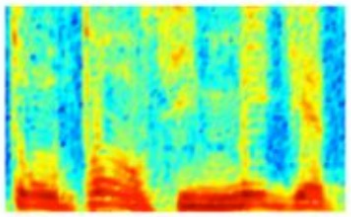
第11章 NLP中的表征基础

9.1 NLP词的表示方法

数据表示是机器学习的核心问题。
DeepNLP的核心关键：语言表示（Representation）。

AUDIO



Audio Spectrogram

DENSE

IMAGES

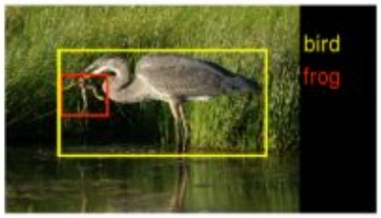
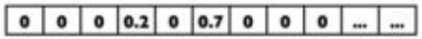


Image pixels

DENSE

TEXT

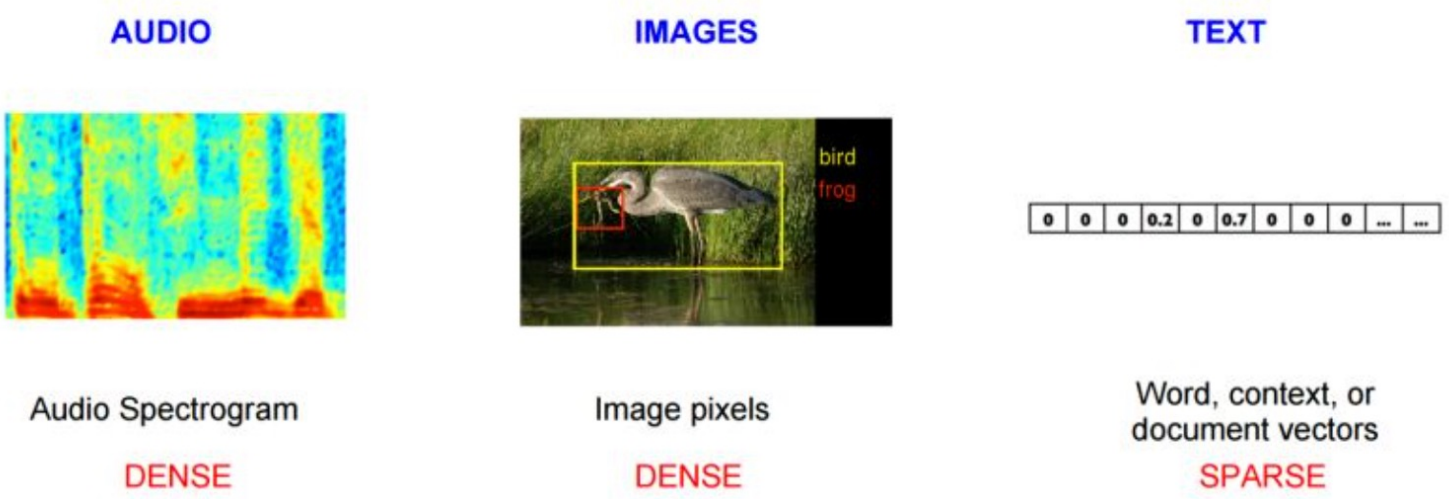


Word, context, or document vectors

SPARSE

其他类型数据是如何表示的？

9.1 NLP词的表示方法



语音：音频频谱序列向量所构成的矩阵作为前端输入

图像：图片的像素构成的矩阵作为输入

NLP：将每一个词用一个向量表示？

“麦克风”和“话筒”？

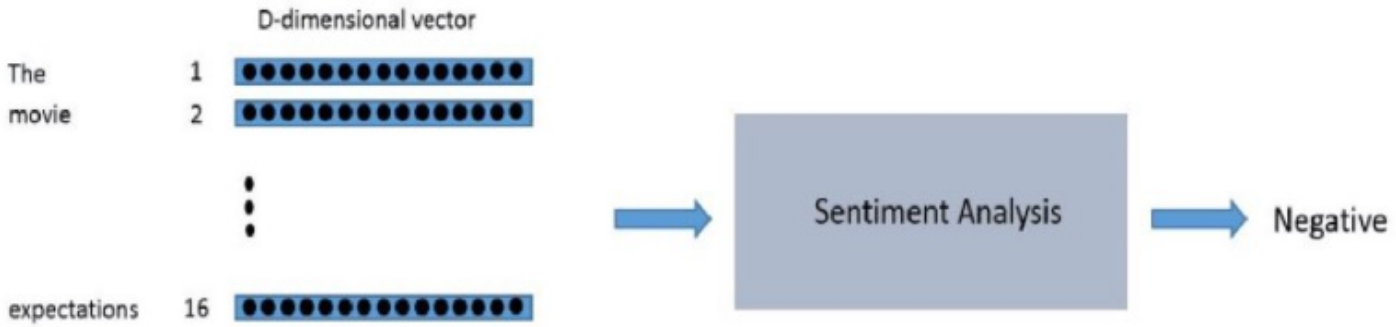
9.1 NLP词的表示方法

处理自然语言时，一般是把词向量作为模型的输入。

将整个句子作为输入，不能满足计算机的一些基本运算操作。

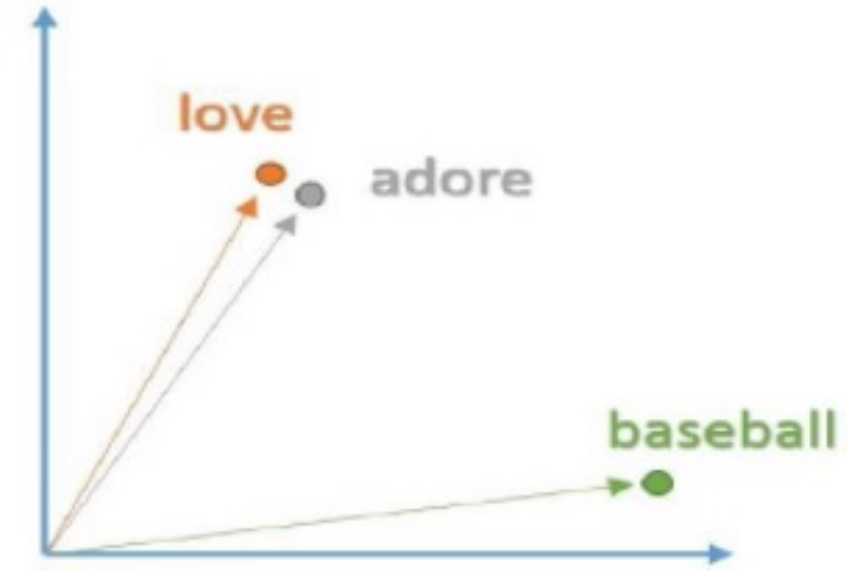
9.1 NLP词的表示方法

将每一个词转换为向量，用词向量作为输入数据。



本例中词向量组成一个16*D的矩阵。

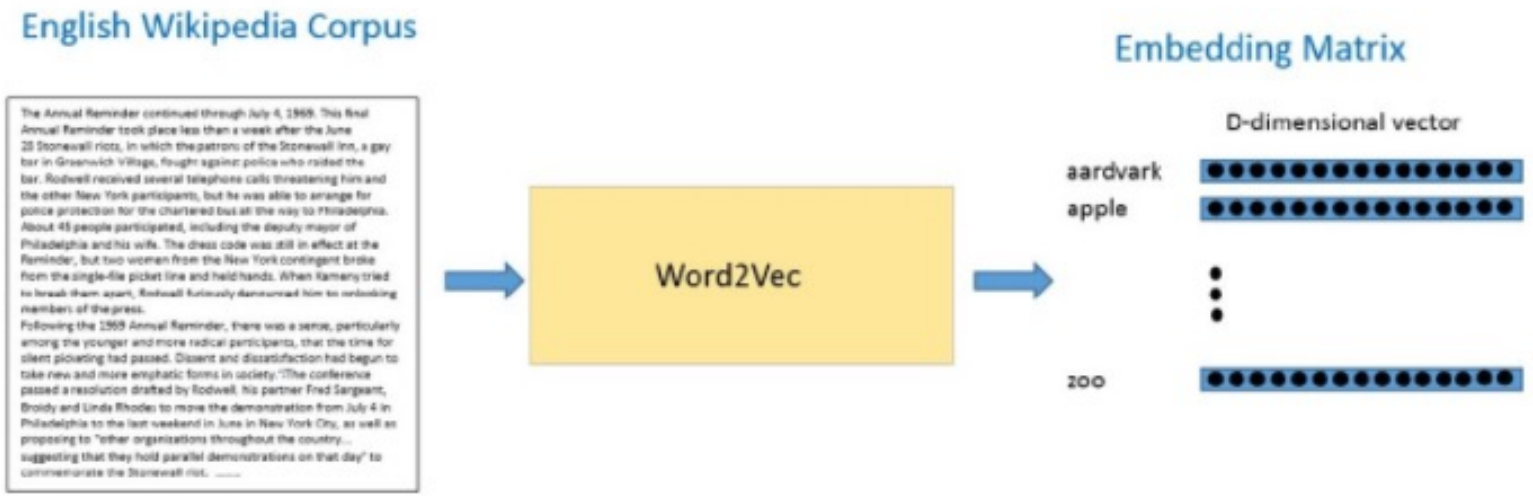
9.1 NLP词的表示方法



词向量不仅表示当前单词，而且可以表示上下文意思。

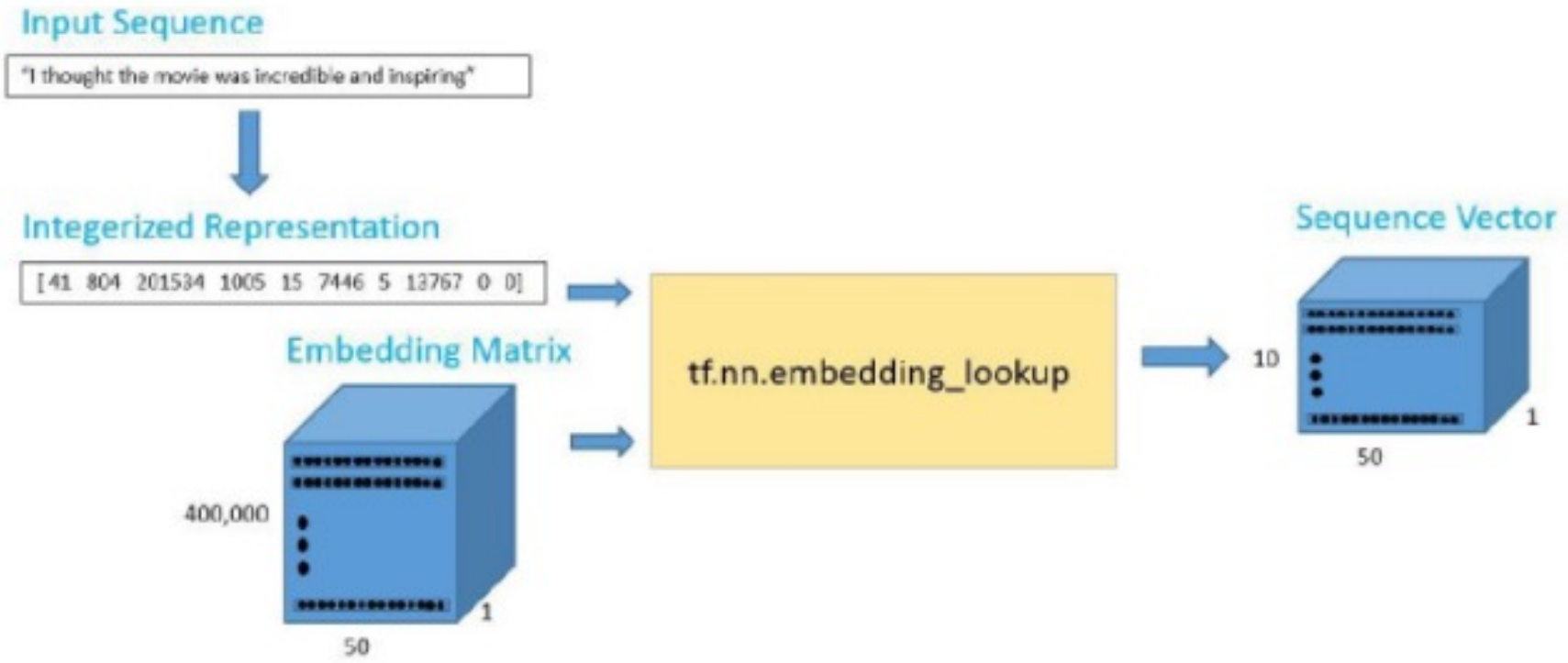
9.1 NLP词的表示方法

Word2Vec的作用是从一堆句子里为每个独一无二的单词进行建模，并且输出一个唯一的向量。Word2Vec模型的输出被称为一个词向量矩阵。



