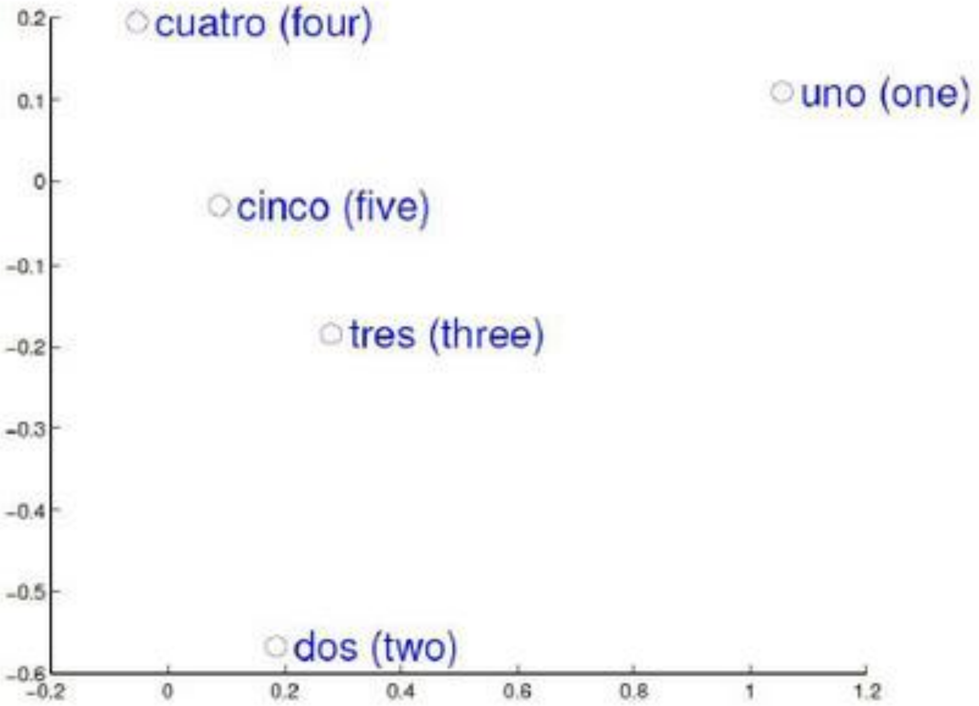

rana


eleutherodactylus

词向量

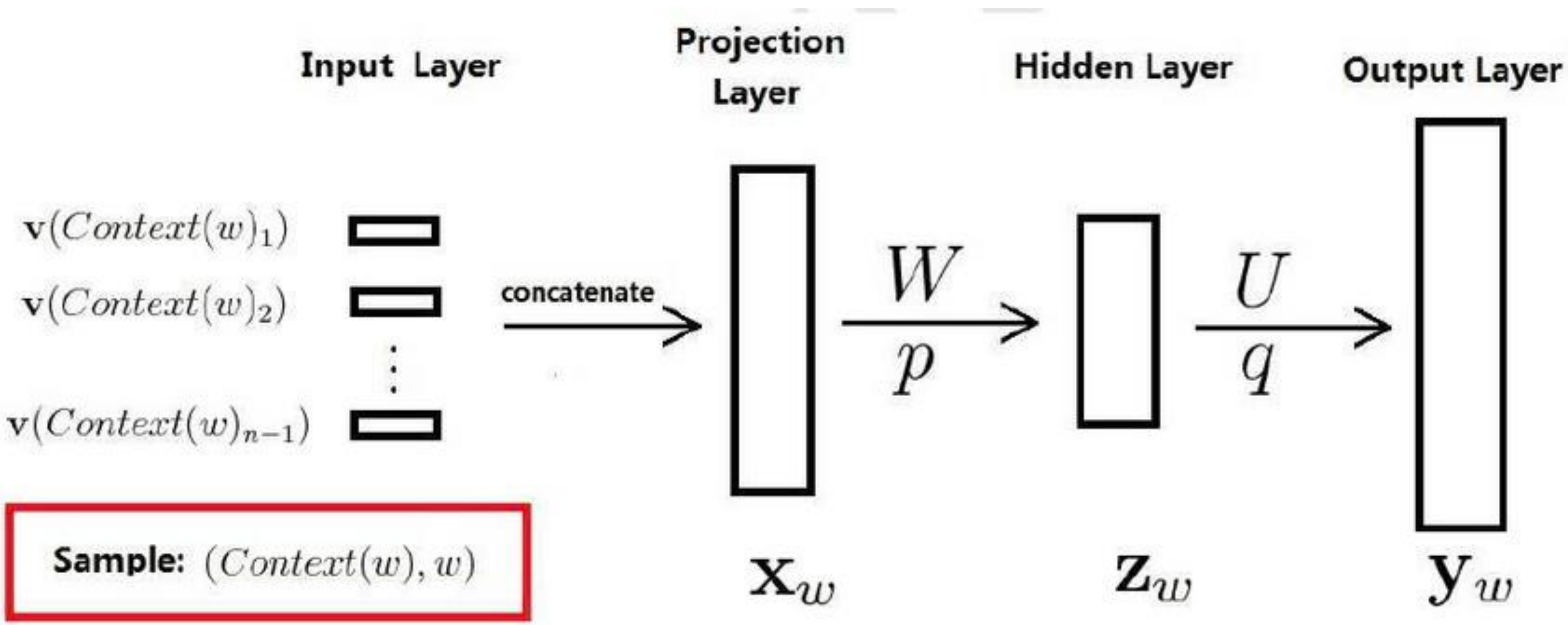


0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
four
five
one
three
two



0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
cuatro (four)
uno (one)
cinco (five)
tres (three)
dos (two)

