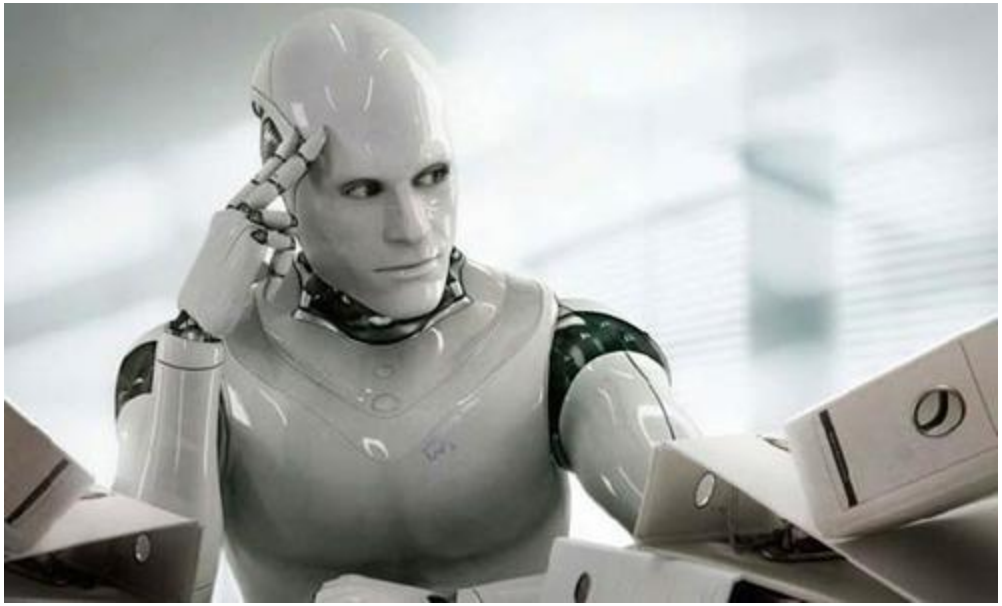
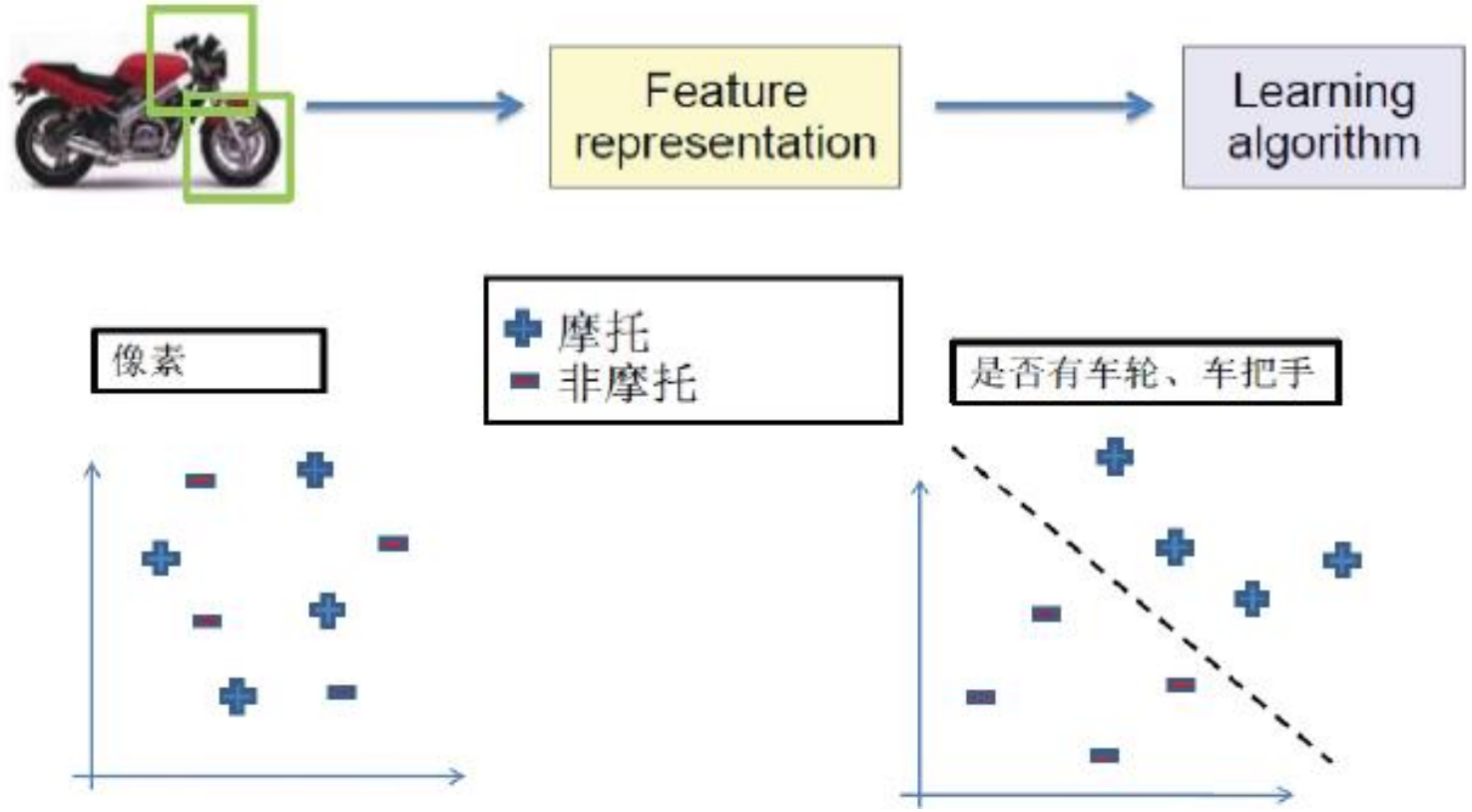
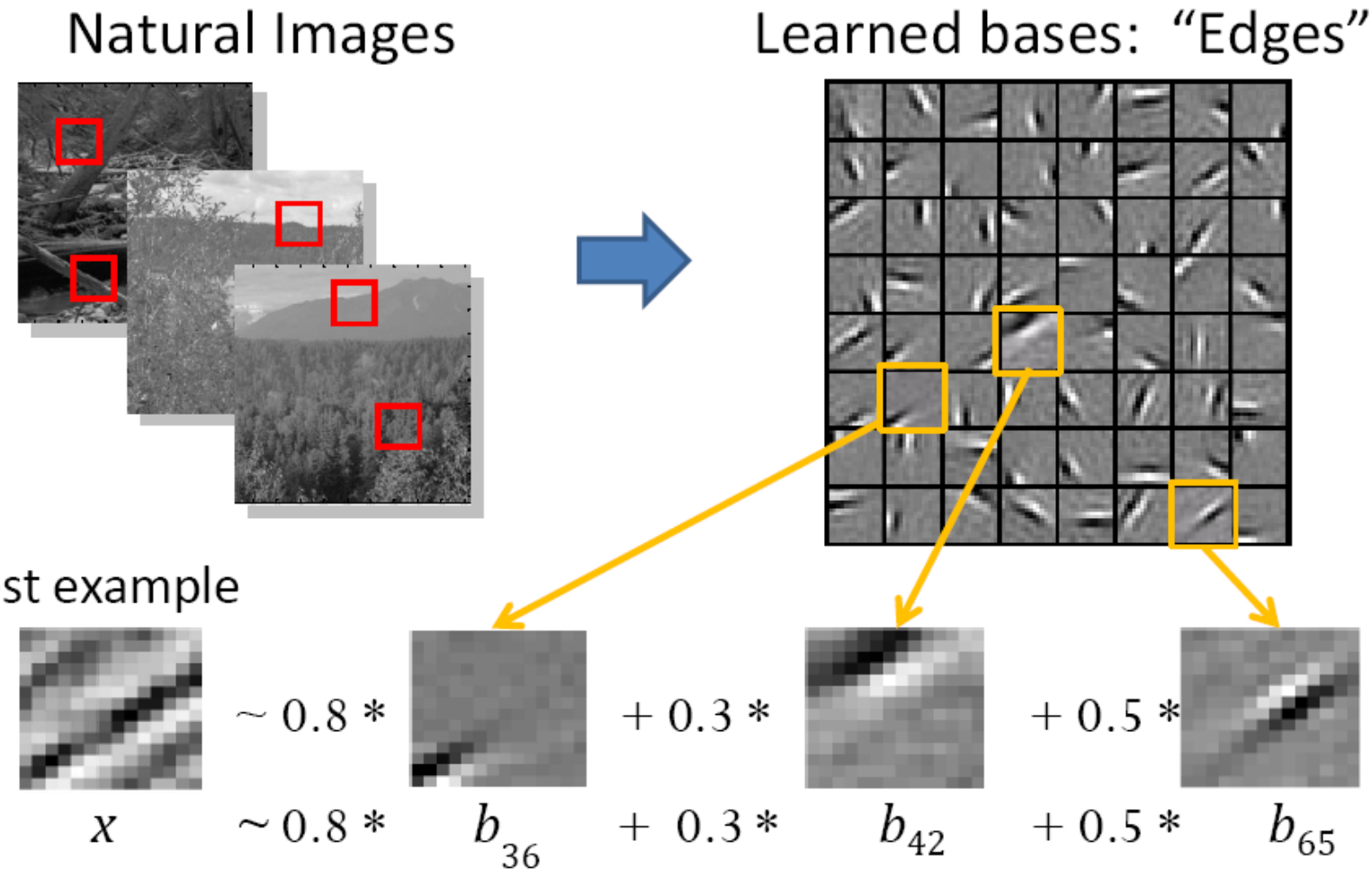
NLP基础