这个词向量矩阵包含训练集中每一个词的一个向量。

9.1 NLP词的表示方法



输出数据是一个50*50的词矩阵，其中包括10个词，每个词的向量维度是50。

9.1 NLP词的表示方法

文本映射：

如果将机器学习算法应用到自然语言文本的处理中，就必须将文本转化为数值。对于实现从文本到数字的转化，其实现过程经过了不断演化，主要方法包括：

1. 词袋模型 bags of words
2. TF-IDF算法
3. 词嵌入方法：Word2Vec为例
4. 其他

9.2 Bags of Words（词袋模型）

One-Hot（“独热”）表示:

One-Hot表示在传统NLP很常见

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

Similarity(dog,cat)=0

把每个词表示为一个很长的向量。这个向量的维度是词表大小，其中绝大多数元素为 0，只有一个维度的值为 1，这个维度就代表了当前的词。

9.2 Bags of Words（词袋模型）

One-Hot的缺点：

1、向量的维度会随着句子的词的数量类型增大而增大；

2、任意两个词之间都是孤立的，根本无法表示出在语义层面上词语词之间的相关信息

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

Similarity(dog,cat)=0

9.2 Bags of Words（词袋模型）

词向量：单词的分布向量表示
（Distributional Representation）

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

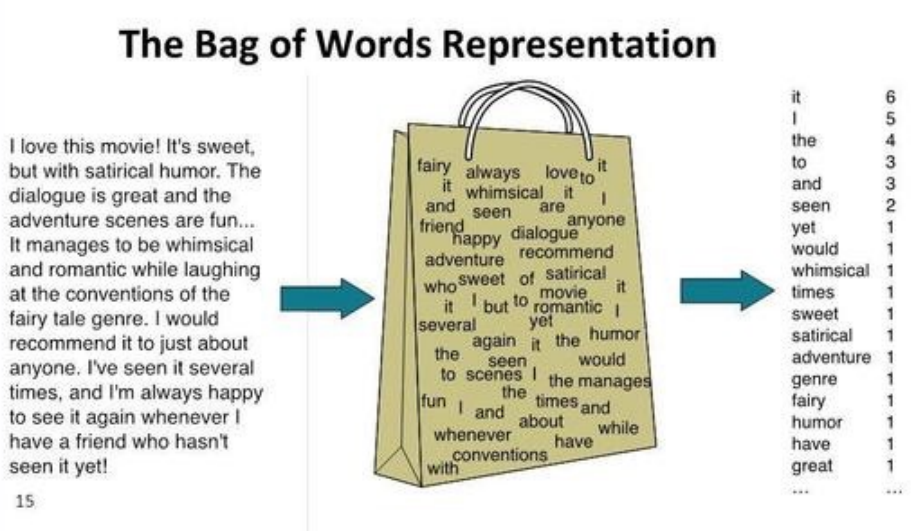
Similarity(dog,cat)>Similarity(dog,the)