神经网络模型



Input Layer
Projection Layer
Hidden Layer
Output Layer
$\mathbf{v}(Context(w)_1)$
$\mathbf{v}(Context(w)_2)$
$\mathbf{v}(Context(w)_{n-1})$
concatenate
W
p
U
q
Sample: $(Context(w), w)$
$\mathbf{x}_w$
$\mathbf{z}_w$
$\mathbf{y}_w$

神经网络模型

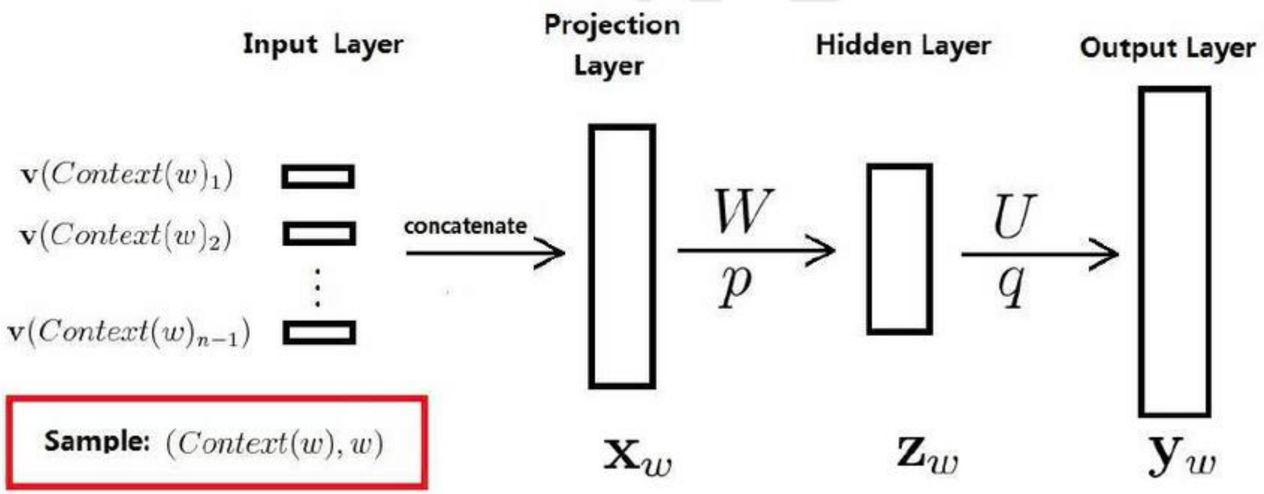
训练样本：$(Context(w), w)$ 包括前n-1个词分别的向量,假定每个词向量大小m

投影层：(n-1)*m 首尾拼接起来的大向量

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

表示上下文为 $Context(w)$ 时,下一个词恰好为词典中第i个词的概率

归一化：$p(w|Context(w)) = \frac{e^{y_{w,i_w}}}{\sum_{i=1}^{N} e^{y_{w,i}}}$



神经网络模型

S1 = “我 今天 去 网咖”　　出现了1000次
S2 = “我 今天 去 网吧”　　出现了10次

对于N-gram模型：P(S1) >> P(S2)
而神经网络模型计算的P(S1) ≈ P(S2)

