HXH 2022.10.25

负采样和分层softmax都是为了减少训练代价

一、负采样（negative sampling） — 核心就是减少计算代价

1. 思想

in order to deal with the difficulty of having too many output vectors that need to be updated per iteration, we only update a sample of them.

2. 例子

当训练样本（input word: "fox"，output word: "quick"）来训练我们的神经网络时，“fox”和“quick”都是经过one-hot编码的。如果vocabulary大小为10000时，在输出层，我们期望对应“quick”单词的那个神经元结点输出1，其余9999个都应该输出0。这9999个我们期望输出为0的神经元结点所对应的单词我们称为“negative” word。使用负采样时，我们将随机选择一小部分的negative words（比如选5个negative words）来更新对应的权重。我们也会对我们的“positive” word进行权重更新（在我们上面的例子中，这个单词指的是”quick“）。

positive

3. 具体数学原理

首先我们直接写出word2vec的目标函数，假设有一句话：$\text{query} = w_1, w_2, w_3, .., w_n$，由n个词组成的一句话，我们需要最大化窗口中上下文词的概率：

$$\arg\max_{\theta} \prod_{w \in \text{query}} \prod_{c \in c(w)} P(c|w;\theta)$$

这里的$c(w)$表示中心词的context words，我们在计算的时候，可以把相乘的元素转换成对数函数：

$$\arg\max_{\theta} \sum_{w \in \text{query}} \sum_{c \in c(w)} \log P(c|w;\theta)$$

softmax

我们把概率函数可以进行展开就可以得到：

$$\arg\max_{\theta} \sum_{w \in query} \sum_{c \in c(w)} \log \frac{e^{u_c \cdot v_w}}{\sum_{c' \in vocab} e^{u_{c'} \cdot v_w}}$$

这个式子可以表示成：

$$\arg\max_{\theta} \sum_{w \in query} \sum_{c \in c(w)} (e^{u_c \cdot v_w} - \log \sum_{c' \in vocab} e^{u_{c'} \cdot v_w})$$

要遍历整个词库，复杂度非常高 → 简化运算

我们可以看到这个式子第二项，因为c'要遍历整个词库，所以复杂度非常高，所以我们要简化这一步的计算，减小运算的复杂度。这里的u_c表示c的上下文向量，v_w表示中心词w的向量。

为了减小上述表达式的复杂度，我们不妨改变一下上述概率的表达方式，原来的 $p(w_i|w_j)$ 表示以 w_j 为中心词的时候 w_i 出现的概率，这里我们用 $p(D=1|w_i, w_j; \theta)$ 表示 w_i 和 w_j 作为上下文词出现的概率，$p(D=0|w_i, w_j; \theta)$ 表示 w_i 和 w_j 不作为上下文词出现的概率。

由上述新的表达式可以写出新的目标函数：

$$\arg\max_{\theta} \prod_{(w,c) \in D} p(D=1|w,c;\theta) \prod_{(w,c) \in \tilde{D}} p(D=0|w,c;\theta)$$

这里的 D 表示上下文词的集合，$\tilde{D}$ 表示非上下文的集合，我们来举一个例子，这里有一句话：“川建国同志是一名优秀的党员”，这句话分词去停之后变成：**川建国 同志 一名 优秀 党员。**那么 D 表示上下文集合，我们假设 window size为1，那么可以写出：

D = {(川建国，同志)，(同志，川建国)，(同志，一名)，(一名，同志)，(一名，优秀)，(优秀，一名)，(优秀，党员)}

$\tilde{D}$ = {(川建国，一名)，(川建国，优秀)，(川建国，党员)，(同志，优秀)，(同志，党员)，(一名，川建国)，(一名，党员)，(优秀，川建国)，(优秀，同志)，(党员，川建国)，(党员，同志)，(党员，一名)}。

上述的 D 表示正样本，$\tilde{D}$ 表示负样本。我们可以继续表示上述的目标函数，我们可以吧正负样本的概率表示成sigmoid的表达形式：

$$\arg\max_{\theta} \prod_{(w,c) \in D} \frac{1}{1+e^{-u_c \cdot v_w}} \prod_{(w,c) \in \tilde{D}} (1 - \frac{1}{1+e^{-u_c \cdot v_w}}) = \arg\max_{\theta} \sum_{(w,c) \in D} \log \sigma(u_c \cdot v_w) + \sum_{(w,c) \in \tilde{D}} \log \sigma(-u_c \cdot v_w)$$

$\sigma = \frac{1}{1+e^{-x}}$

在词库数量级为10^5的时候，正样本加负样本$\tilde{D}$的数量级可以达到10^{10}左右，里面绝大部分都是负样本，所以我们需要降低负样本计算中的时间复杂度，这就是**Negative Sampling 负采样**的核心。负采样就是对于一个中心词，我们从中心词对应的负样本中随机抽取几组来做梯度下降。还是川建国的例子，对于正样本（川建国，同志），我们随机抽取负样本（川建国，一名），（川建国，党员）来做训练，不用全部的负样本都拿来训练，这就是负采样，减小了计算的复杂度。所以，上述的目标函数可以写成：

$$\approx \arg\max_{\theta} \sum_{(w,c) \in D} [\log \sigma(u_c \cdot v_w) + \sum_{c' \in N(w)} \log \sigma(-u_{c'} \cdot v_w)]$$

采n个负样本

从上述表达式可以看出，负样本我们不需要取所有的都拿来训练，我们只需要每个中心词抽几个负样本就可以了，这样可以大大降低计算的复杂度。这就是word2vec训练过程中的Negative Sampling 负采样技巧，可以大大减小梯度下降的时间复杂度，这就有点像SGD随机梯度下降，就是随机一个样本进行梯度下降，大体的方向还是朝着最低点下降。

PS：参考CSDN博主：
深圳湾刘能.
<<Negative Sampling
负采样详解>>