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

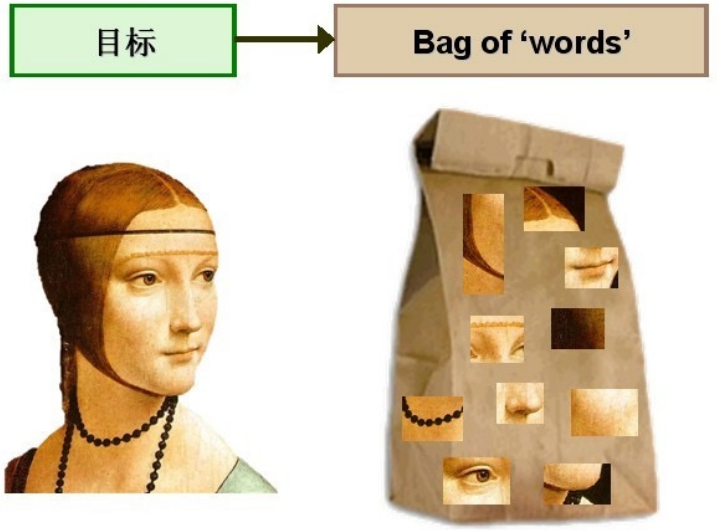
词向量表征了单词使用上下文中的句法语义特征；One-Hot只能实现字面的语义匹配。

9.2 Bags of Words（词袋模型）

词袋模型是将一个文本或文档看做是一袋子单词，不考虑其语法和词序关系，每个词都是独立的，然后对这一袋子单词进行编码。



NLP里的词袋



CV里的词袋

9.2 Bags of Words（词袋模型）

构造词典：

Dictionary = {1:”Bob”, 2. “like”, 3. “to”, 4. “play”, 5. “basketball”, 6. “also”, 7. “football”, 8. “games”, 9. “Jim”, 10. “too”}。

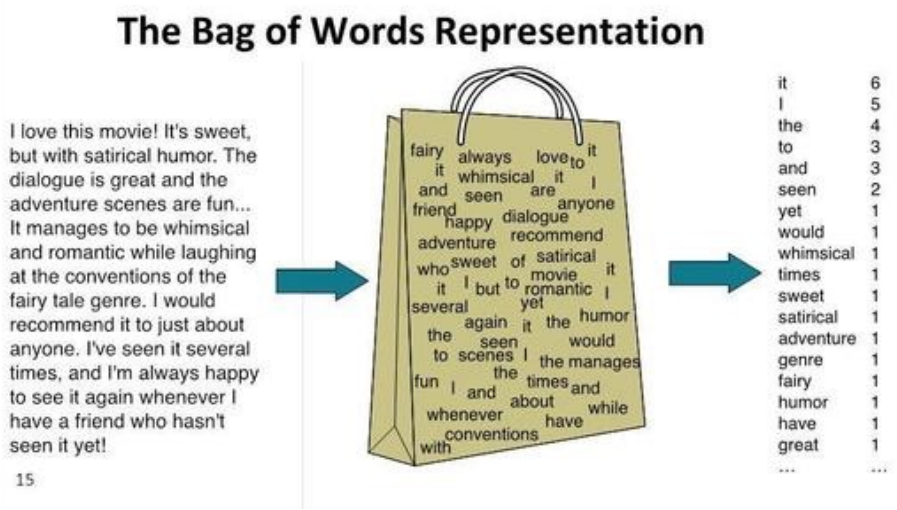
两个文档：

Doc 1：Bob likes to play basketball, Jim likes too.

Doc 2：Bob also likes to play football games.

两个文档的向量表示：

Doc 1：[1, 2, 1, 1, 1, 0, 0, 0, 1, 1]
Doc 2：[1, 1, 1, 1 ,0, 1, 1, 1, 0, 0]



9.2 Bags of Words（词袋模型）

词袋模型处理过程：

1. 构造所有单词的词典
2. 使用该词典，采用每个单词的One-Hot编码对语句进行编码。

词袋模型特征：

1. 词袋模型是没有考虑词语在文本中的上下文之间的相关信息，损失了语句中单词的顺序特征。
2. 对于每一个语句，无论语句长短，都会使用相同词典长度的编码。
3. 每一个单词都具有相同的数值化索引，无法体现单词在语音中的重要性。

9.3 TF-IDF算法

TF-IDF（Term Frequency-Inverse Document Frequency，词频-逆文件频率）算法。利用词频和文件频率来评估一个字词对于一个文件集或一份文件对于一个语料库的重要程度。

词频（TF）：某个单词在文档中出现的频率，根据词频来确定每个单词的权重。

$$TF_w = \frac{\text{在某一类中词条}w\text{出现的次数}}{\text{该类中所有的词条数目}}$$

9.3 TF-IDF算法

IDF（逆文件频率）:对于the、a等通用词，在每一篇文档中出现的频率都非常高，权重降低。而与文档主题有关的单词，仅仅会在某一篇文档中出现的频率较高，权重提高。则计算表示为：

$$IDF = log(\frac{\text{语料库的文档总数}}{\text{包含词条}w\text{的文档数}+1})$$

分母之所以要加1，是为了避免分母为0。

9.3 TF-IDF算法

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的TF-IDF。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语。

$$TF - IDF = TF * IDF$$

9.3 TF-IDF算法

以《中国的蜜蜂养殖》为例，假定文本长度为1000个词，“中国”“蜜蜂”“养殖”各出现20次，则这三个词的“词频TF”都是0.02。

然而，搜索发现，含有“的”字的网页有250亿个，假定这就是中文网页的总数。包含“中国”“蜜蜂”“养殖”的网页各有62.3亿个、0.484亿个和0.973亿个。则他们的TF-IDF为：

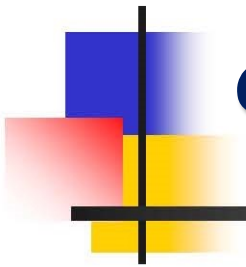
	包含该词的文档数（亿）	IDF	TF-IDF
中国	62.3	0.603	0.0121
蜜蜂	0.484	2.713	0.0543
养殖	0.973	2.410	0.0482

9.3 TF-IDF算法

	包含该词的文档数（亿）	IDF	TF-IDF
中国	62.3	0.603	0.0121
蜜蜂	0.484	2.713	0.0543
养殖	0.973	2.410	0.0482