神经网络模型

A dog is running in the room

A cat is running in the room

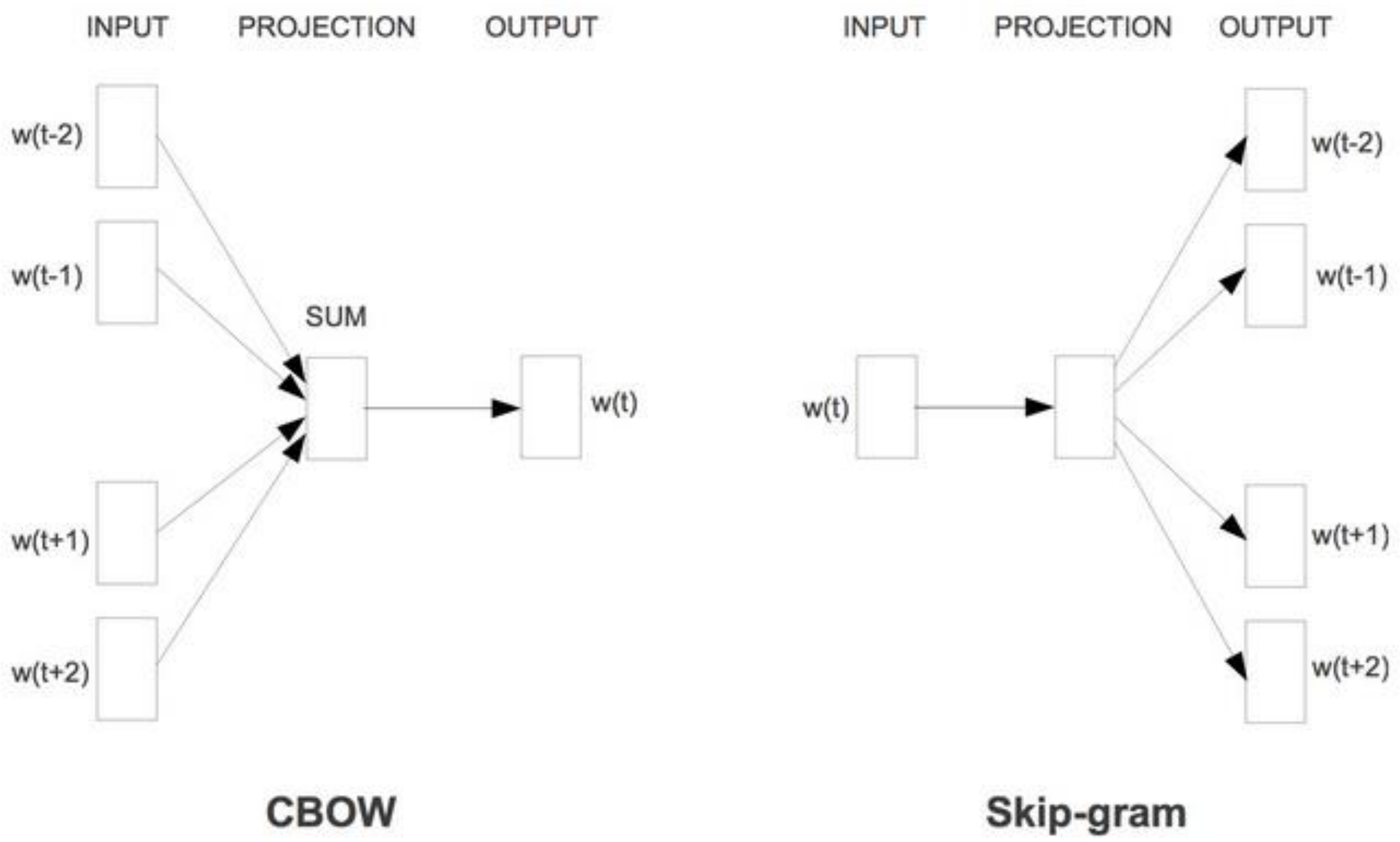
The cat is running in a room

A dog is walking in a bedroom

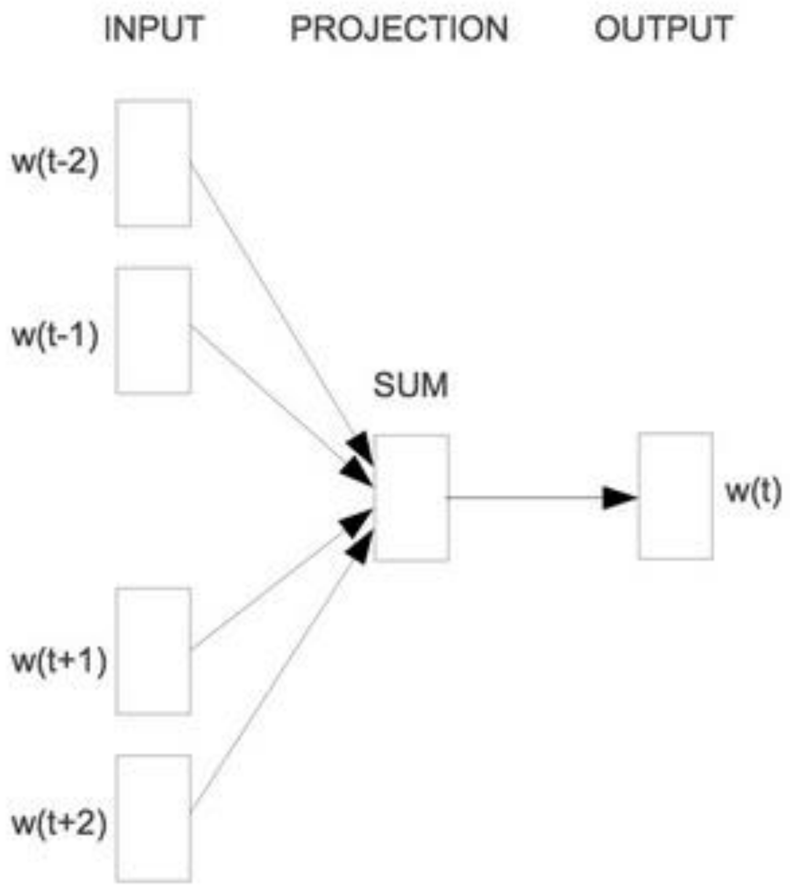
The dog was walking in the room

只要语料库中出现其中一个
其他句子的概率也会相应的增大

Hierarchical Softmax



CBOW



CBOW

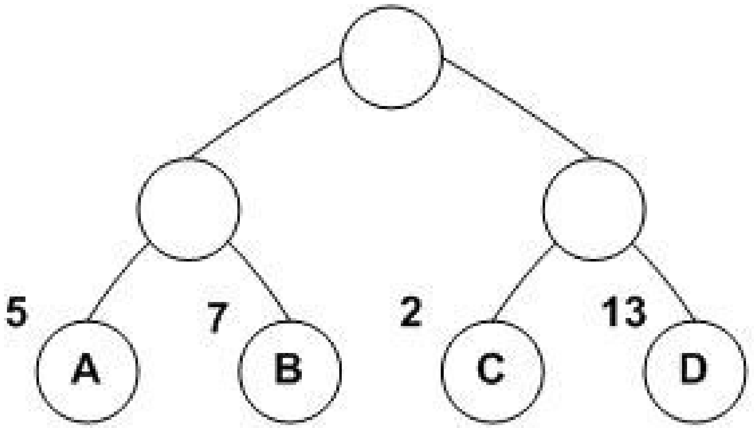
CBOW 是 Continuous Bag-of-Words Model 的缩写，是一种根据上下文的词语预测当前词语的出现概率的模型