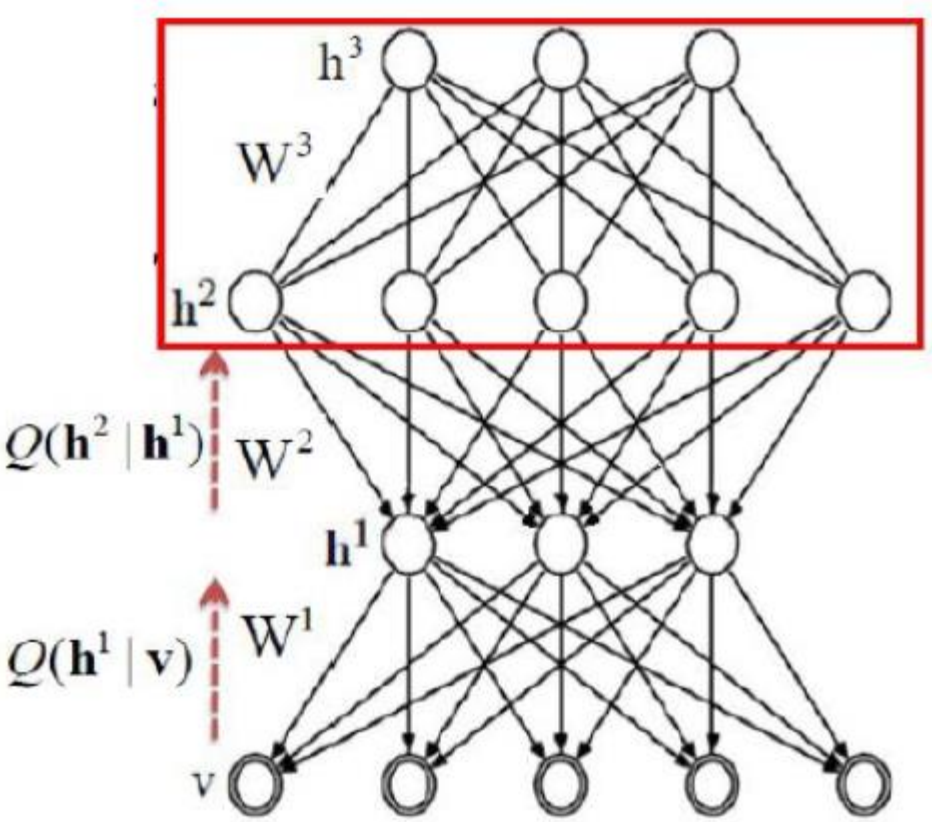
深度学习（表示学习）


Feature representation
Learning algorithm
像素
摩托
非摩托
是否有车轮、车把手

深度学习（表示学习）


Natural Images
Learned bases: “Edges”
Test example
$\sim 0.8 *$
$+ 0.3 *$
$+ 0.5 *$
$x \sim 0.8 * b_{36} + 0.3 * b_{42} + 0.5 * b_{65}$
[0, 0, …, 0, 0.8, 0, …, 0, 0.3, 0, …, 0, 0.5, …]
= coefficients (feature representation)

Layer-Wise Pre-Training



自然语言交互的时代

IBM WATSON

词袋模型BoW

- 使用一组无序的单词（word）来表达一段文字或一个文档，并且文档中每个单词的出现都是独立的。

例如：首先给出两个简单的文本文档如下：

John likes to watch movies. Mary likes too.
John also likes to watch football games.

