Word2Vec

xbZhong

2025-06-24

本页PDF

Word2Vec

基于**CBOW**或者**skip-gram**来计算词向量矩阵，主要目的是**为了得到词向量**

中心词和上下文词用一个**窗口来维护**

词嵌入向量：$v = W \times one_hot(v)$

其中：W是$Word2Vec$中的权重输入矩阵

CBOW

核心：用**上下文词预测中心词**

模型结构

输入是**上下文**文本单词的`one-hot`向量，通过**线性变换**压缩成一个单词向量，再通过一次线性变换得到**一个单词得分表**，最后经过**多分类**得到要预测的单词。

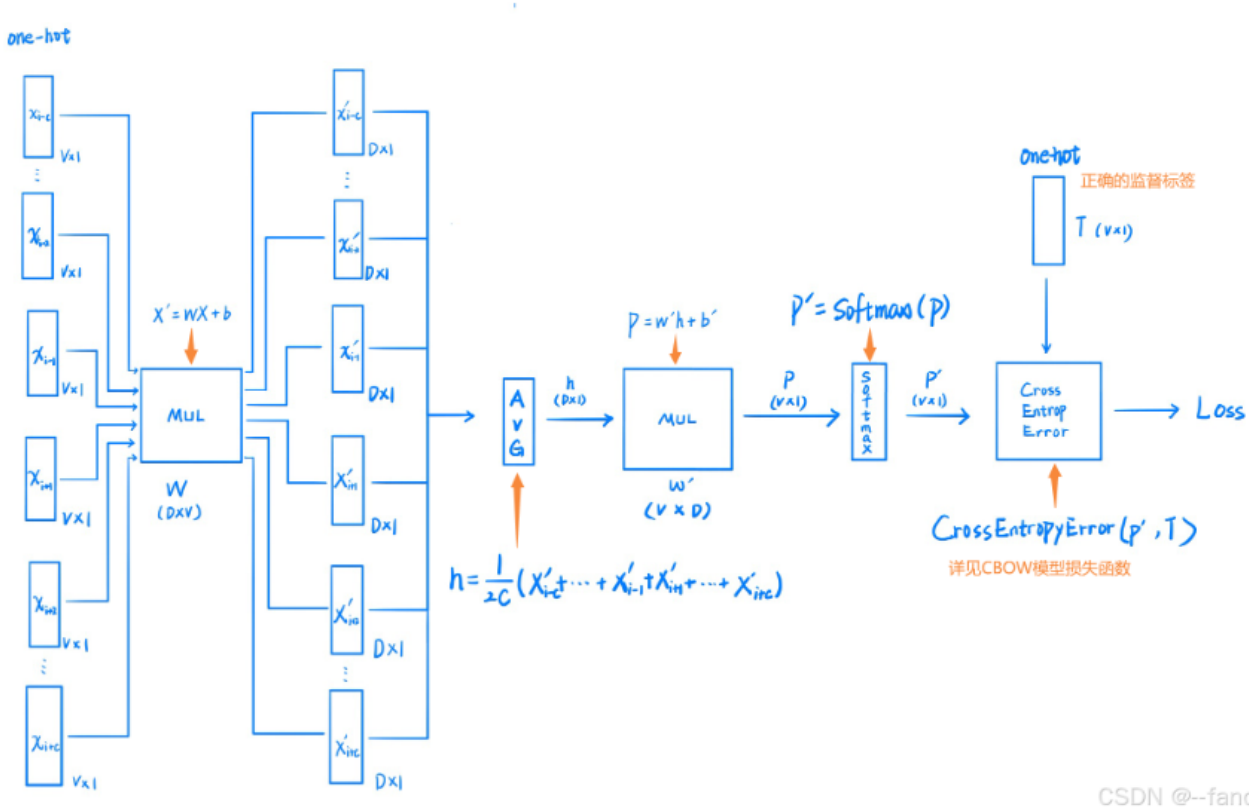

image-20250705224627824

模型输入

上下文（输入）词的多少取决于窗口大小C，因此**输入**为

$$X = (x_{i-c}, x_{i-c+1}, \ldots, x_{i-1}, x_{i+1} \ldots, x_{i+c}) \in \mathbb{R}^{V \times 2C}$$

