干货|咦？还可以这样图解Word2Vec

原创 2017-08-23 Shenglei 机器学习算法与自然语言处理

1 Word2Vec的含义

一个单词，神经网络理解不了，需要人转换成数字再喂给它。**最naive的方式就是one-hot，但是太过于稀疏，不好。**所以在改进一下，把one-hot进一步压缩成一个dense vector。

word2vec算法就是根据上下文预测单词，从而获得词向量矩阵。

预测单词的任务只是一个幌子，我们需要的结果并不是预测出来的单词，而是通过预测单词这个任务，不断更新着的参数矩阵weights。

预测任务由一个简单的三层神经网络来完成，其中有两个参数矩阵V与U，$V \in R^{Dh*|W|}$,$U \in R^{|W|*Dh}$。

V是输入层到隐藏层的矩阵，又被称为look-up table（因为，输入的是one-hot向量，一个one-hot向量乘以一个矩阵相当于取了这个矩阵的其中一列。将其中的每一列看成是词向量）

U是隐藏层到输出层的矩阵，又被称为word representation matrix（将其中的每一行**看成是词向量**）

最后需要的词向量矩阵是将两个词向量矩阵相加 $=V+U^T$，然后每一列就是词向量。

2 两种实现方法

2.1. Skip-Gram

训练任务：根据中心词，预测出上下文词
输入：一个中心词（center word，$x \in R^{|W|*1}$）
参数：一个look up table $V \in R^{Dh*|W|}$，一个word representation matrix $U \in R^{|W|*Dh}$

输出：T个上下文词（context word，$\hat{y} \in \mathbb{R}^{|W|\times 1}$）

损失函数：cross-entropy - $J_t(\theta) = y \log \hat{y}$

详细步骤：

$$v_c = Vx \in \mathbb{R}^{D_h \times 1}$$

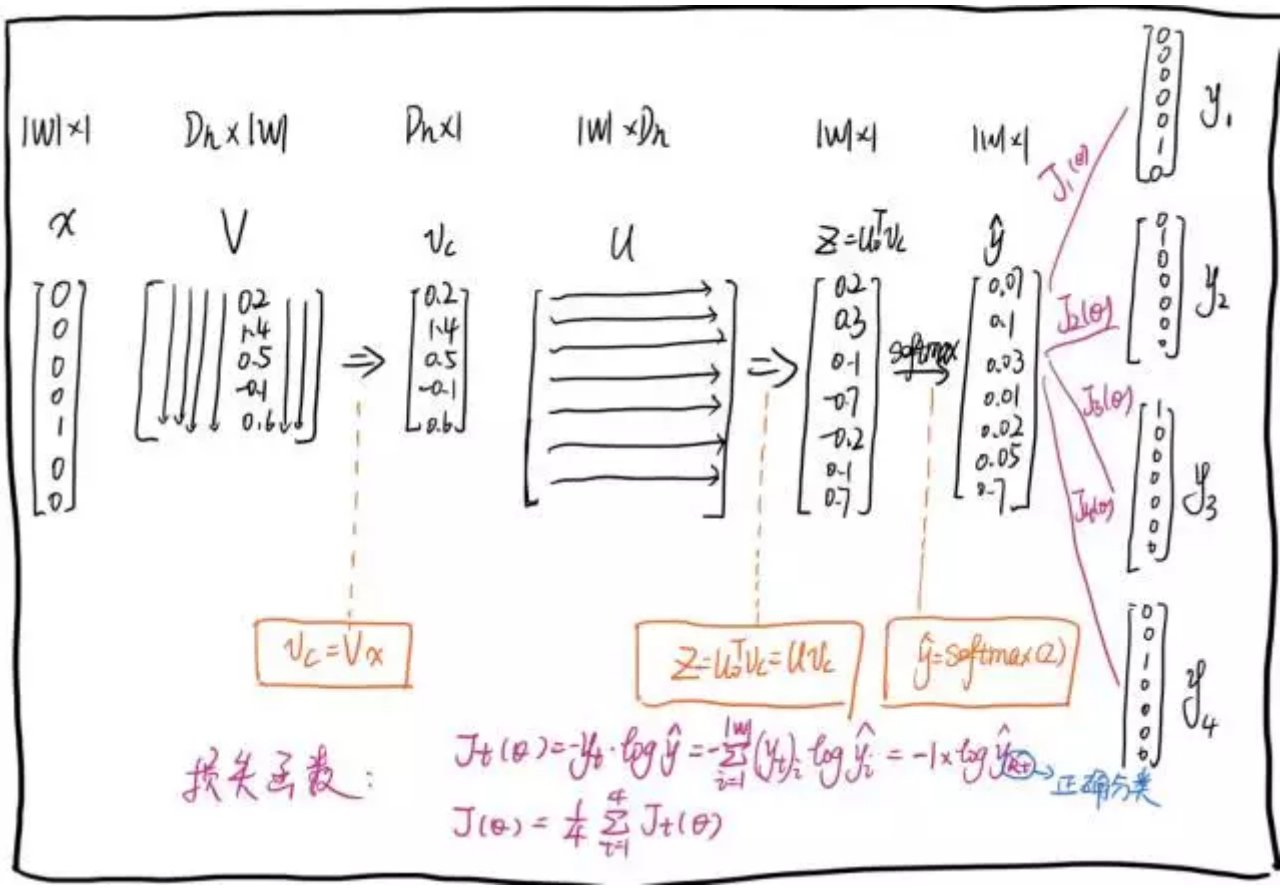
$$z = Uv_c \in \mathbb{R}^{|W|\times 1}$$

$$\hat{y} = softmax(z) \in \mathbb{R}^{|W|\times 1}$$

$$J_t(\theta) = y \log \hat{y}$$

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J_t(\theta)$$

Skip-Gram步骤图：



2.2. CBOW

与Skip-Gram相反，是通过完成上下文词预测中心词的任务来训练词向量的。

训练任务：根据上下文词，预测出中心词

输入：T个上下文词（context word，$x \in \mathbb{R}^{|W|\times 1}$）

参数：一个look up table $V \in \mathbb{R}^{D_h\times|W|}$，一个word representation matrix $U \in \mathbb{R}^{|W|\times D_h}$