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

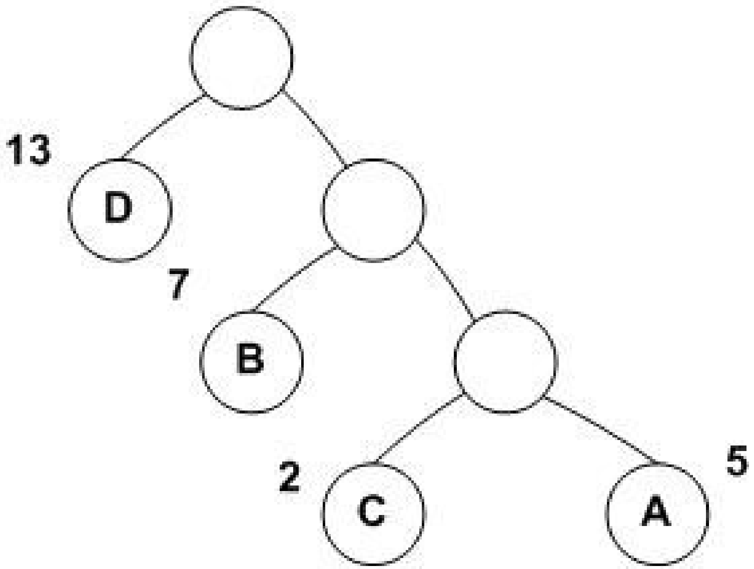
哈夫曼树

什么是哈夫曼树呢？

哈夫曼树是一种带权路径长度最短的二叉树，也称为最优二叉树。下面用一幅图来说明。



图a



图b

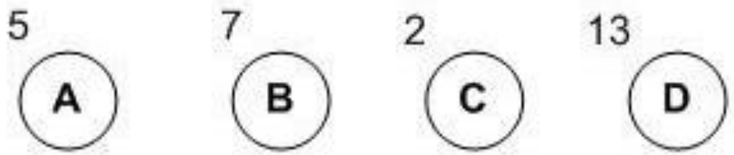
它们的带权路径长度分别为：

图a： WPL=5*2+7*2+2*2+13*2=54

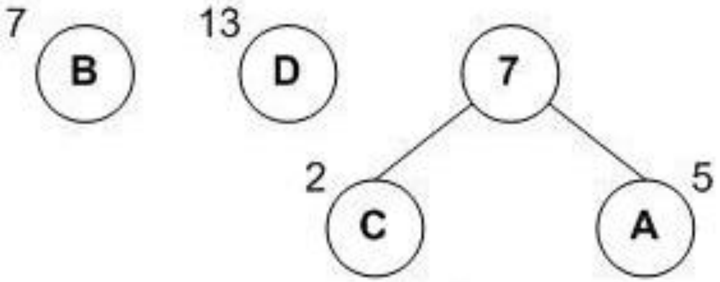
图b： WPL=5*3+2*3+7*2+13*1=48

可见，图b的带权路径长度较小，我们可以证明图b就是哈夫曼树(也称为最优二叉树)。

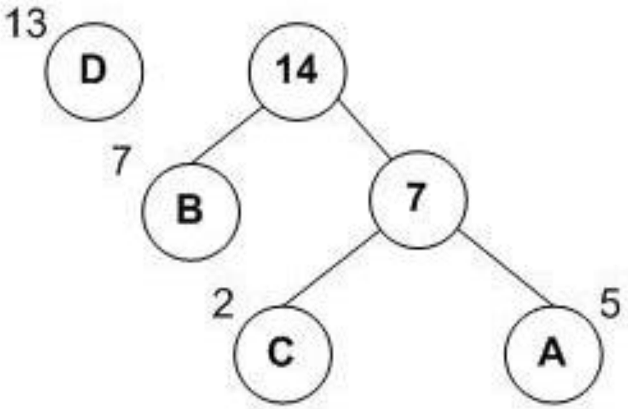
哈夫曼树