词袋模型BoW

- 上面的词典中包含10个单词, 每个单词有唯一的索引, 那么每个文本我们可以使用一个10维的向量来表示, 向量中的元素是词典中对应的词语出现的频数。
- 如下所示：
  - [1, 2, 1, 1, 1, 0, 0, 0, 1, 1]
  - [1, 1,1, 1, 0, 1, 1, 1, 0, 0]

One-Hot 表示

- One Hot表示在传统NLP中很常用

$$
\begin{aligned}
&\text{"dog"} = [1, 0, 0, \ldots, 0] \\
&\text{"cat"} = [0, 1, 0, \ldots, 0] \\
&\text{"the"} = [0, 0, 0, \ldots, 1]
\end{aligned}
$$

Similarity(dog,cat)=0

Word Embedding

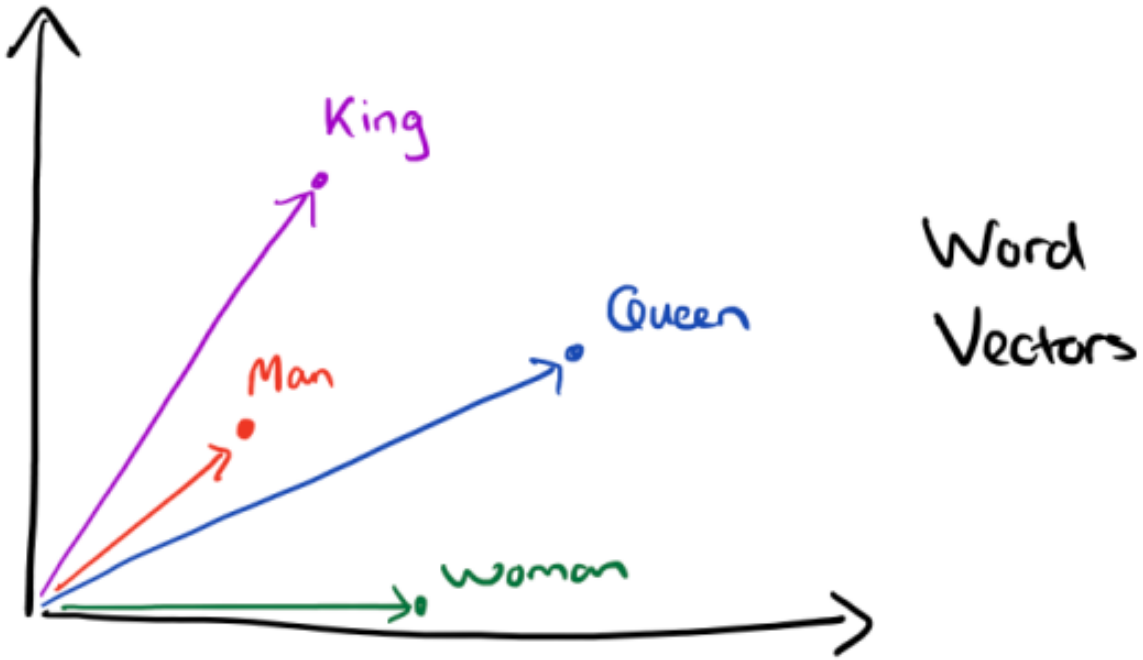
- 词向量：单词的分布式表示（Distributional Representation）

$$\text{"dog"} = [1, 0, 0.9, 0.0]$$
$$\text{"cat"} = [1, 0, 0.5, 0.2]$$
$$\text{"the"} = [0, 1, 0.0, 0.0]$$

Similarity(dog,cat)>Similarity(dog,the)

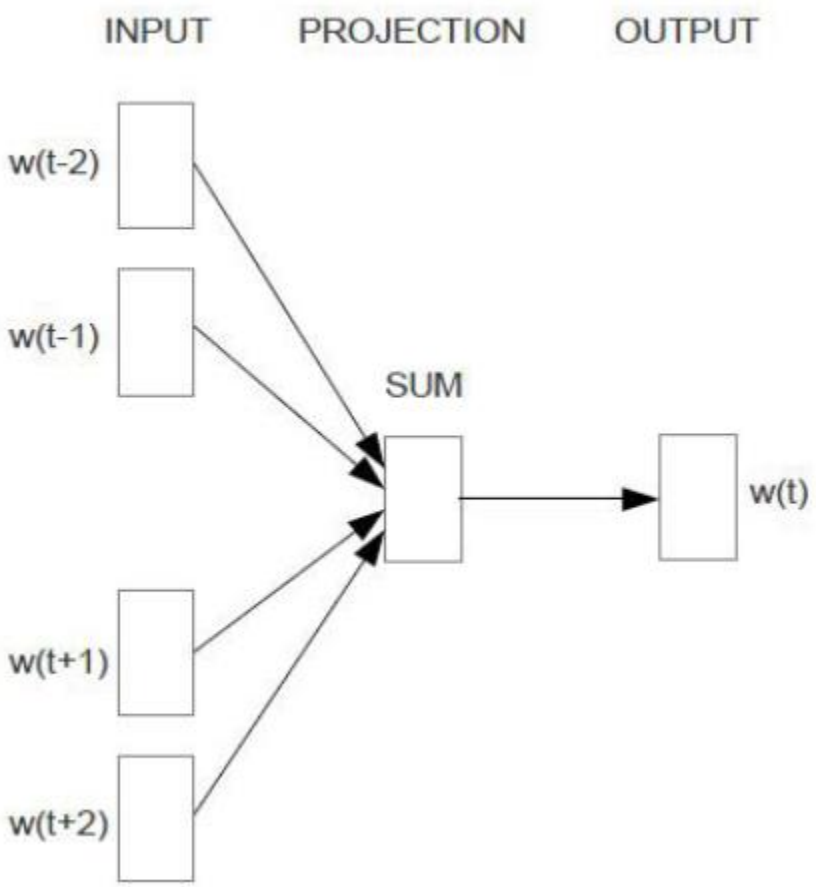
Similarity（“the dog smiles.”，“one cat cries.”）

- 词向量表征了单词使用上下文中的句法语义特征
  - One-Hot的字面匹配到DR的语义匹配



King−Man+Woman=Queen

Word2vec



CBOW: $P(w_t | w_{t-k}, w_{t-(k-1)}..., w_{t-1}, w_{t+1}, w_{t+2}..., w_{t+k})$

CBOW

An example of CBOW Model

Corpus = { I drink coffee everyday }

W^{I} = [1,0,0,0]

W^{drink} = [0,1,0,0]

target W^{coffee} = [0,0,1,0]

