CS224n： Assignment2， written part

[from Cris Lee work]

版权为原作者所有，进行了一定更正，仅为学习交流使用，请勿用于其他用途，翻译仅供参考

[qusetion website A2作业链接]

Variables notation

Attention: All the variables’ dimensions here are consistent with the code part in Assignment 2 for easy understanding.

U , matrix of shape (vocab_size,embedding_dim) ,all the ‘outside’ vectors .

V, matrix of shape (vocab_size,embedding_dim) ,all the ‘center’ vectors .

y, vector of shape (vocab_size,1), the true empirical distribution **y** is a one-hot vector with a 1 for the true outside word o, and 0 everywhere else .

$\hat{\boldsymbol{y}}$, vector of shape (vocab_size,1), the predicted distribution $\hat{\boldsymbol{y}}$ is the probability distribution$ P(O|C = c)$ given by our model .

question a

> (a) (3 points) Show that the naive-softmax loss given in Equation (2) is the same as the cross-entropy loss
> between **y** and $\hat{y}$; i.e., show that

> 给定**y** and $\hat{y}$等，证明了方程(2)中给出的naive-softmax loss等于cross-entropy loss(交叉熵损失函数)

Given outside word o and context word c.

The distribution of **y** is as follows:

$$y_w = \begin{cases} 1 & \text{w=o} \\ 0 & \text{w!=o} \end{cases}$$

$$-\sum_{w=1}^{V} y_w log(\hat{y_w}) = -y_o log(\hat{y}_o) = -log(\hat{y}_o)$$

Here , V represents the vocab_size.

question b

> Compute the partial derivative of $J_{naive-softmax}(v_c, o, U)$ with respect to v_c .
> Please write your
> answer in terms of **y** and $\hat{y}$ and **U**.

> 计算$J_{naive-softmax}(v_c, o, U)$ 对于v_c的偏微分，最后结果用**y** and $\hat{y}$ 和**U**表示。

$$\begin{aligned} &\frac{\partial J_{naive-softmax}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{v}_c} \\ &= -\frac{\partial log(P(O=o|C=c))}{\partial \boldsymbol{v}_c} \\ &= -\frac{\partial log(exp(\boldsymbol{u}_o^T \boldsymbol{v}_c))}{\partial \boldsymbol{v}_c} + \frac{\partial log(\sum_{w=1}^{V} exp(\boldsymbol{u}_w^T \boldsymbol{v}_c))}{\partial \boldsymbol{v}_c} \\ &= -\boldsymbol{u}_o + \sum_{w=1}^{V} \frac{exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)}{\sum_{w=1}^{V} exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)} \boldsymbol{u}_w \\ &= -\boldsymbol{u}_o + \sum_{w=1}^{V} P(O=w|C=c)\boldsymbol{u}_w \\ &= \boldsymbol{U}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) \end{aligned}$$

question c

> Compute the partial derivatives of $J_{naive-softmax}(v_c, o, U)$ with respect to each of the ‘outside’
> word vectors, u_w’s. There will be two cases: when w = o, the true ‘outside’ word vector, and $w \neq o$, for
> all other words. Please write you answer in terms of **y,** $\hat{y}$ **and** v_c .

> 计算$J_{naive-softmax}(v_c, o, U)$ 对于u_w的偏微分，最后结果用**y,** $\hat{y}$ **和** v_c 表示。这里有2种情况，当w = o时和$w \neq o$。

$$\frac{\partial J_{naive-softmax}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_w}$$
$$= -\frac{\partial log(exp(\boldsymbol{u}_o^T \boldsymbol{v}_c))}{\partial \boldsymbol{u}_w} + \frac{\partial log(\sum_{w=1}^{V} exp(\boldsymbol{u}_w^T \boldsymbol{v}_c))}{\partial \boldsymbol{u}_w}$$

when w = o,

$$\frac{\partial J_{naive-softmax}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_w}$$
$$= -\boldsymbol{v}_c + \frac{1}{\sum_{w=1}^{V} exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)} \frac{\partial \sum_{w=1}^{V} exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)}{\partial \boldsymbol{u}_o}$$
$$= -\boldsymbol{v}_c + \frac{1}{\sum_{w=1}^{V} exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)} \frac{\partial exp(\boldsymbol{u}_o^T \boldsymbol{v}_c)}{\partial \boldsymbol{u}_o}$$
$$= -\boldsymbol{v}_c + \frac{exp(\boldsymbol{u}_o^T \boldsymbol{v}_c)}{\sum_{w=1}^{V} exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)} \boldsymbol{v}_c$$
$$= (P(O = o|C = c) - 1))\boldsymbol{v}_c$$

when w != o,

$$\frac{\partial J_{naive-softmax}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_w}$$
$$= \frac{exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)}{\sum_{w=1}^{V} exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)} \boldsymbol{v}_c$$
$$= P(O = w|C = c)\boldsymbol{v}_c$$

In summary,

$$\frac{\partial J_{naive-softmax}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{U}}$$
$$= (\hat{\boldsymbol{y}} - \boldsymbol{y})^T \boldsymbol{v}_c$$

question d

> The sigmoid function is given by Equation 4:
> $
>
> $$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$
>
> $
> Please compute the derivative of σ(x) with respect to x, where x is a vector.