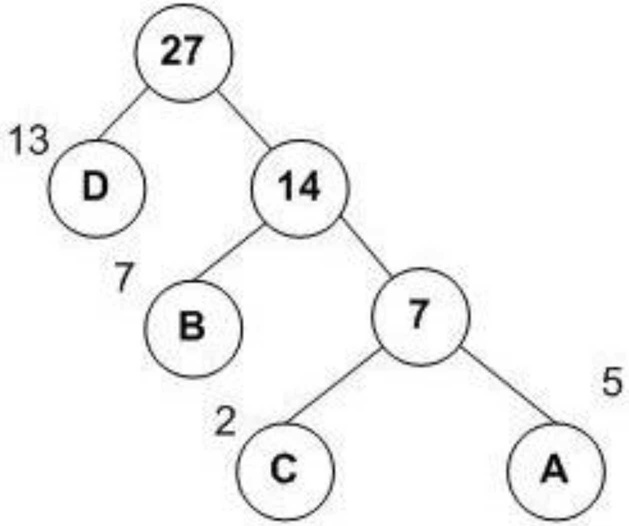
(a)初始森林



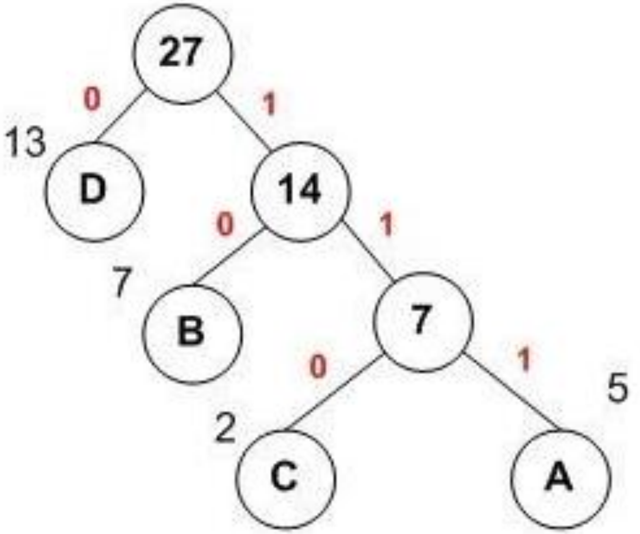
(b)一次合并



(c)二次合并

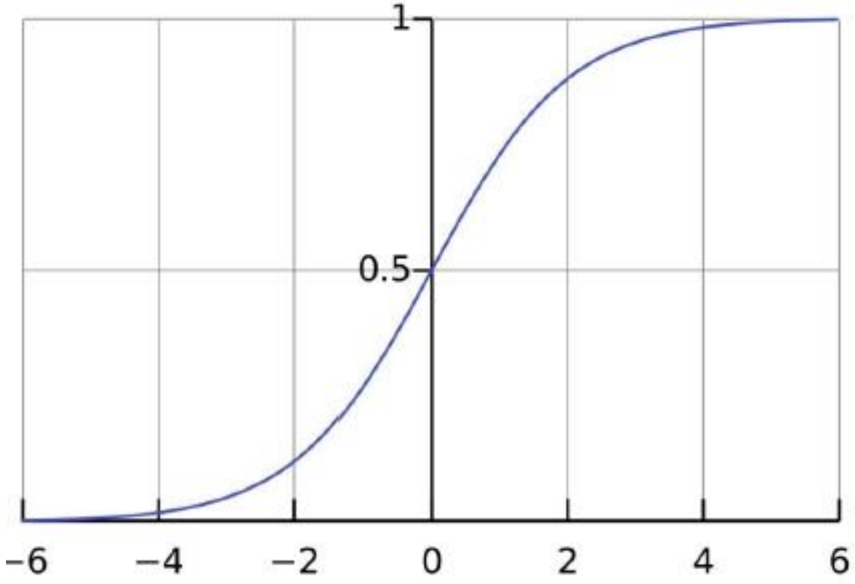


(d)哈夫曼树



Logistic回归

$$h_\theta(x) = g\left(\theta^T x\right) = \frac{1}{1+e^{-\theta^T x}}$$



Sigmoid函数

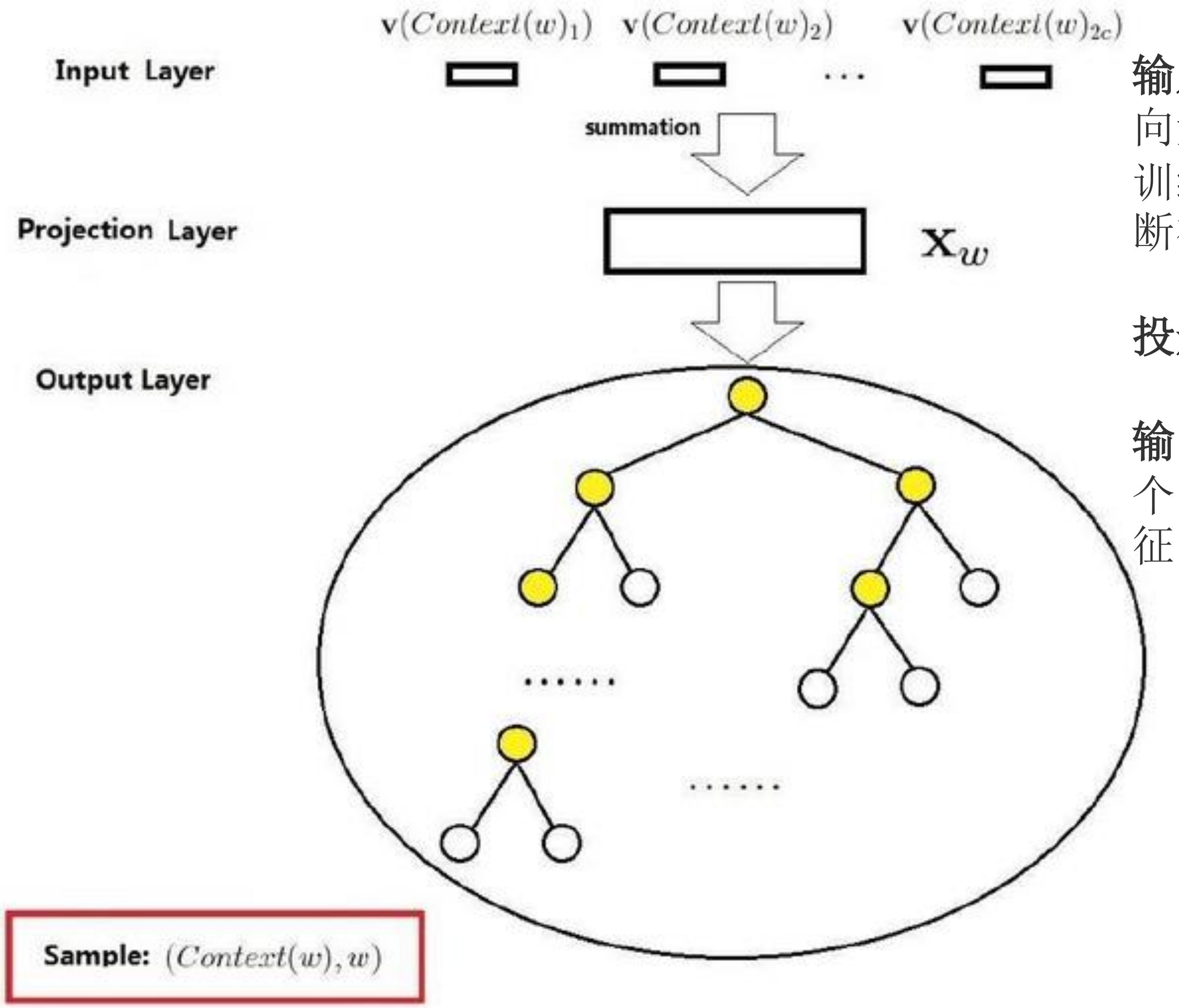
$$g(z) = \frac{1}{1+e^{-z}}$$

$$g'(x) = \left(\frac{1}{1+e^{-x}}\right) = \frac{e^{-x}}{\left(1+e^{-x}\right)^2}$$