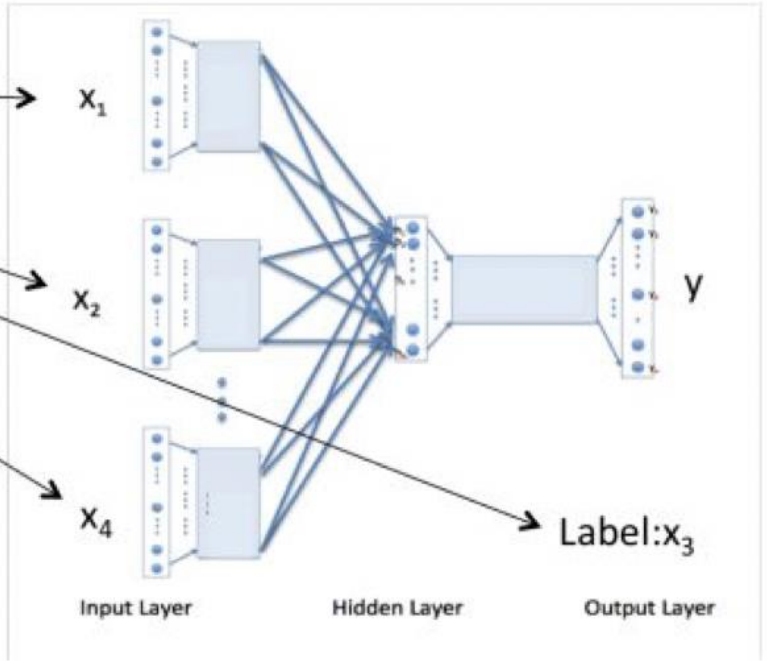
$W^{everyday}$ = [0,0,0,1]

Set:

Window size: 2

Target word: coffee

Context word: I, drink, everyday



CBOW

An example of CBOW Model

Corpus = { I drink coffee everyday }

Initialize:

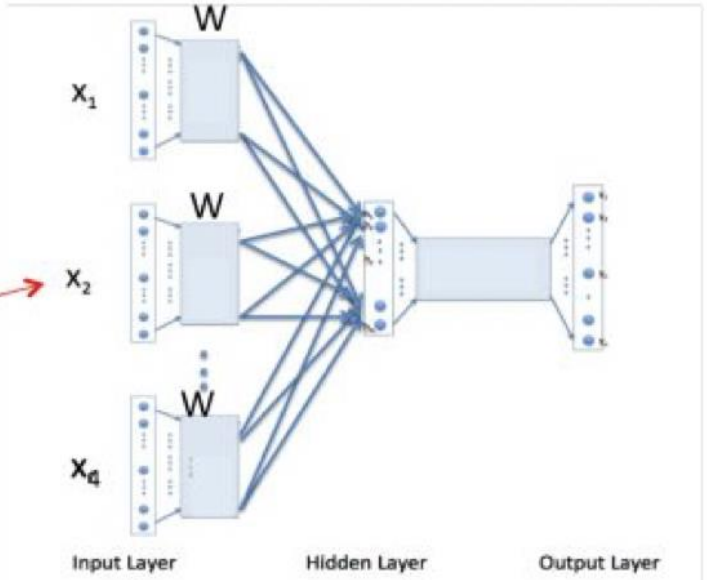
$$W = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 2 & 1 & 2 \\ -1 & 1 & 1 & 1 \end{bmatrix}$$

Ex:

$W^{drink} = [0,1,0,0]$

$$Wx_2 = v_2$$

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 2 & 1 & 2 \\ -1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$



Continuous bag-of-words (Mikolov et al., 2013)

CBOW

An example of CBOW Model

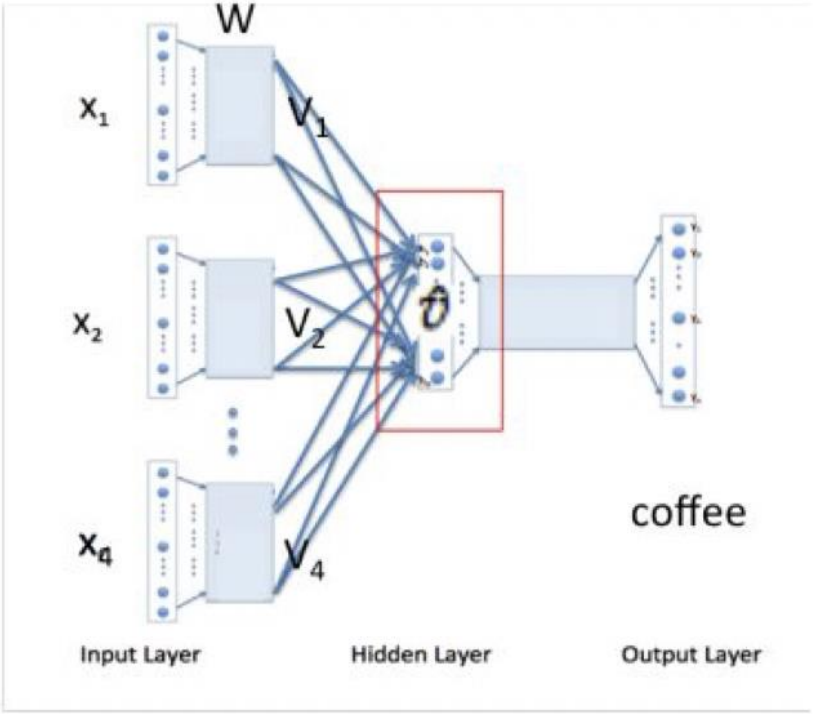
Corpus = { I drink coffee everyday }

Initialize:

$$W = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 2 & 1 & 2 \\ -1 & 1 & 1 & 1 \end{bmatrix}$$

$$\frac{V_1 + V_2 + V_4}{3} = \hat{v}$$

$$\frac{1}{3}\left(\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 1.67 \\ 0.33 \end{bmatrix}$$



CBOW

An example of CBOW Model

Corpus = { I drink coffee everyday }

W^{I} = [1,0,0,0]

W^{drink} = [0,1,0,0]

target W^{coffee} = [0,0,1,0]

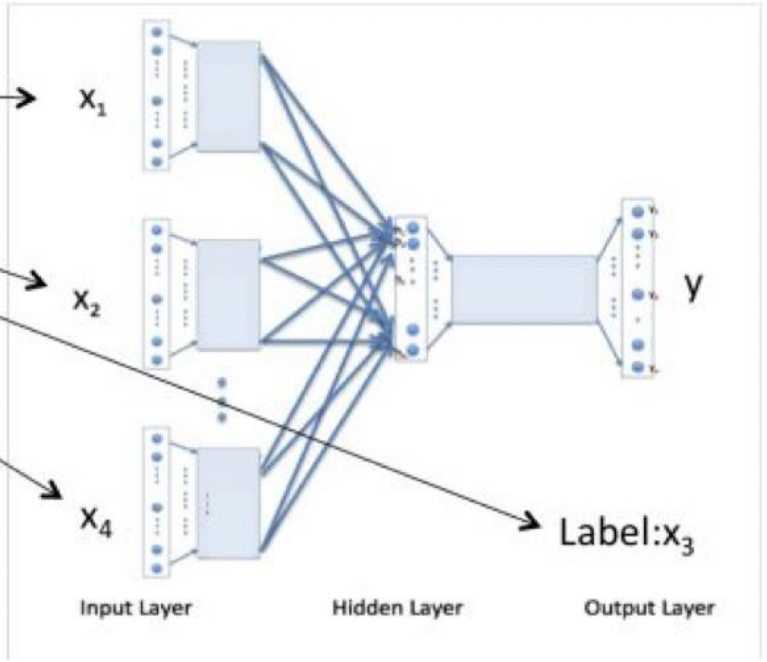
$W^{everyday}$ = [0,0,0,1]