其中，x_i为目标单词，$x_i \in \mathbb{R}^{V \times 1}$

权重输入层

将目标单词x_i的单热编码与隐藏层的输入权重W相乘再加上偏置$b \in \mathbb{R}^{D \times 1}$得到$x'_j$，即$X'_j = WX_j + b$，写成矩阵形式

$$X' = WX + b$$

其中

$$X = [x_{i-c}, x_{i-c+1}, \ldots, x_{i-1}, x_{i+1} \ldots, x_{i+c}] X' = [x'_{i-c}, x'_{i-c+1}, \ldots, x'_{i-1}, x'_{i+1} \ldots, x'_{i+c}]$$

加权平均层

将输入层得到的所有X'_j进行加权平均得到h

$$h = \sum_{j=i-C, j \neq i}^{i+C} \frac{1}{2C}(x'_{i-c} + x'_{i-c+1} + x'_{i-1} + x'_{i+1} + x'_{i+c})$$

权重输出层

将得到的h作线性变换得到每个单词得分的向量P，$P = (P_1, P_2, \ldots, P_V)^T$，$P_i \in R$表示为位置索引为$i$处的**单词得分**

$$P = W'h + b'$$

Softmax层

将输出层得到的得分用Softmax处理为概率P'，$P' = (p'_1, p'_2, \ldots, p'_V)$，$p'_i$表示位置索引为i处的单词概率

$$p'_i = softmax(p_i) = \frac{exp(p_i)}{\sum_{k=1}^{V} exp(p_k)}$$

模型的输出是在P'中取出**最大概率对应位置**的值设为1，其他位置设为0，得到一个单热编码。这个单**热编码对应的词就是模型作为预测结果的词**。

损失函数

用的是**交叉熵损失（Cross Entropy）**，交叉熵损失的输入是Softmax层计算得到的概率向量P'和正确的监督标签T，其中$P' = (P'_1, P'_2, \ldots, P'_V)^T$，正确的监督标签$T = (t_1, t_2, \ldots, t_V)$就是**正确答案单词的单热编码**。

$$Loss = -\sum_{i=1}^{V} t_i \log(P'_i)$$

skip-gram

核心：用**中心词预测上下文词**

模型结构

模型输入时**目标单词的单热编码**，通过线性变换形成预测上下文单词的向量，再通过一次线性变换得到每一个上下文单词的**得分表**，最后经过**多分类**得到要预测的**上下文单词**。