输出：一个中心词（center word，$\hat{y} \in \mathbb{R}^{|W|\times 1}$）

损失函数：cross-entropy - $J_t(\theta) = y\log\hat{y}$

详细步骤：

$$v_{ot} = V \cdot x_t \in \mathbb{R}^{D_h \times 1}$$

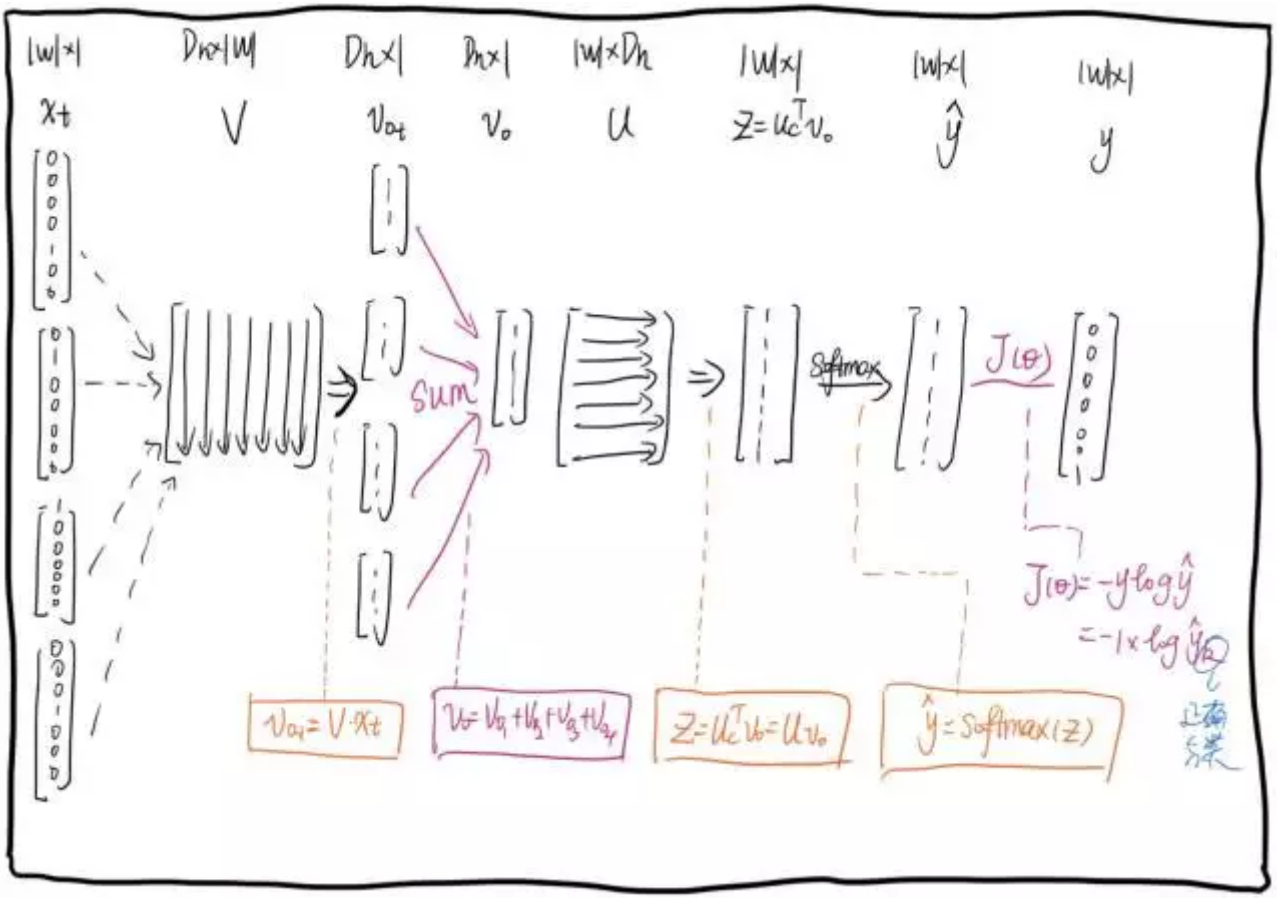
$$v_o = \sum_{t=1}^{T} v_{ot}$$

$$z = U v_o \in \mathbb{R}^{|W| \times 1}$$

$$\hat{y} = softmax(z) \in \mathbb{R}^{|W| \times 1}$$

$$J(\theta) = J_t(\theta) = y \log \hat{y}$$

CBOW步骤图：



推荐阅读：

精选干货|近半年干货目录汇总

干货|台湾大学林轩田机器学习基石课程学习笔记5 -- Training versus Testing

干货|MIT线性代数课程精细笔记[第一课]

欢迎关注公众号学习交流~

长按二维码扫描关注

机器学习算法与自然语言处理

ID: yizhennotes

通俗笔记，分享交流

欢迎加入交流群交流学习

机器学习&nlp

扫一扫二维码，加入该群。