$$= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right)$$

$$= g(x) \cdot (1 - g(x))$$

CBOW



输入层是上下文的词语的词向量，在训练CBOW模型，词向量只是个副产品，确切来说，是CBOW模型的一个参数。训练开始的时候，词向量是个随机值，随着训练的进行不断被更新）。

投影层对其求和，所谓求和，就是简单的向量加法。

输出层输出最可能的w。由于语料库中词汇量是固定的|C|个，所以上述过程其实可以看做一个多分类问题。给定特征，从|C|个分类中挑一个。

CBOW

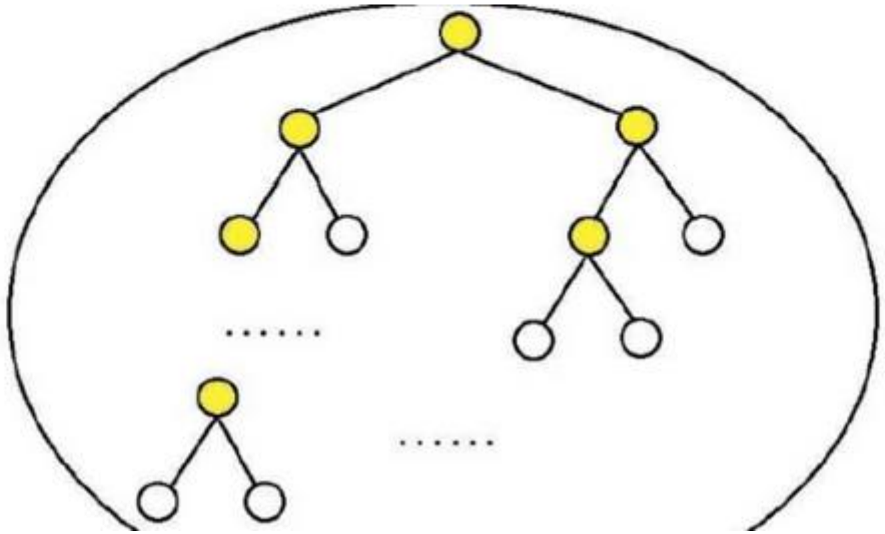
1. p^w从根结点出发到达w对应叶子结点的路径.

2. l^w路径中包含结点的个数

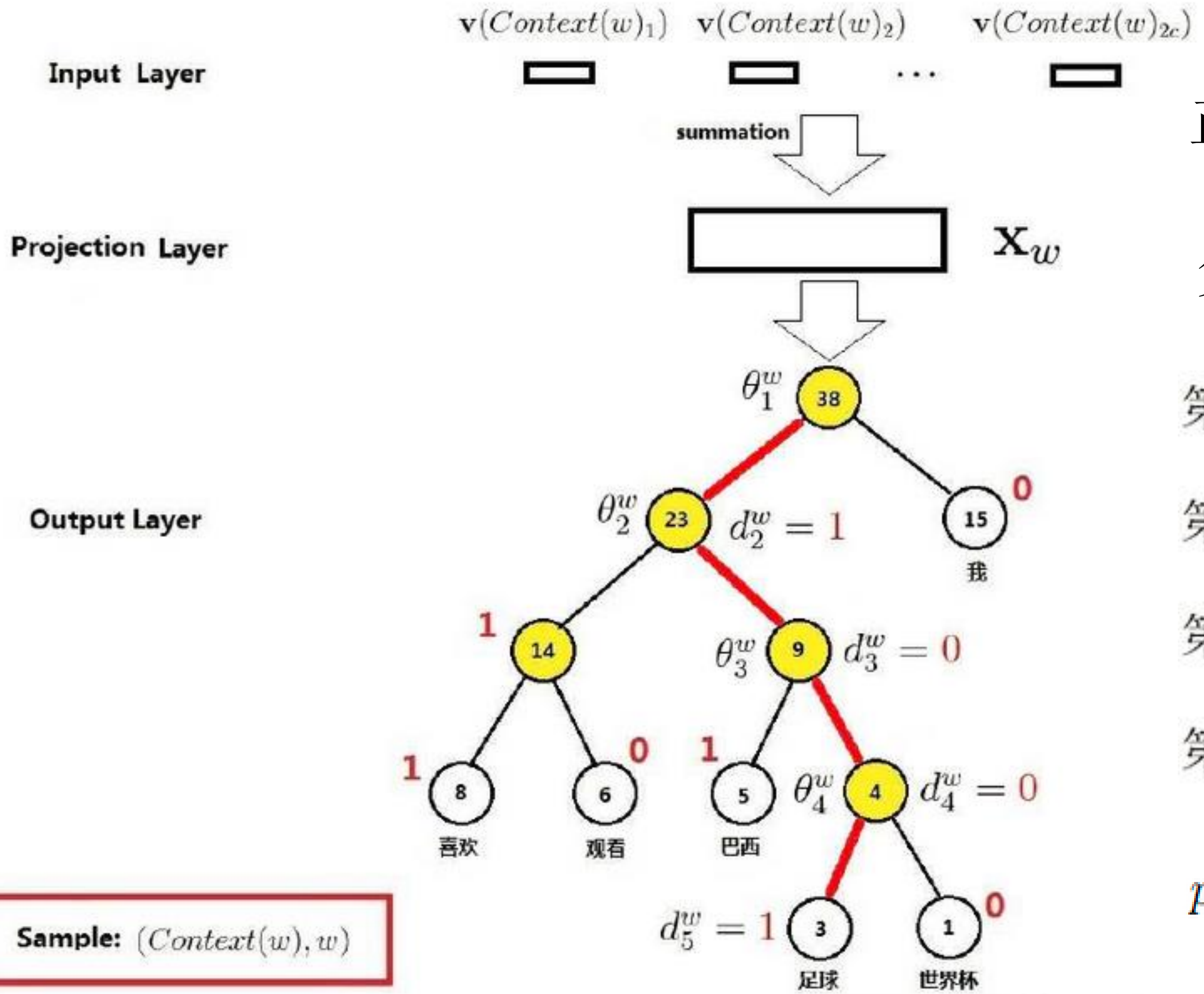
3. $p_1^w, p_2^w, \cdots, p_{l^w}^w$ 路径 p^w 中的各个节点