Set:

Window size: 2

Target word: coffee

Context word: I, drink, everyday



CBOW

An example of CBOW Model

Output: Probability distribution

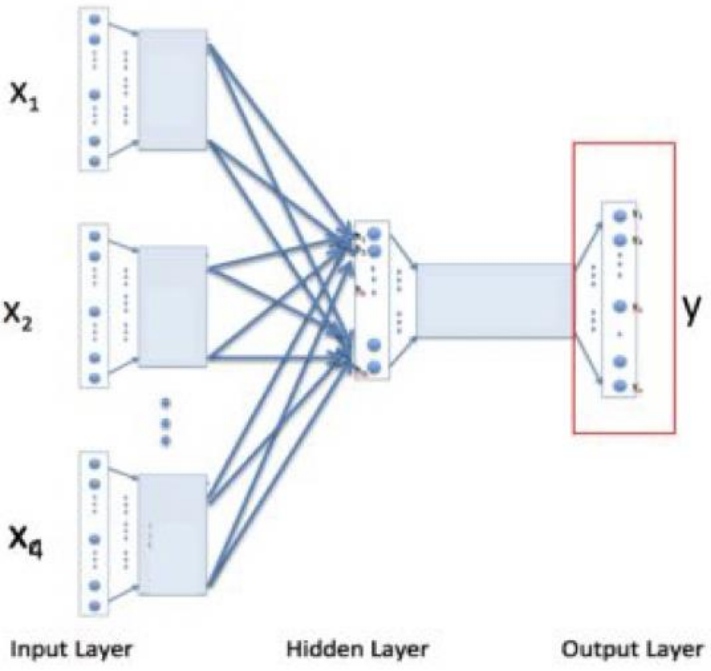
$$softmax(\mathbf{u}_o) = \mathbf{y}$$

$$softmax\left(\begin{bmatrix} 4.01 \\ 2.01 \\ 5.00 \\ 3.34 \end{bmatrix}\right) = \begin{bmatrix} 0.23 \\ 0.03 \\ 0.62 \\ 0.12 \end{bmatrix}$$

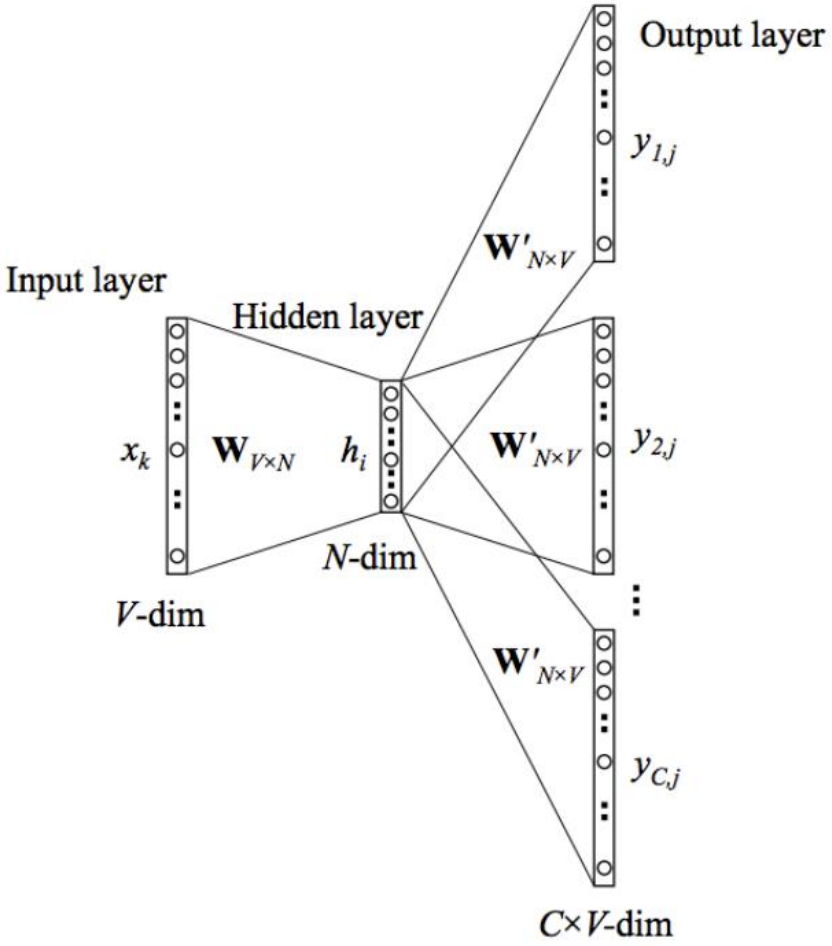
Probability of “coffee”

We desire probability generated to match the true probability(label) x_3 [0,0,1,0]