由上表可知，蜜蜂TF-IDF值最高，“养殖”其次，“中国”最低。所以，如果只选择一个词，“蜜蜂”就是这篇文章的关键词。

问：“的”字的TF-IDF更可能是多少？



9.4 词嵌入（Word Embedding）

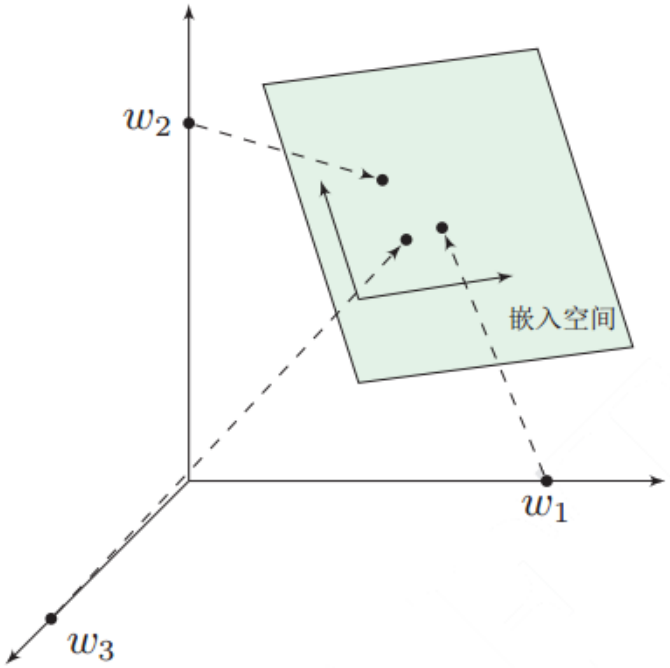
基于神经网络的分布表示又称为词向量、词嵌入。

神经网络词向量模型与其它分布表示方法一样，均基于分布假说，核心依然是上下文的表示以及上下文与目标词之间的关系的建模。

9.4 词嵌入（Word Embedding）

基于One-Hot方法进行改进：

- 向量中每个数值从整型改为浮点型
- 将稀疏的巨大维度压缩到一个更小的维度



9.4 词嵌入（Word Embedding）

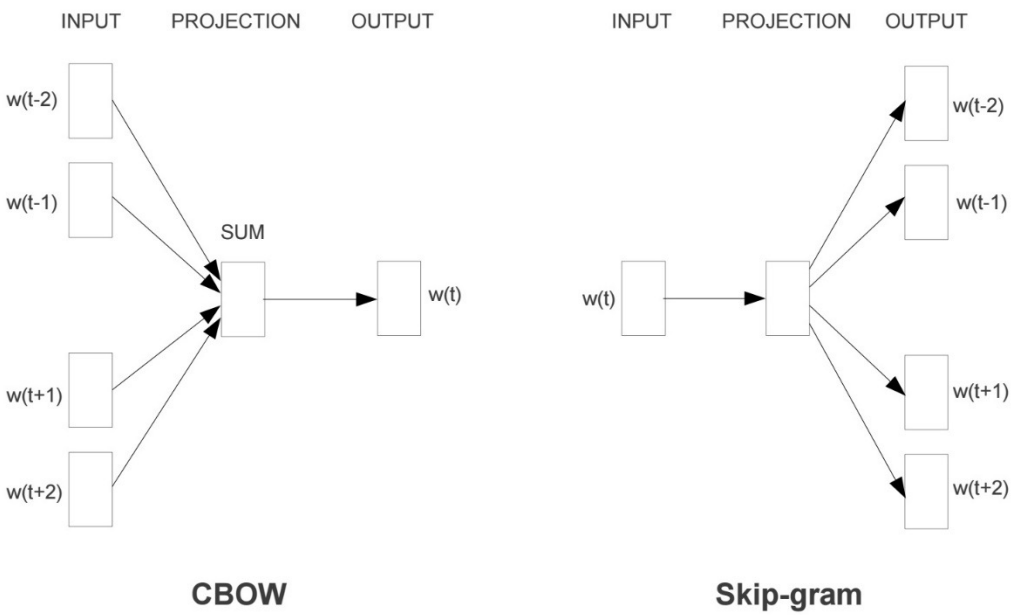
著名的神经网络语言模型：

- Neural Network Language Model ，NNLM
- Log-Bilinear Language Model， LBL
- Recurrent Neural Network based Language Model，RNNLM
- Collobert 和 Weston 在2008 年提出的 C&W 模型)
- Mikolov 等人提出了 CBOW（ Continuous Bagof-Words）和 Skip-gram 模型

9.5 Word2Vec（词嵌入）

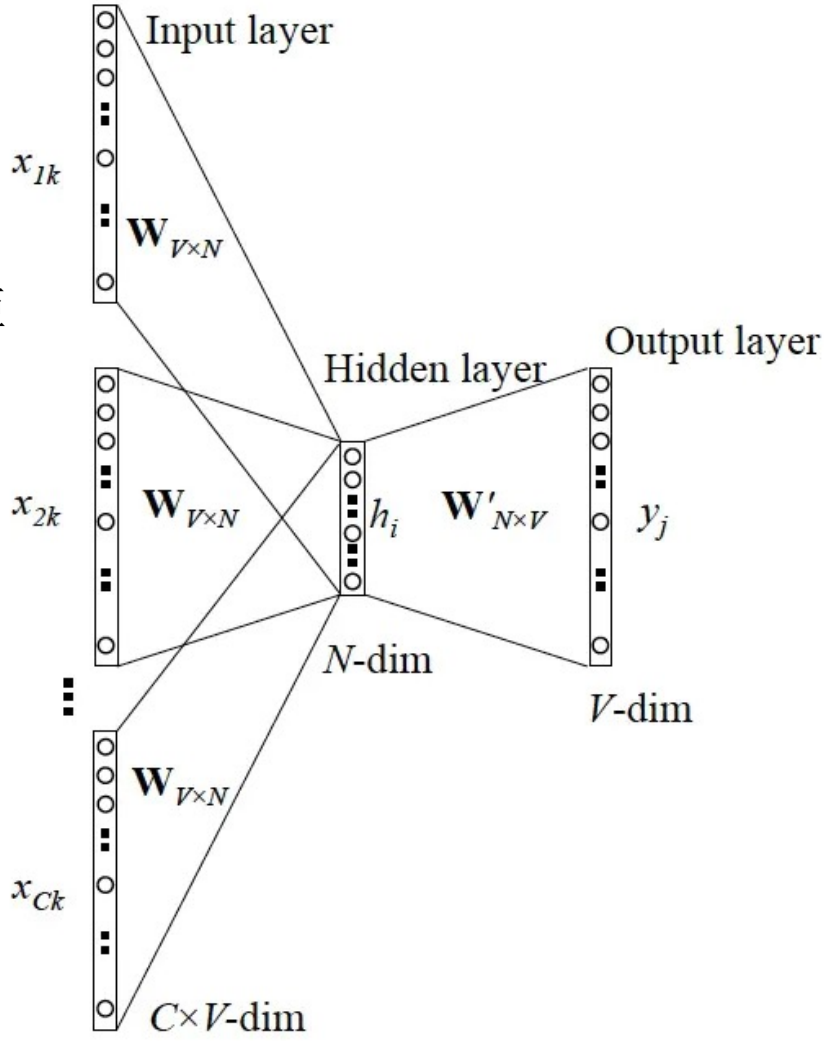
Word2Vec是一种基于神经网络的词分布表示的实现方法，提出两种模型：

- CBOW(Continuous Bag of Words,连续词袋模型）是通过上下文来预测目标词的模型；
- Skip-gram语言模型是从一个词文字来预测上下文的模型。