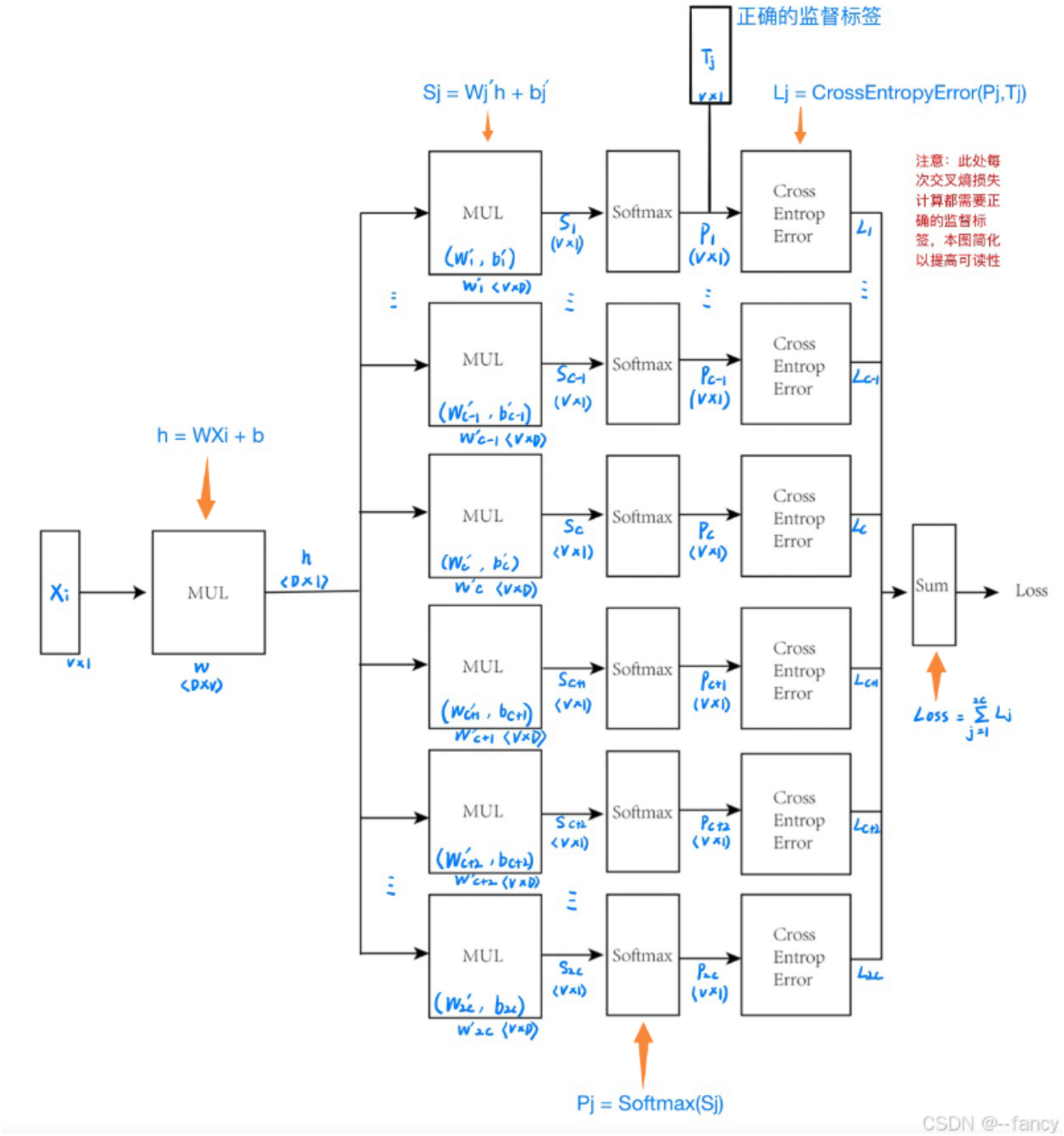

image-20250705224657637

模型输入

将目标词表示为单热编码，作为模型输入 $x_i \in \mathbb{R}^{V\times 1}$，$i$ 表示目标单词所在位置。

权重输入层

将目标词的单热编码 $x_i \in \mathbb{R}^{V\times 1}$ 作线性变换得到隐藏层向量h

$$h = Wx_i + b$$

权重输出层

将得到的h与隐藏层的权重输出矩阵W_j'相乘再加上偏置项b_j'得到多个上下文单词得分的向量$S_j \in \mathbb{R}^{V\times 1}$，$S=(S_1,S_2,\dots,S_{2C})^T$。

$$S_j = W_j'h + b_j'$$

Softmax层

将输出层得到的得分S_j用softmax处理为概率P_j，$P_j=(P_j(0),P_j(1),\dots,P_j(V-1))^T$

$$P_j(k) = Softmax(S_j) = \frac{exp(S_j(k))}{\sum_{l=0}^{V-1} exp(S_j(l))}$$

模型输出

模型输出是在P中取出最大概率对应位置的值设为1，其他设为0，得到一个单热编码，这个单热编码对应的词就是预测上下文单词的结果。

损失函数

使用交叉熵损失（Cross Entropy）进行计算，其输入是概率向量$P_j(k)$，和正确的监督标签T，其中$P_j=(P_1,P_2,\dots,P_{2C})$，正确的监督标签$T_j=[t_j(1),t_j(2),\dots,t_j(V)]$是**正确答案**单词的单热编码。