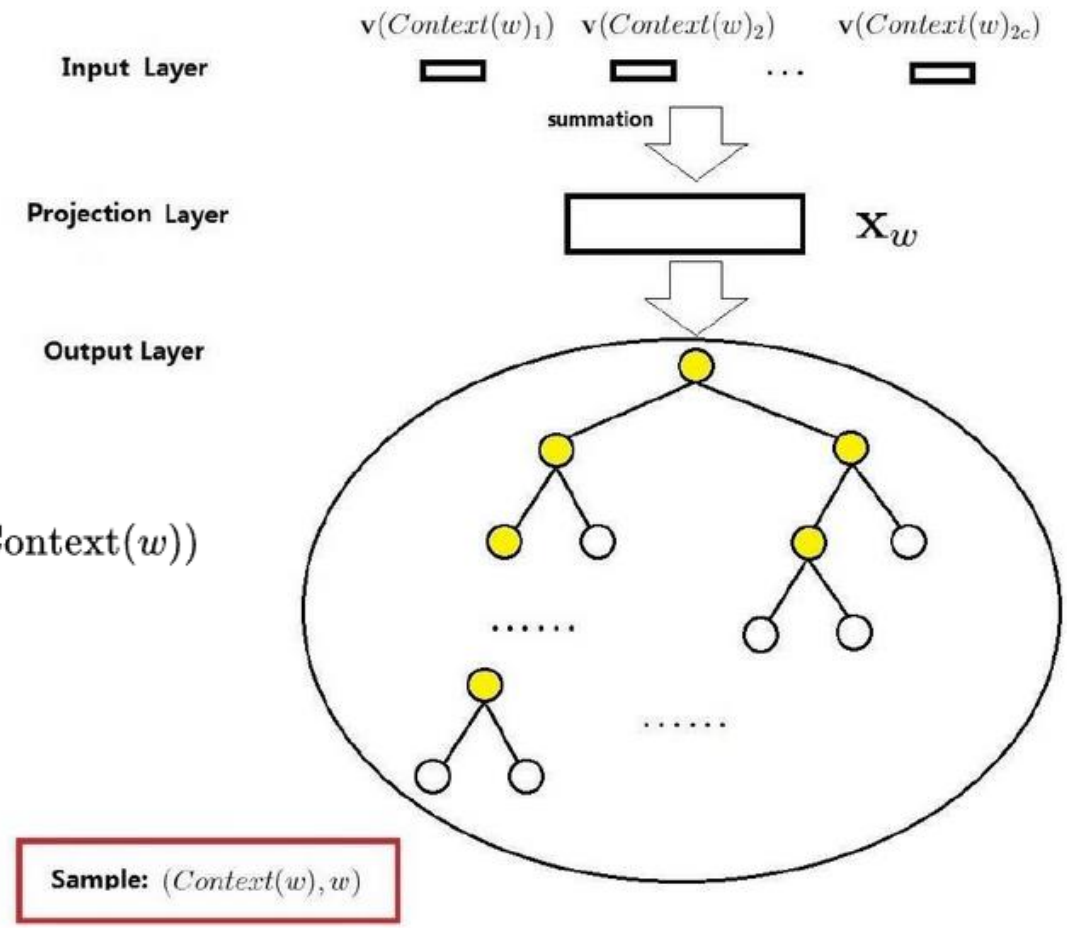
Use gradient descent to update W and W’



Skip-Gram


Output layer
$y_{1,j}$
$\mathbf{W}'_{N\times V}$
Input layer
Hidden layer
x_k
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
$y_{2,j}$
N-dim
V-dim
$\mathbf{W}'_{N\times V}$
$y_{C,j}$
$C\times V$-dim