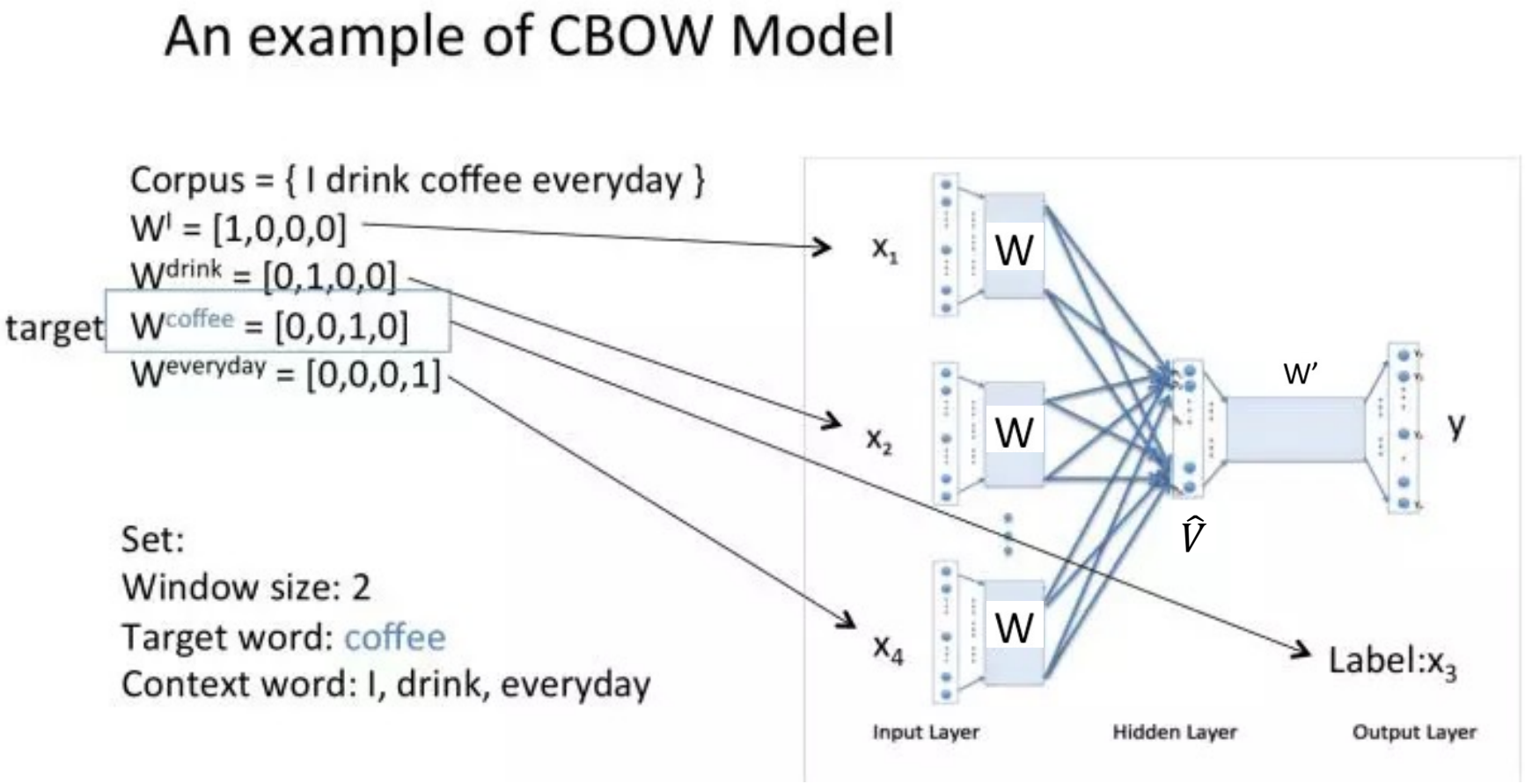


9.5 Word2Vec：CBOW

1. 输入层：上下文单词的one-hot.（假设单词向量空间dim为V，上下文单词个数为C）。
2. 所有one-hot分别乘以共享的输入权重矩阵W.（V*N矩阵，N为自己设定的数，初始化权重矩阵W）。
3. 所得的向量（因为是one-hot所以为向量）相加求平均作为隐层向量，size为1*N。
4. 乘以输出权重矩阵W’{N*V}。
5. 得到向量 {1*V} 激活函数处理得到V-dim概率分布 （one-hot其中的每一维都代表一个单词）。
6. 概率最大的index所指示的单词为预测出的中间词（target word）与true label的one-hot做比较，误差越小越好（根据误差更新权重矩阵）。
7. 训练，用梯度下降算法更新W和W’。



9.5 Word2Vec: CBOW



窗口大小n表示选取前面n个单词和后面n个单词。

9.5 Word2Vec: CBOW

An example of CBOW Model

Corpus = { I drink coffee everyday }

Initialize:

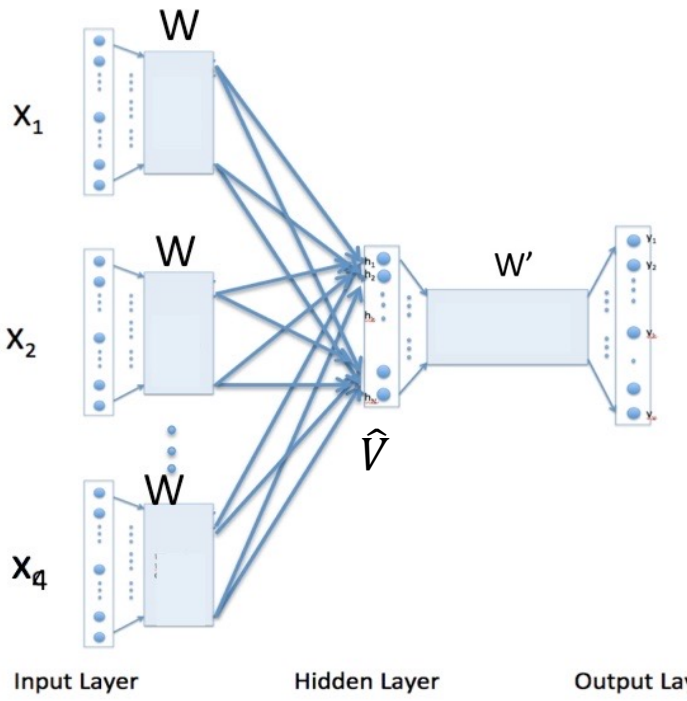
$$W = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 2 & 1 & 2 \\ -1 & 1 & 1 & 1 \end{bmatrix}$$

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)

9.5 Word2Vec: CBOW

An example of CBOW Model

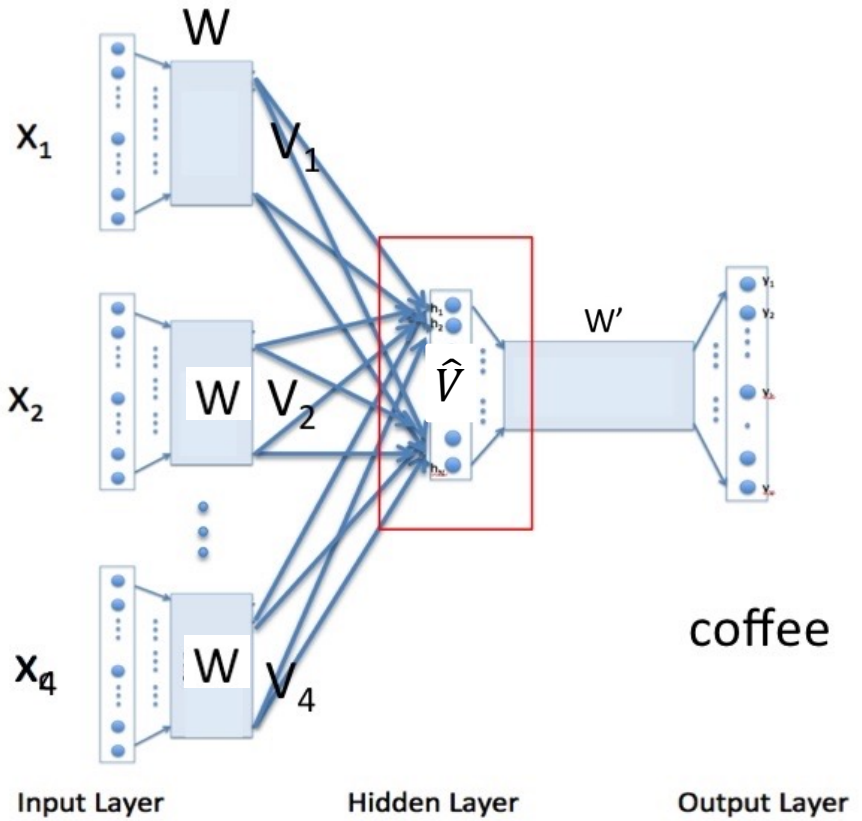
Corpus = { I drink coffee everyday }

Initialize:

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

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

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



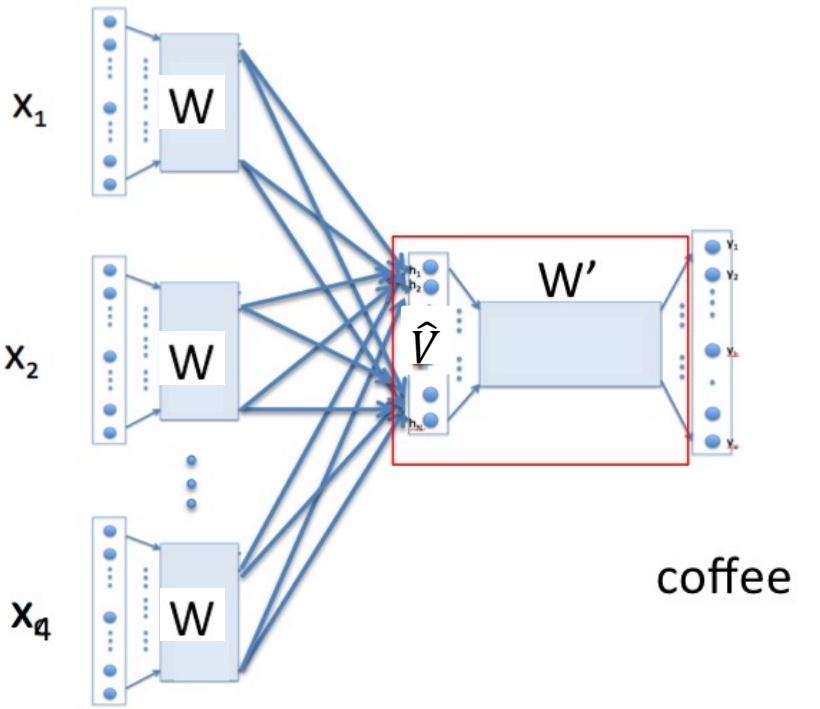
9.5 Word2Vec: CBOW

An example of CBOW Model

Corpus = { I drink coffee everyday }

Initialize:

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



$$W'\hat{V} = U_0$$

$$\begin{bmatrix} 1 & 2 & -1 \\ -1 & 2 & -1 \\ 1 & 2 & 2 \\ 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 1.00 \\ 1.67 \\ 0.33 \end{bmatrix} = \begin{bmatrix} 4.01 \\ 2.01 \\ 5.00 \\ 3.34 \end{bmatrix} \begin{matrix} u_1 \\ u_2 \\ u_c \\ u_4 \end{matrix}$$

9.5 Word2Vec: CBOW

An example of CBOW Model

Output: Probability distribution

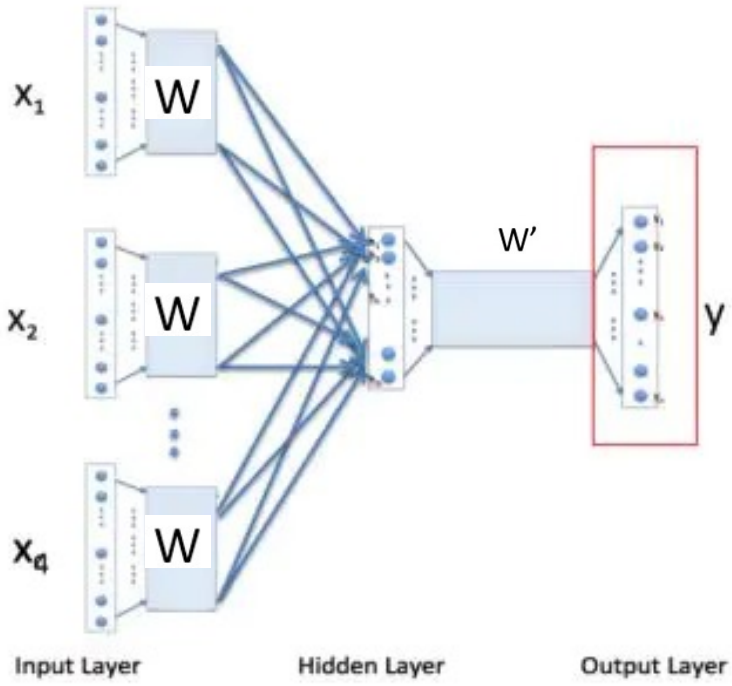
$$softmax(\mathbf{u}_o) = \mathbf{y}$$

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

Probability of "coffee"

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

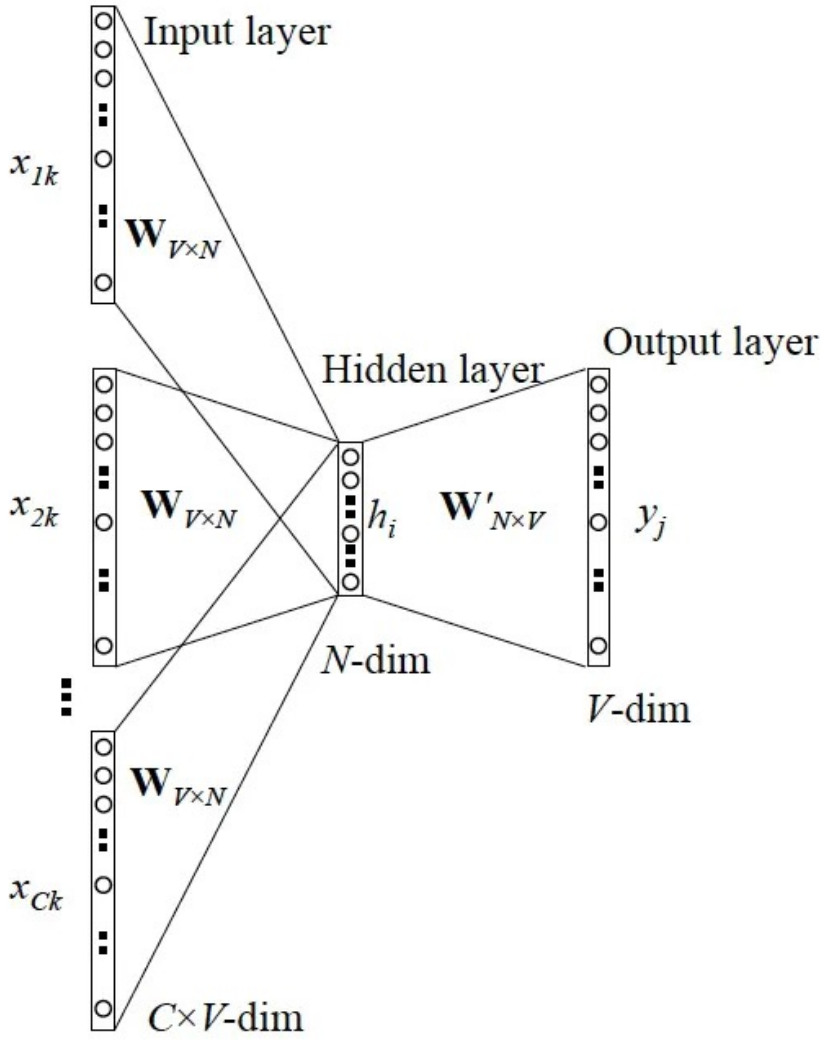
Use gradient descent to update W and W'