4. $d_2^w, d_3^w, \cdots, d_{l^w}^w \in \{0,1\}$词w的编码， d_j^w 表示路径 p^w 第j个节点对应的编码（根节点无编码）

5. $\theta_1^w, \theta_2^w, \cdots, \theta_{l^w-1}^w \in \mathbb{R}^m$ 路径 p^w 中非叶节点对应的参数向量



CBOW



Sample: $(Context(w), w)$

正例概率：$\sigma(\mathbf{x}_w^\top\theta) = \dfrac{1}{1+e^{-\mathbf{x}_w^\top\theta}}$

负例概率：$1-\sigma(\mathbf{x}_w^\top\theta)$

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

第 2 次：$p(d_3^w|\mathbf{x}_w, \theta_2^w) = \sigma(\mathbf{x}_w^\top\theta_2^w);$

第 3 次：$p(d_4^w|\mathbf{x}_w, \theta_3^w) = \sigma(\mathbf{x}_w^\top\theta_3^w));$

第 4 次：$p(d_5^w|\mathbf{x}_w, \theta_4^w) = 1-\sigma(\mathbf{x}_w^\top\theta_4^w)$

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

CBOW

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

代入

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

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

CBOW

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

sigmoid函数的导数: $\sigma'(x) = \sigma(x)[1-\sigma(x)]$

代入上上式得到: $(1-d_j^w)[1-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]\mathbf{x}_w - d_j^w \sigma(\mathbf{x}_w^\top \theta_{j-1}^w)\mathbf{x}_w$

合并同类项得到: $[1-d_j^w-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]\,\mathbf{x}_w$

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

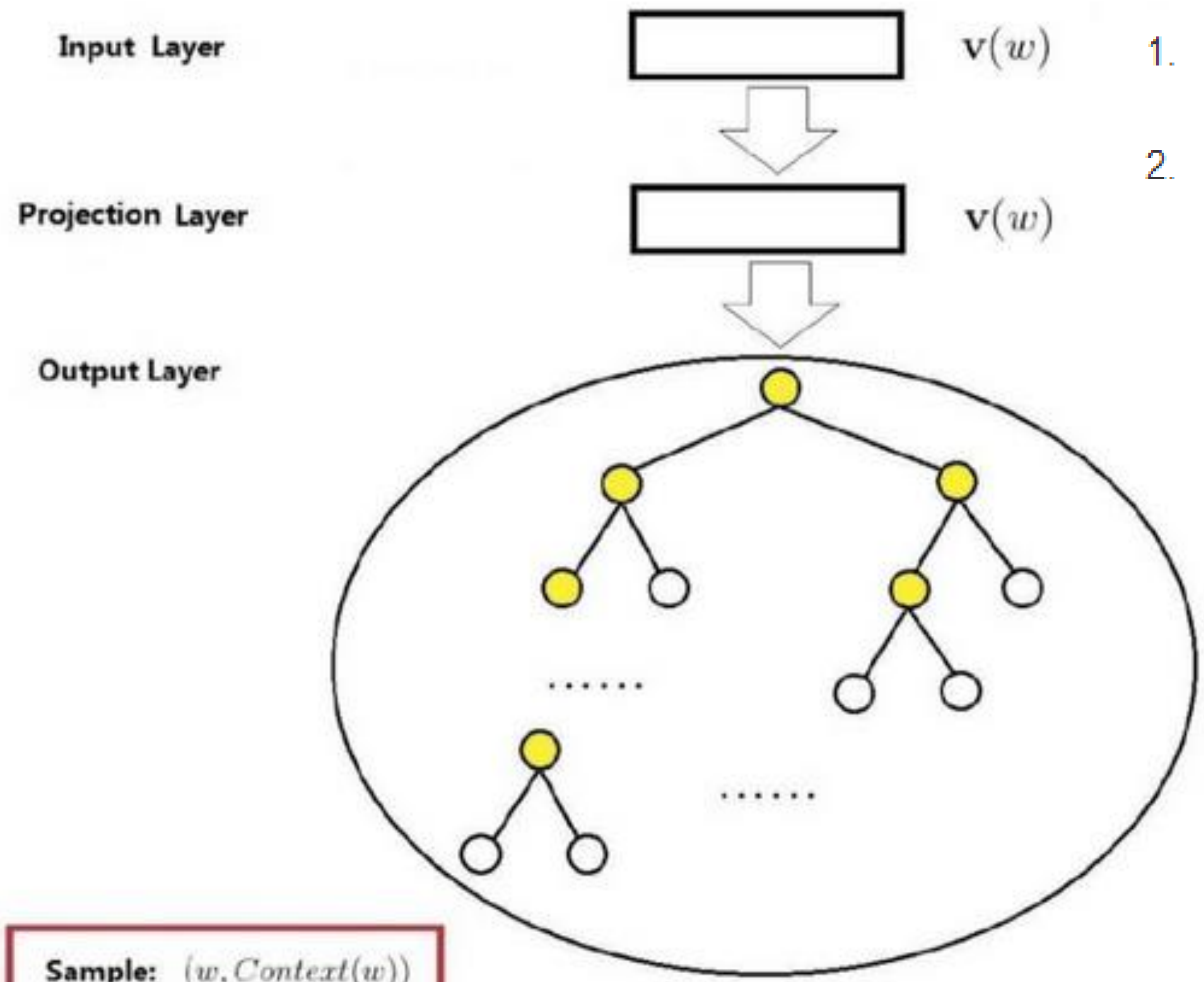
$$\frac{\partial \mathcal{L}(w,j)}{\partial \mathbf{x}_w} = [1-d_j^w-\sigma(\mathbf{x}_w^\top \theta_{j-1}^w)]\,\theta_{j-1}^w$$