word2vec

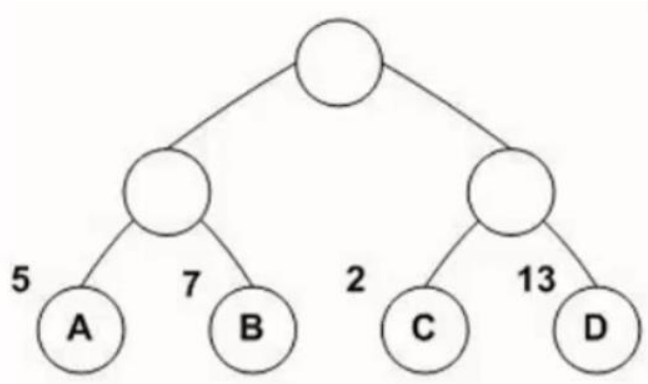


$$\mathcal{L} = \sum_{w \in \mathcal{C}} \log p(w|\,\text{Context}(w))$$

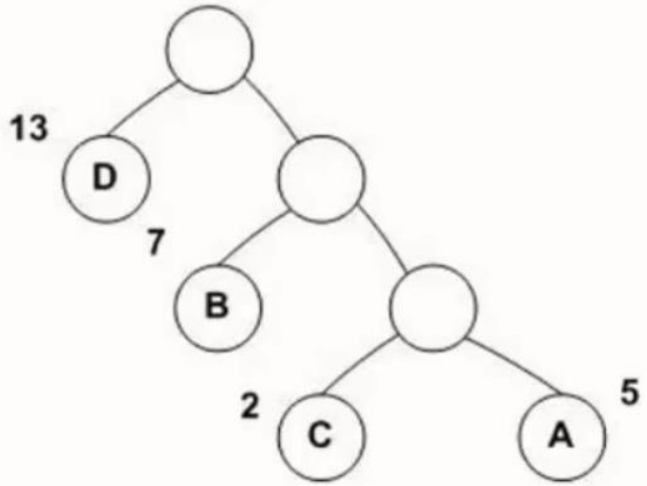
CBOW+ Hierarchical Softmax

层次Softmax

- 霍夫曼编码



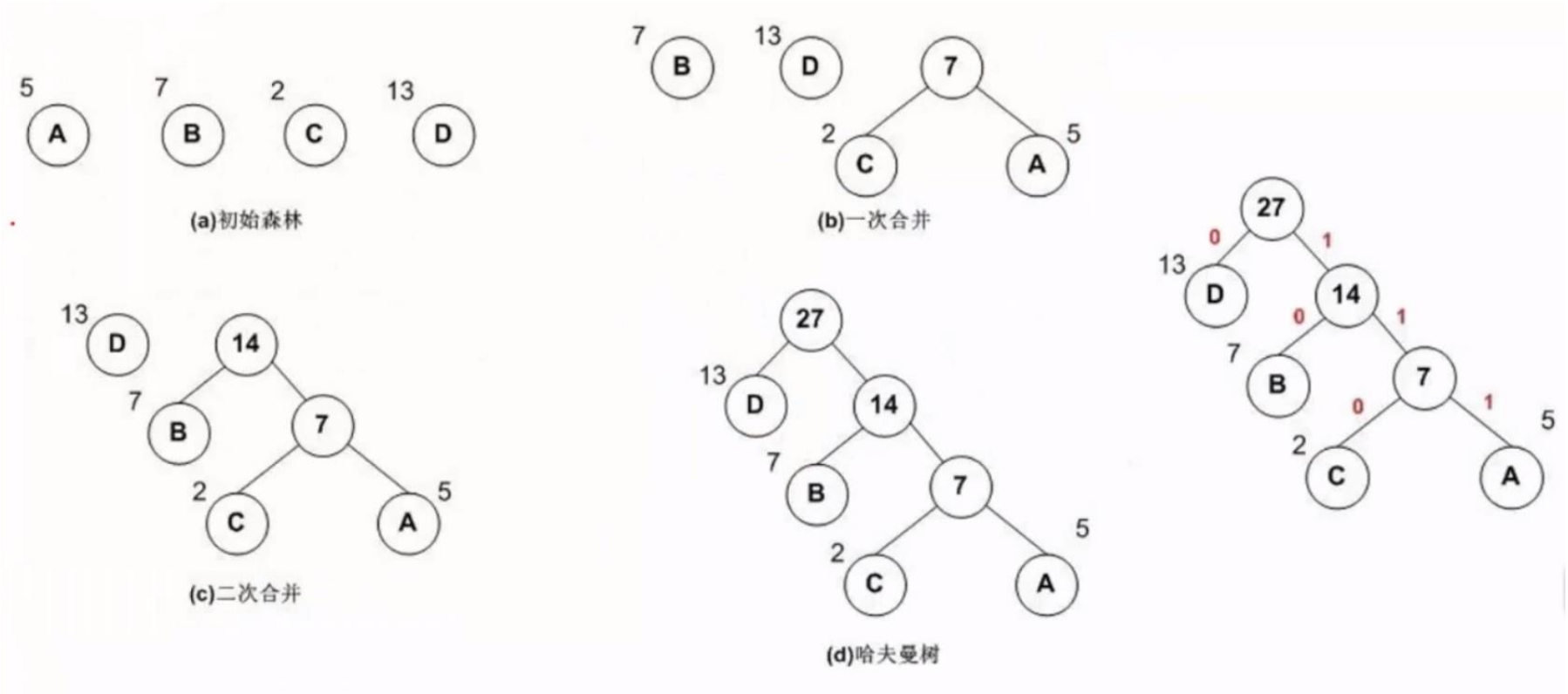
图a



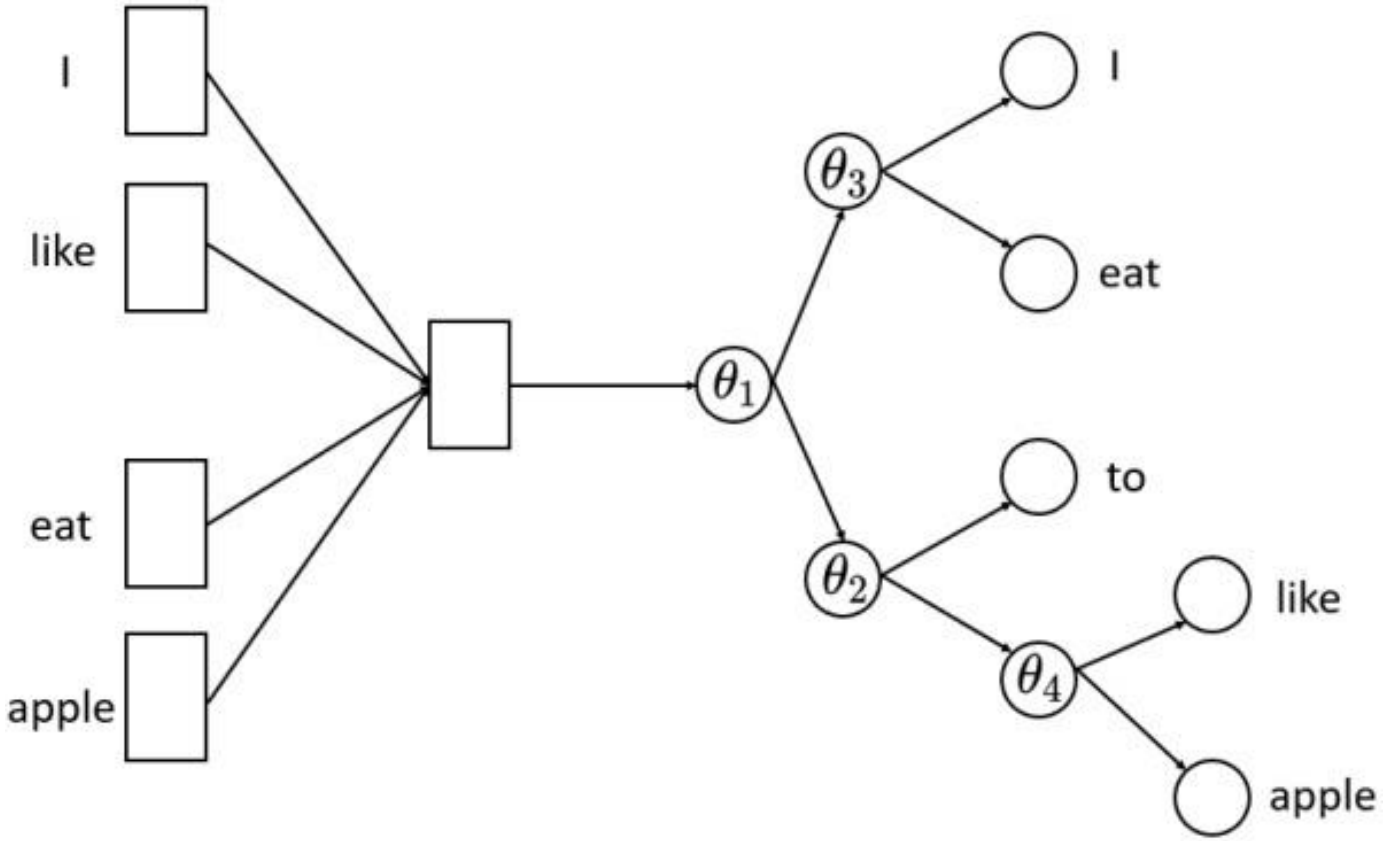
图b

层次Softmax

- 霍夫曼编码的构造



层次Softmax


I
like
eat
apple
θ_1
θ_2
θ_3
θ_4
I
eat
to
like
apple

层次Softmax

正类别的概率： $\sigma(X_i\theta) = \frac{1}{1+e^{-x_i\theta}}$

负类别的概率： $1-\sigma(X_i\theta)$

每个label都会有又一条路径，对于训练样本的特征向量X_i 和对应的label Y_i，预测出来 X_i 的样本属于所对应的label是 Y_i 的概率：

$$P(Y_i|X_i) = \prod_{j=2}^{l} P(d_j|X_i,\theta_{j-1})$$

其中：

$$P(d_j|X_i,\theta_{j-1}) = \begin{cases} \sigma(X_i\theta), & \text{if d\_j=1} \\ 1-\sigma(X_i\theta), & \text{if d\_j=0} \end{cases}$$