$$L_j = -\sum_{k=1}^{V} t_k \log(P_j(k)) Loss = \sum_{j=1}^{2C} L_j$$

优化

正常使用word2vec模型，我们是要预测V个单词出现的概率，但是语料库十分巨大，这么做肯定不现实，因此我们可以使用**Huffman树**来加速，把运算次数压缩到$\log V$。

分层Softmax

基本思想：将词典中的每个词按照词频大小构建出一颗Huffman树，**词频大的词处于浅层，词频小的词处于深层**。

- 叶子节点都是词
- 非叶子节点具有**要学习的参数**，是一个**二分类器，全部都接受相同的输入（上下文向量）**

如图所示：

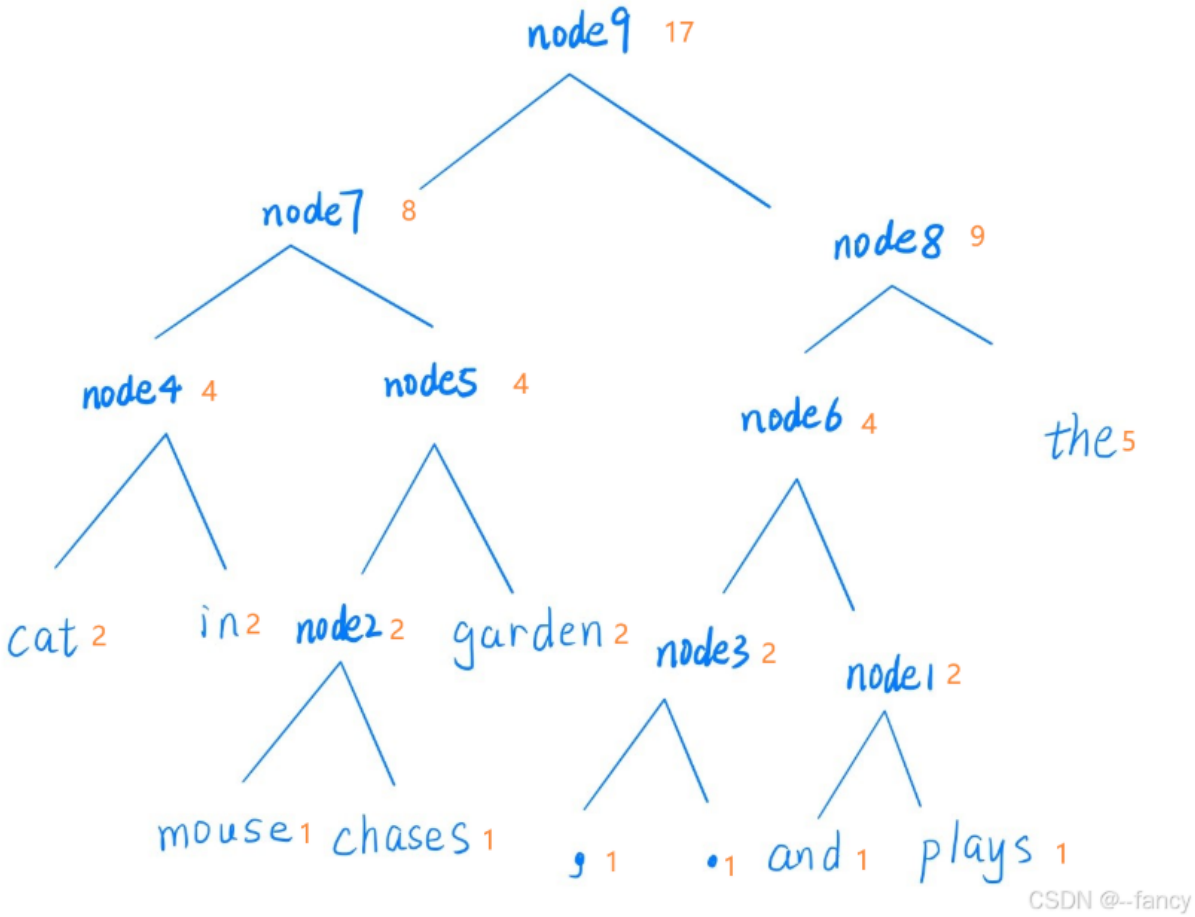

image-20250705162047281

计算概率的步骤：

1. 已知每个非叶子节点具有**网络参数**，可以用网络参数算出并**使用softmax**算出来一个概率值P，称为正向概率
2. 因为哈夫曼树是二叉树，知道一个分支的概率P，我们就可以算出**另外一个分支**的概率为$1-P$
3. 按上述步骤不断进行，我们就可以算出每个非叶子节点每个分支的概率
4. 根据上述算出来的值，可以通过**概率连乘**算出每个词的概率

负采样

正样本：真实有效的上下文本对，（中心词，上下文词）

负样本：从词汇表中随机选择一些不相关的词作为负样本。通过负样本来训练模型，使模型学习到区分正样本和负样本的能力

基本思想：通过从词汇表中随机选择一些“负样本”来代替计算所有可能的上下文词，从而大幅度**降低计算复杂度**。将**多分类问题转化为二分类问题**，让模型对**正样本预测的概率逼近1**，对负**样本预测的概率逼近0**。

损失函数

这里重点讲一下损失函数，因为前面的流程基本都差不多

正样本损失

我们可以知道T_1是正确的标签，$T_1=(t_1,t_2,\dots,t_V)^T$，$P$是每个单词对应的概率，$P=(P_1,P_2,\dots,P_V)$，在这里进行优化。在$T$中，只有**正确索引位置为1**，进行损失函数计算时，只保留正确索引位置单词的得分概率，因此我们将**得分向量**中正确的得分**直接取出**，即$S_1=(\theta_1 h)^T T_1$

