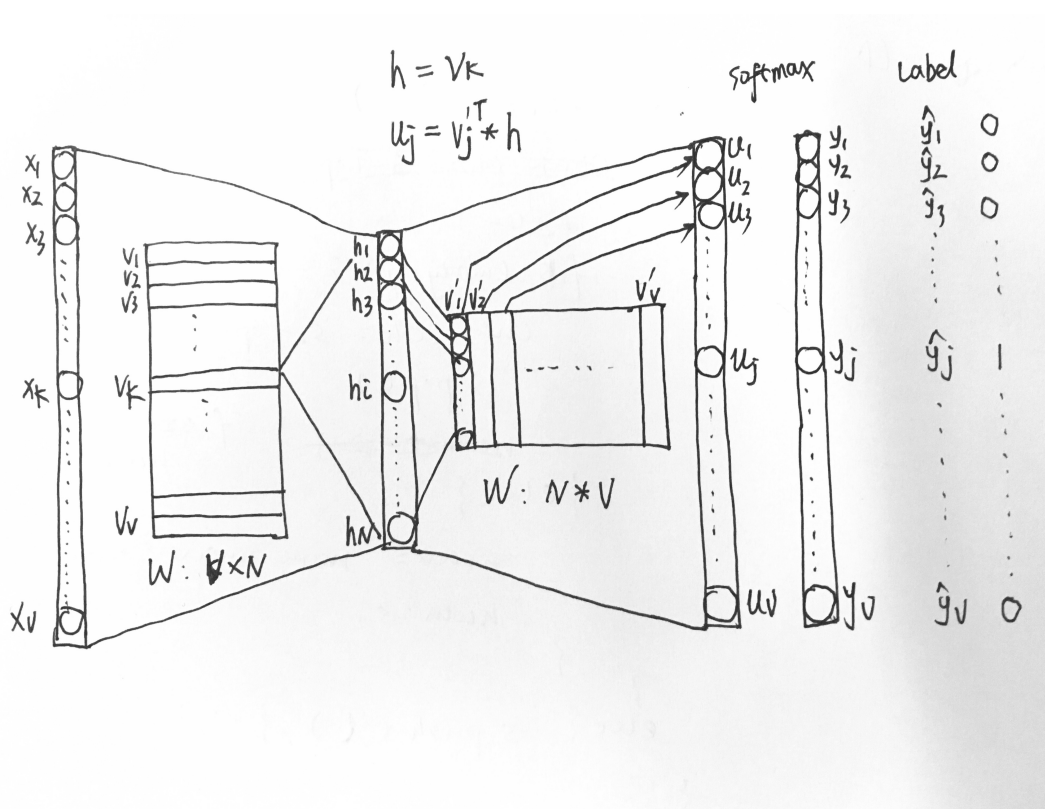


word2vec 数学推导

word2vec 是一种训练词向量的工具，可以将词库中的所有词语映射到 k 维的向量中。主要有两种训练方式，CBOW 和 skip-gram，CBOW 根据上下文来预测目标单词，skip-gram 根据目标单词来预测上下文。下面进行两种方式的数学推导。

CBOW

1. one-word context



假设词表大小为 V ，隐藏层大小为 N ，input-hidden 权重矩阵为 $W_{V \times N}$ ，其中第 k 行的转置为 v_k ，hidden-output 的权重矩阵为 $W'_{N \times V}$ ，其中第 j 列为 v'_j 。计算过程如下：

$$h = v_I$$

$$u_j = v'_j{}^T * h \quad j = 1, 2, \dots, V$$

$$p(w_j | w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^V \exp(u_{j'})}$$

$$E = - \sum_{j=1}^V \hat{y}_j \log y_j = - \sum_{j=1}^V \hat{y}_j (u_j - \log(\sum_{j'=1}^V \exp(u_{j'})))$$

$E u_j$

$$\frac{\partial E}{\partial u_j} = y_j - \hat{y}_j = e_j$$

$E v'_j$

$$\frac{\partial E}{\partial v'_j} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial v'_j} = e_j \cdot h$$

$$v'_j{}^{(new)} = v'_j{}^{(old)} - \eta \cdot e_j \cdot h \quad \text{for } j = 1, 2, \dots, V$$

$$\eta e_j y_j > \hat{y}_j v'_j h v'_j v_I y_j < \hat{y}_j \hat{y}_j = 1 v'_j h$$