CBOW

不过$\mathbf{x}_w$是上下文的词向量的和，不是上下文单个词的词向量。怎么把这个更新量应用到单个词的词向量上去呢？word2vec采取的是直接将$\mathbf{x}_w$的更新量整个应用到每个单词的词向量上去：

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

Skip-gram



Sample: $(w, Context(w))$

1. 输入层不再是多个词向量，而是一个词向量
2. 投影层其实什么事情都没干，直接将输入层的词向量传递给输出层

Negative Sampling

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

负样本那么多，该如何选取呢?

对于一个给定的正样本 $(Context(w), w)$，我们希望最大化

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

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

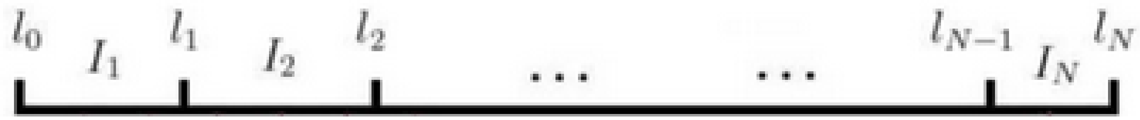
Negative Sampling

任何采样算法都应该保证频次越高的样本越容易被采样出来。基本的思路是对于长度为1的线段，根据词语的词频将其公平地分配给每个词语：

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

counter就是w的词频。

于是我们将该线段公平地分配了：



接下来我们只要生成一个0-1之间的随机数，看看落到哪个区间，就能采样到该区间对应的单词了，很公平。

Negative Sampling

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

$\sigma(\mathbf{x}_w^\top \theta^w)$ 表示当上下文为 $Context(w)$ 时，预测中心词为 w 的概率

$\sigma(\mathbf{x}_w^\top \theta^u)$, $u \in NEG(w)$ 则表示当上下文为 $Context(w)$ 时，预测中心词为 u 的概率

对于一个给定的语料库 $\mathcal{C}$

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

Negative Sampling

$$\begin{aligned}\mathcal{L} &= \log G = \log \prod_{w\in\mathcal{C}} g(w) = \sum_{w\in\mathcal{C}} \log g(w)\\ &= \sum_{w\in\mathcal{C}} \log \prod_{u\in\{w\}\cup NEG(w)} \left\{\left[\sigma(\mathbf{x}_w^\top\theta^u)\right]^{L^w(u)} \cdot \left[1-\sigma(\mathbf{x}_w^\top\theta^u)\right]^{1-L^w(u)}\right\}\\ &= \sum_{w\in\mathcal{C}} \sum_{u\in\{w\}\cup NEG(w)} \left\{L^w(u)\cdot\log\left[\sigma(\mathbf{x}_w^\top\theta^u)\right] + \left[1-L^w(u)\right]\cdot\log\left[1-\sigma(\mathbf{x}_w^\top\theta^u)\right]\right\}\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{L}(w,u)}{\partial\theta^u} &= \frac{\partial}{\partial\theta^u}\left\{L^w(u)\cdot\log\left[\sigma(\mathbf{x}_w^\top\theta^u)\right] + \left[1-L^w(u)\right]\cdot\log\left[1-\sigma(\mathbf{x}_w^\top\theta^u)\right]\right\}\\ &= L^w(u)\left[1-\sigma(\mathbf{x}_w^\top\theta^u)\right]\mathbf{x}_w - \left[1-L^w(u)\right]\sigma(\mathbf{x}_w^\top\theta^u)\mathbf{x}_w\\ &= \left\{L^w(u)\left[1-\sigma(\mathbf{x}_w^\top\theta^u)\right] - \left[1-L^w(u)\right]\sigma(\mathbf{x}_w^\top\theta^u)\right\}\mathbf{x}_w\\ &= \left[L^w(u)-\sigma(\mathbf{x}_w^\top\theta^u)\right]\mathbf{x}_w\end{aligned}$$

Negative Sampling

θ^u 的更新公式可写为 $\theta^u := \theta^u + \eta \left[L^w(u) - \sigma(\mathbf{x}_w^\top \theta^u) \right] \mathbf{x}_w$

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

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