极大似然估计

当模型是条件概率分布，损失函数可用对数函数表示，经验风险最小化等价于极大似然估计。

对数似然函数：

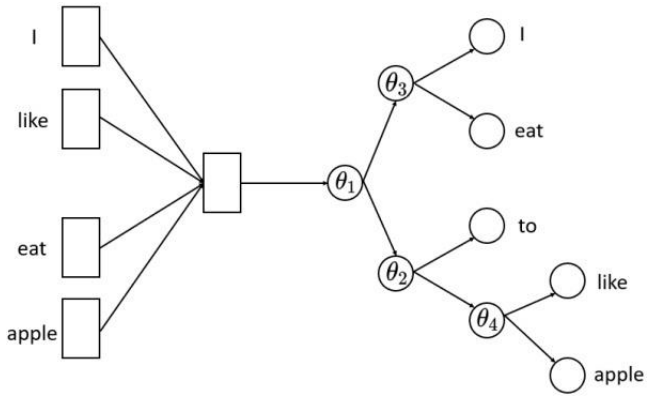
$$L(Y,P(Y|X)) = -\log P(Y|X)$$

目标函数：

$$\iota = \frac{1}{n}\Sigma_{i=1}^{n} \log P(Y_i|X_i)$$

将上述所用到的式子不断的带入带入再带入，变换变换再变换，就变成了一个只关于 θ_j 的式子，用随机梯度上升法求出当 θ_j 取何值时式子的值最大。

层次Softmax



采样到 I 的概率 $p(I|context) = (1 - \sigma(\theta_1 h)) * (1 - \sigma(\theta_3 h))$

采样到 eat 的概率 $p(eat|context) = (1 - \sigma(\theta_1 h)) * \sigma(\theta_3 h)$

采样到 to 的概率 $p(to|context) = \sigma(\theta_1 h) * (1 - \sigma(\theta_2 h))$

采样到 like 的概率 $p(like|context) = \sigma(\theta_1 h) * \sigma(\theta_2 h) * (1 - \sigma(\theta_4 h))$

采样到 apple 的概率 $p(apple|context) = \sigma(\theta_1 h) * \sigma(\theta_2 h) * \sigma(\theta_4 h)$

N-gram特征

- n-gram是基于语言模型的算法，基本思想是将文本内容按照子节顺序进行大小为N的窗口滑动操作，最终形成窗口为N的字节片段序列。

字粒度和词粒度

- 我爱深度学习。

- Bigram字粒度：

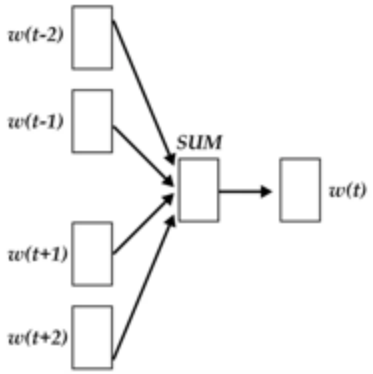
  我爱；爱深；深度；度学；学习

- Bigram词粒度：

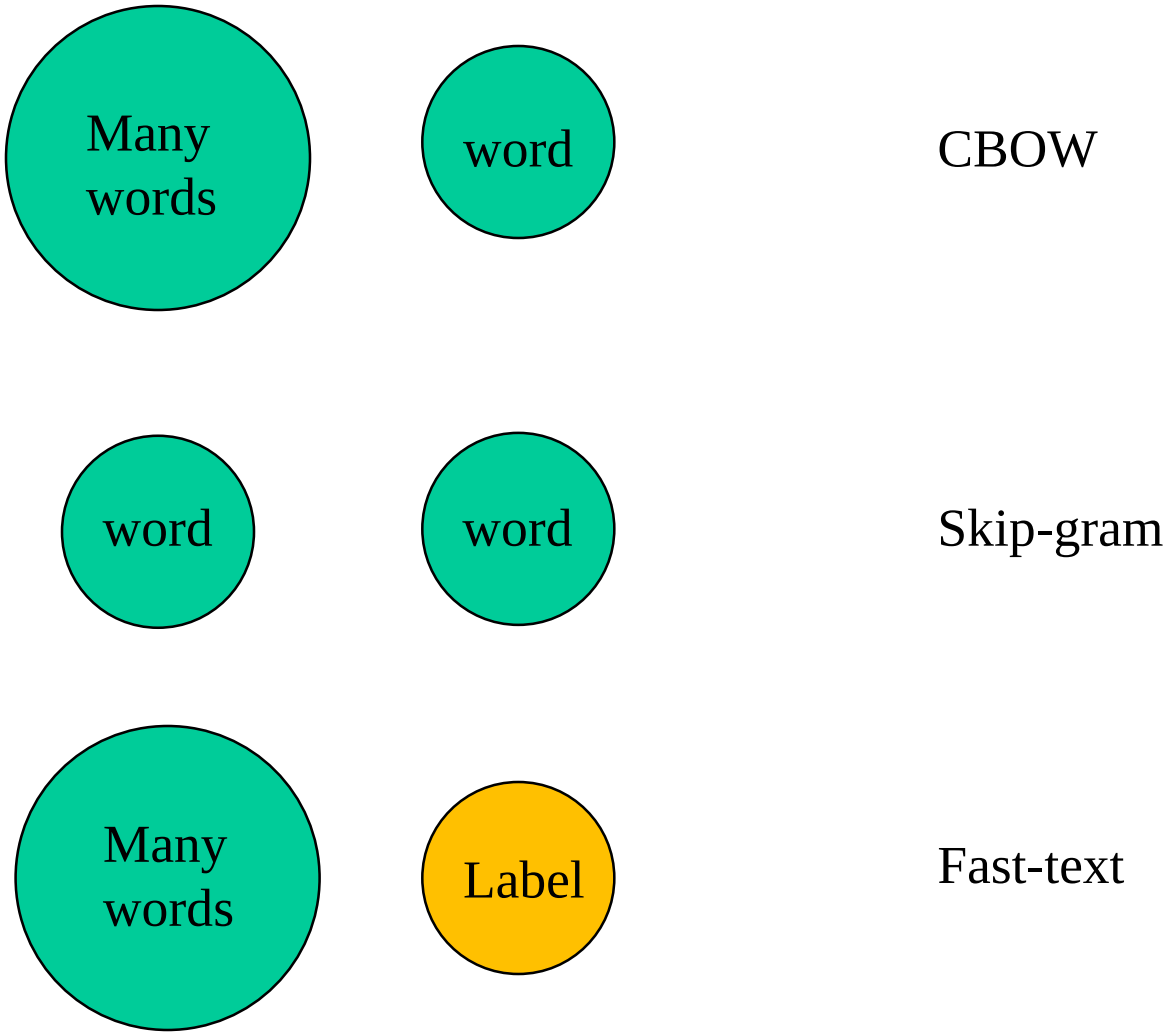
  我爱；爱深度；深度学习

Fast-text

- **FastText是一个开源的、免费的、轻量级的库，允许用户学习文本表示和文本分类器。它适用于标准的通用硬件。模型可以在以后缩小，甚至可以在移动设备上使用。**



Fast-text


Many words
word
CBOW
word
word
Skip-gram
Many words
Label
Fast-text

FastText词向量与word2vec对比

- FastText= word2vec中 cbow + h-softmax的灵活使用
- 模型的输出层：word2vec的输出层，对应的是每一个term，计算某term的概率最大；而fasttext的输出层对应的是分类的label。
- 模型的输入层：word2vec的输入层，是 context window 内的term；而fasttext 对应的整个sentence的内容，包括term，也包括 n-gram的内容；两者本质的不同，体现在 h-softmax的使用。