E 对 h_i 求导:

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^V \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^V e_j \cdot w'_{ij}$$

$$h_i = \sum_{k=1}^V x_k \cdot w_{ki}$$

E 对 w_{ki} 求导, 得到:

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \sum_{j=1}^V e_j \cdot w'_{ij} \cdot x_k$$

由于 x 只有一个值非零 (that is 1), $\frac{\partial E}{\partial w}$ 中只有一行非零, 上式写成向量形式:

$$\frac{\partial E}{\partial v_k} = \sum_{j=1}^V e_j \cdot v'_j$$

所以, 更新 input-hidden 权重矩阵:

$$v_I^{(new)} = v_I^{(old)} - \eta \cdot \sum_{j=1}^V e_j \cdot v'_j$$

这时候, v_I 会加上一点预测值 v'_j 向量, 减去一点点其它词向量。

2. multi-word context

对于多单词上下文的情况, 计算隐藏层输出时, 不再像单词语上下文那样, 直接复制上下文单词的输入向量, 而是取所有上下文单词的平均值:

$$h = \frac{1}{C} W^T (x_1 + x_2 + \dots + x_C) = \frac{1}{C} (v_{w_1} + v_{w_2} + \dots + v_{w_C})^T$$

其中， C 是输入上下文单词的总数， w_1, w_2, \dots, w_C 是上下文单词， v_w 是单词 w 的输入向量，损失函数同样是交叉熵函数。hidden-output 的权重更新和单词语上下文一致：

$$v_j'^{(new)} = v_j'^{(old)} - \eta \cdot e_j \cdot h \quad \text{for } j = 1, 2, \dots, V$$

input-hidden 权重更新公式和单词语类似，将下面等式应用到每一个输入的上下文单词 $w_{I,c}$ ：

$$v_{I,c}^{(new)} = v_{I,c}^{(old)} - \frac{1}{C} \eta \sum_{j=1}^V e_j \cdot v_j' \quad c = 1, 2, \dots, C$$

skip-gram

skip-gram 模型中，目标单词出现在输入层，上下文单词出现在输出层。输出层是 C 个后验概率：

$$p(w_{c,j} | w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^V \exp(u_{j'})}$$

由于输出层共享权重，因此有：

$$u_{c,j} = u_j = v_j' h \quad c = 1, 2, \dots, C$$

交叉熵函数： $E_c = - \sum_{j=1}^V \hat{y}_{c,j} \log y_{c,j}$

损失函数为 C 个交叉熵函数相加：

$$E = \sum_{c=1}^C E_c = - \sum_{c=1}^C \sum_{j=1}^V \hat{y}_{c,j} \log \frac{\exp(u_{c,j})}{\sum_{j'=1}^V \exp(u_{j'})}$$

E 对 $u_{c,j}$ 求导：

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - \hat{y}_{c,j} := e_{c,j}$$

E 对 v_j' 求导，得到：

$$\frac{\partial E}{\partial v_j'} = \sum_{c=1}^C \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial v_j'} = \sum_{c=1}^C e_{c,j} \cdot h$$

更新 hidden-output 权重矩阵：

$$v_j'^{(new)} = v_j'^{(old)} - \eta \sum_{c=1}^C e_{c,j} \cdot h \quad \text{for } j = 1, 2, \dots, V$$

input-hidden 的权重更新函数和 one-word 的 cbow 类似, e_j 改为 $\sum_{c=1}^C e_{c,j}$, 因此:

$$v_I^{(new)} = v_I^{(old)} - \eta \cdot \sum_{j=1}^V \sum_{c=1}^C e_{c,j} \cdot v_j'$$

模型优化

对于这两种模型, 词汇表中每个单词都有两个向量表示: 输入向量 v_w 和 v_w' , 学习 v_w 很容易, 但学习输出向量很费时间, 为了更新 v_j' , 对每一个训练实例, 需要迭代词汇表中每个单词 w_j , 计算它的输出 u_j 和后验概率 y_j (对 skip-gram, 是 $y_{c,j}$), 然后计算预测误差, 最后更新 v_j' 。为了解决这个问题, 对每个样本, 要限制被更新的输出向量的数量, 一般有两种方法: 一个是 hierarchical softmax, 一个是 negative sampling。