> 公式4给出了 sigmoid函数:，请计算σ(x)对 x 的导数，其中 x 是一个向量。

$$\frac{\partial\sigma(x)}{\partial x} = \frac{\partial\frac{e^x}{e^x+1}}{\partial x} = \frac{e^x(e^x+1) - e^x e^x}{(e^x+1)^2}$$
$$= \frac{e^x}{(e^x+1)^2} = \sigma(x)(1-\sigma(x))$$

question e

> Now we shall consider the Negative Sampling loss, which is an alternative to the Naive
> Softmax loss. Assume that K negative samples (words) are drawn from the vocabulary. For simplicity
> of notation we shall refer to them as $w_1, w_2, \ldots, w_K$ and their outside vectors as $u_1, \ldots, u_K$. Note that $o \notin w_1, \ldots, w_K$. For a center word c and an outside word o, the negative sampling loss function is given by:
> 现在我们来考虑Negative Sampling (负取采样) 损失函数，它是 Naive Softmax loss 的一种替代方法。 假设 k 负样本(词)是从词汇表中抽取的。为了简单起见我们将它们称为$w_1, w_2, \ldots, w_K$，它们的outside vectors为$u_1, \ldots, u_K$。注意$o \notin w_1, \ldots, w_K$.对于中心词 c 和外部词 o，负抽样损失函数函数如下:

> $$J_{neg-sample}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -log(\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c)) - \sum_{k=1}^{K} log(\sigma(-u_k^T v_c))$$

> for a sample $w_1, w_2, \ldots, w_K$, where σ(·) is the sigmoid function.
>
> Please repeat parts （b） and （c） , computing the partial derivatives of $J_{neg-sample}$ with respect to v_c , with respect to u_o , and with respect to a negative sample u_k. Please write your answers in terms of the vectors u o , v c , and u k , where k ∈ [1,K]. After you've done this, describe with one sentence why this loss function is much more efficient to compute than the naive-softmax loss. Note, you should be able to use your solution to part (d) to help compute the necessary gradients here.

> 请重复部分（b） 和（c），计算关于v_c，关于 u_o，关于负样本u_k的 $J_{neg-sample}$ 偏导数。 请根据向量 u_o，v_c 和 $u_k 写出你的答案，其中 k ∈[1，k]。 在你做完这些之后，用一句话描述为什么这个损失函数比Naive Softmax loss计算更有效率。注意，您应该能够使用第(d)部分的解决方案来帮助计算这里所需的梯度。

$$\frac{\partial J_{neg-sample}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{v}_c}$$
$$= \frac{\partial(-log(\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c)) - \sum_{k=1}^{K} log(\sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c)))}{\partial \boldsymbol{v}_c}$$
$$= -\frac{\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c)(1-\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c))}{\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c)} \frac{\partial \boldsymbol{u}_o^T \boldsymbol{v}_c}{\partial \boldsymbol{v}_c} - \sum_{k=1}^{K} \frac{\partial log(\sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c))}{\partial \boldsymbol{v}_c}$$
$$= -(1-\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c))\boldsymbol{u}_o + \sum_{k=1}^{K}(1-\sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c))\boldsymbol{u}_k$$

$$\frac{\partial J_{neg-sample}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_o}$$
$$= \frac{\partial(-log(\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c))}{\partial \boldsymbol{u}_o} = -(1-\sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c))\boldsymbol{v}_c$$

$$\frac{\partial J_{neg-sample}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_k}$$
$$= \frac{\partial(-log(\sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c))}{\partial \boldsymbol{u}_k} = (1-\sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c))\boldsymbol{v}_c$$

describe why this loss function is much more efficient :

这个损失函数从V的多分类变成0，1二分类，每次输出概率从V减小到2*K，从V个向量相乘减小到K个向量相乘。(ps 同时也可以提升词向量的效果)

This loss function changes from V multi-classifiers to {0,1} binary classifiers, and the probability need to output decreases from V to 2*K.

qustion f

> Suppose the center word is c = w t and the context window is [w t−m , ..., w t−1 , w t , w t+1 , ...,w t+m], where m is the context window size. Recall that for the skip-gram version of word2vec, the total loss for the context window is:

> Write down three partial derivatives:

> 假设中心词是 c w t，上下文窗口是[w t-m，… ， w t-1，w t，w t + 1，… ， w t + m]，其中 m 是上下文窗口大小。 回想一下，对于 word2vec 的跳过格拉姆版本，上下文窗口的总损失是:
>
> 写下三个偏导数:

i)

$$\frac{\partial J_{skip-gram}(\boldsymbol{v}_c, w_{t-m}, ..., w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{U}} = \sum_{-m<=j<=m, j!=0} \frac{\partial J(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})}{\partial \boldsymbol{U}}$$

ii)

when w=c,

$$\frac{\partial J_{skip-gram}(\boldsymbol{v}_c, w_{t-m}, ..., w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{v}_c} = \sum_{-m<=j<=m, j!=0} \frac{\partial J(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})}{\partial \boldsymbol{v}_c}$$

iii)

when w!=c,

$$\frac{\partial J_{skip-gram}(\boldsymbol{v}_c, w_{t-m}, ..., w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{v}_w} = \boldsymbol{0}$$