随后直接将得分转换为概率，此处是二分类问题，用sigmoid函数，最后应用于Cross Entropy

$$P_1 = \sigma(S_1) = \sigma((\theta_1 h)^T T_1) = \frac{1}{1+e^{-((\theta_1 h + b_1')^T T_1)}} Loss_+ = -log(P_1) = -log(\frac{1}{1+e^{-((\theta_1 h + b_1')^T T_1)}})$$

负样本损失

首先要进行**负采样**，按照词频给出每个单词的概率分布

$$f(w) = \frac{[count(w)]^{\frac{3}{4}}}{\sum_{i=1}^{V}[count(i)]^{\frac{3}{4}}}$$

其中，$count(index)$计算索引位置为index位置单词的词频，w表示目标单词的索引，V为词汇表的大小。

接着按照**概率分布进行采样**，若抽取到正例则重新采样。数据量大，负样本个数k通常为5，数据量小，负样本个数通常为5~20个。

对于采样出的负样本， 我们计算对应的得分**之后**将其取**负号**再使用$Sigmoid$函数，然后使用原来计算正样本的方式进行计算。

我们将负样本权重输出矩阵θ_0与隐藏层的向量h相乘得到单词得分向量，随后依次去除每个负样本对于**索引位置单词的得分**然后**取负号**

$$S_{0,i} = -(\theta_0 h + b_0')^T T_{0,i}$$

其中，$T_{0,i}$为负样本对应的标签，然后使用Sigmoid函数转换为概率

$$P_1 = \sigma(S_{0,i})$$

使用交叉熵计算损失

$$Loss_- = \sum_{i=1}^{k} \log(P_{0,i})$$

最后将$Loss_+$与$Loss_-$相加得到总的损失$Loss$

$$Loss = Loss_- + Loss_+$$