9.5 Word2Vec：CBOW

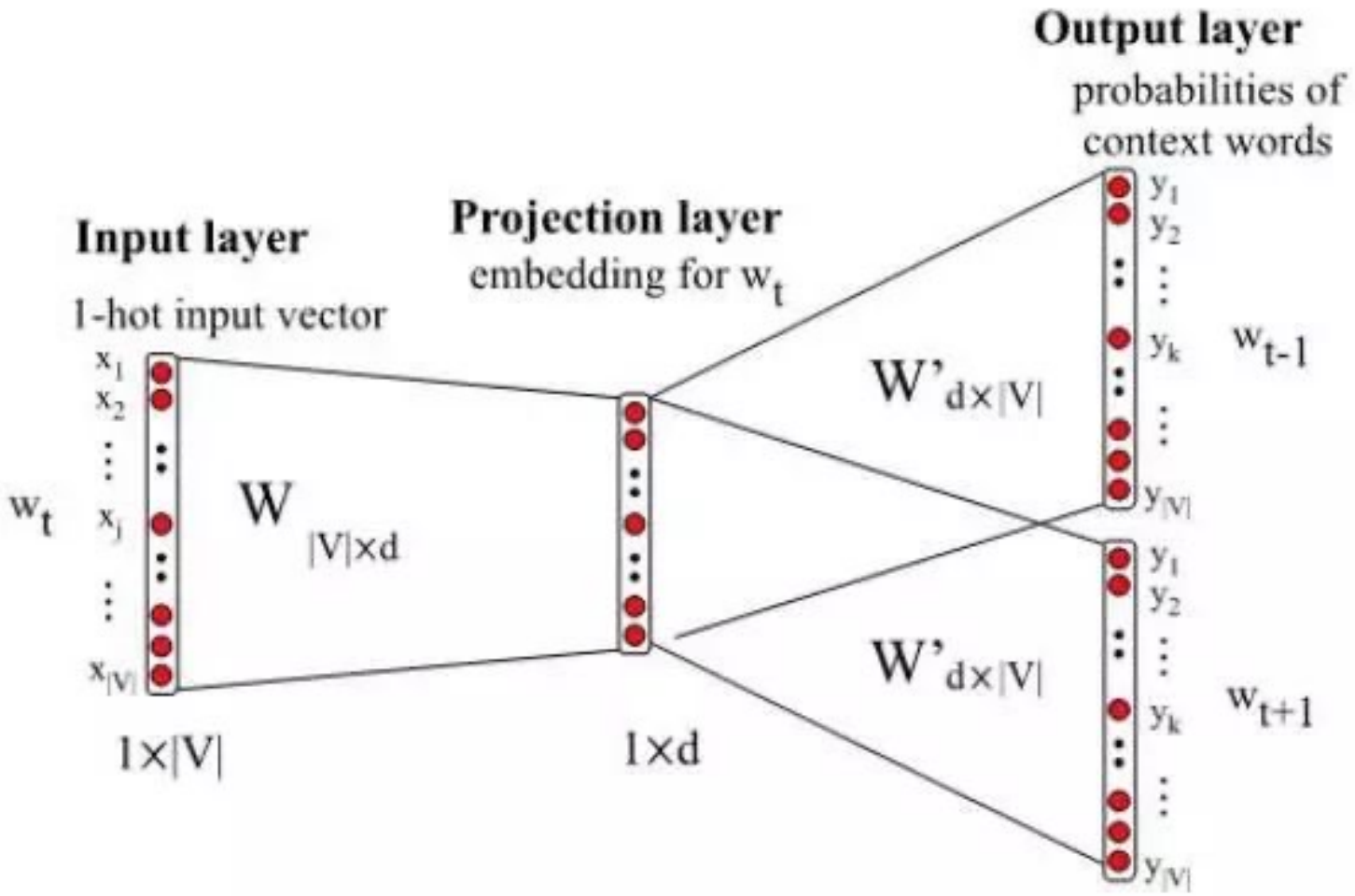
- CBOW

注意输入向量、输入权重矩阵、隐层向量、输出权重矩阵、输出向量维度的变化。



9.5 Word2Vec: Skip-Gram

- Skip-Gram Model

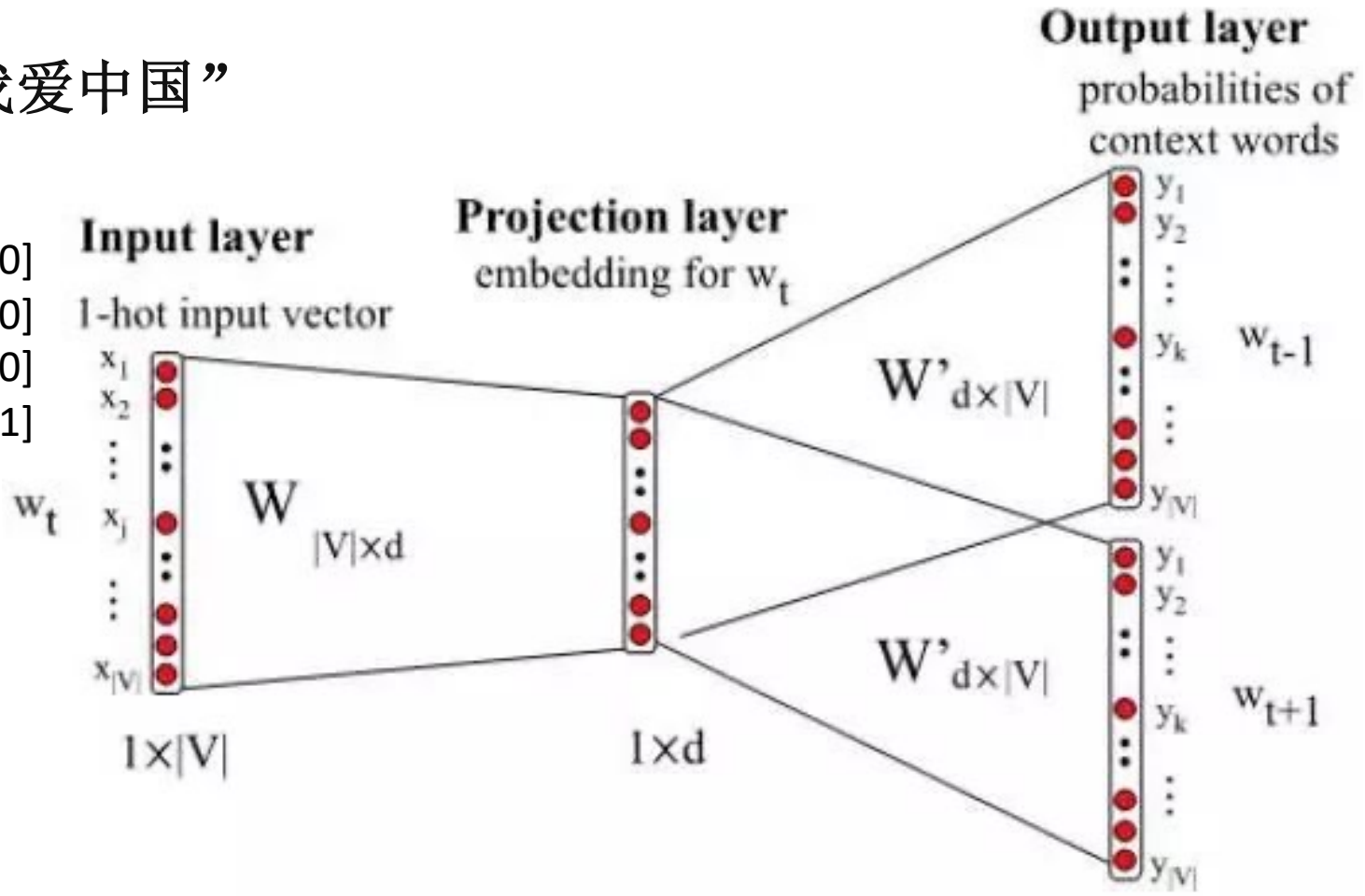


9.5 Word2Vec: Skip-Gram

- Skip-Gram Model

“我爱中国”

W1 = 我 [1,0,0,0]
W2 = 爱 [0,1,0,0]
W3 = 中 [0,0,1,0]
W4 = 国 [0,0,0,1]

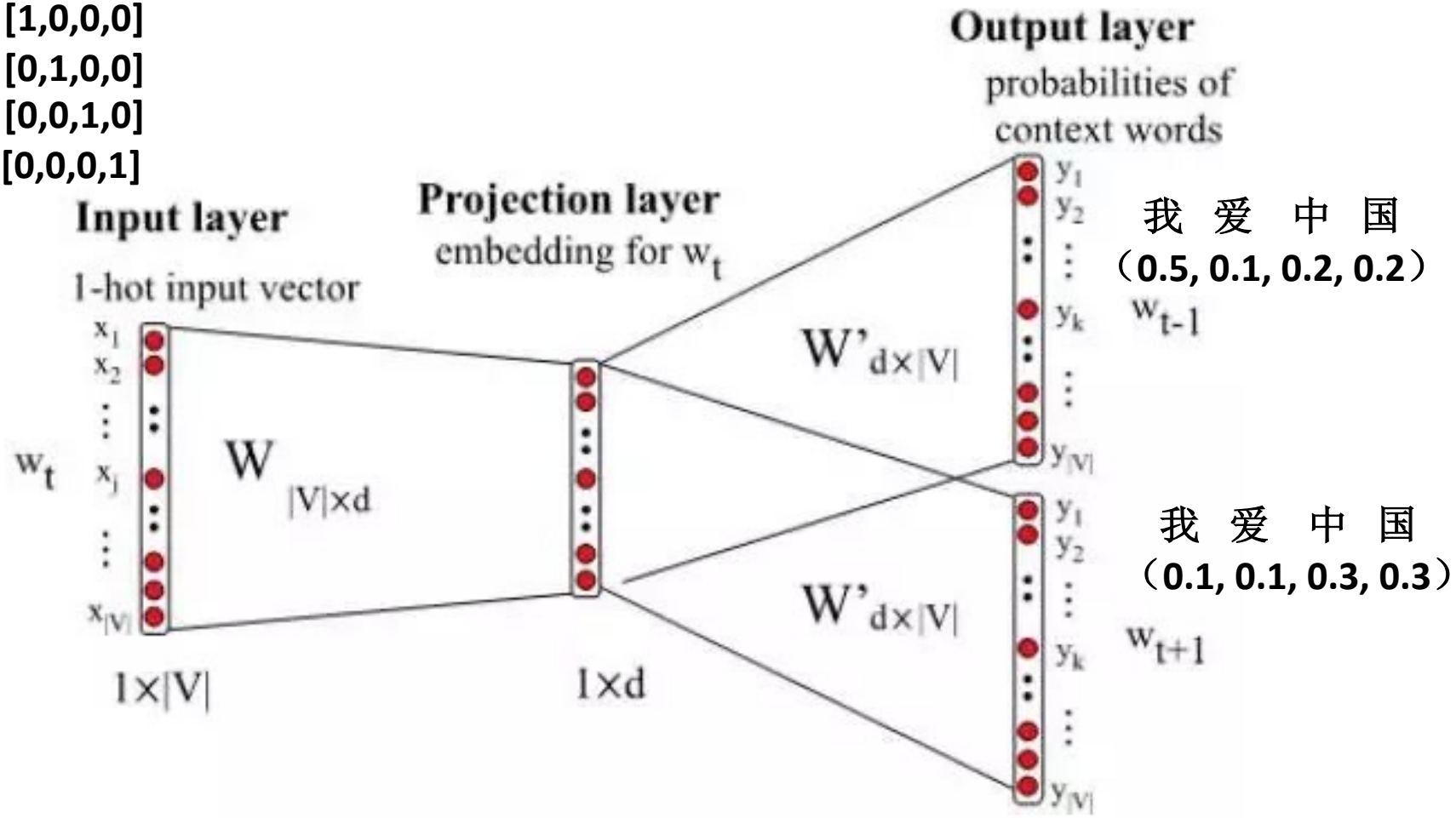


9.5 Word2Vec: Skip-Gram

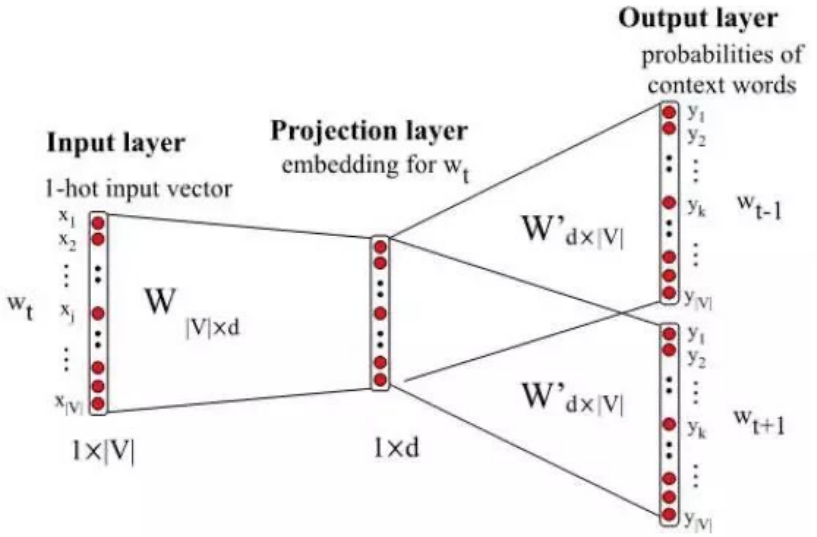
- Skip-Gram Model

W_1 = 我 [1,0,0,0]
W_2 = 爱 [0,1,0,0]
W_3 = 中 [0,0,1,0]
W_4 = 国 [0,0,0,1]

W_t = 爱 [0,1,0,0]



9.5 Word2Vec:



- Skip-Gram Model

优化过程目标函数:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} \log p(w_{t+j}|w_t)$$

T是语料库单词的总个数，p(wt+j|wt)是已知当前词wt，预测周围词的总概率对数值。

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)}$$

9.5 Word2Vec: Skip-Gram

“The dog barked at the mailman”

- Skip-Gram Model

Skip-gram模型是根据中心词预测上下文m个词的算法