CS 224n Assignment #2: word2vec (43 Points)

1 Written: Understanding word2vec (23 points)

Let's have a quick refresher on the `word2vec` algorithm. The key insight behind `word2vec` is that *'a word is known by the company it keeps'*. Concretely, suppose we have a 'center' word c and a contextual window surrounding c. We shall refer to words that lie in this contextual window as 'outside words'. For example, in Figure 1 we see that the center word c is 'banking'. Since the context window size is 2, the outside words are 'turning', 'into', 'crises', and 'as'.

The goal of the skip-gram `word2vec` algorithm is to accurately learn the probability distribution $P(O|C)$. Given a specific word o and a specific word c, we want to calculate $P(O = o|C = c)$, which is the probability that word o is an 'outside' word for c, i.e., the probability that o falls within the contextual window of c.

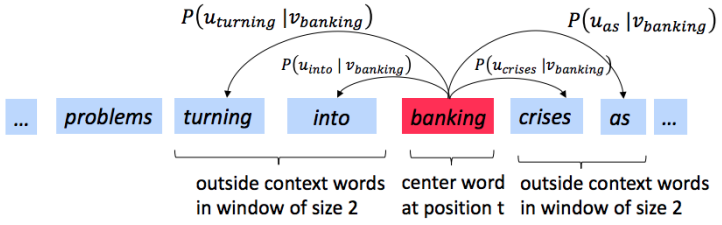


Figure 1: The word2vec skip-gram prediction model with window size 2

In `word2vec`, the conditional probability distribution is given by taking vector dot-products and applying the softmax function:

$$P(O = o \mid C = c) = \frac{\exp(\boldsymbol{u}_o^\top \boldsymbol{v}_c)}{\sum_{w \in \text{Vocab}} \exp(\boldsymbol{u}_w^\top \boldsymbol{v}_c)} \tag{1}$$

Here, $\boldsymbol{u}_o$ is the 'outside' vector representing outside word o, and $\boldsymbol{v}_c$ is the 'center' vector representing center word c. To contain these parameters, we have two matrices, $\boldsymbol{U}$ and $\boldsymbol{V}$. The columns of $\boldsymbol{U}$ are all the 'outside' vectors $\boldsymbol{u}_w$. The columns of $\boldsymbol{V}$ are all of the 'center' vectors $\boldsymbol{v}_w$. Both $\boldsymbol{U}$ and $\boldsymbol{V}$ contain a vector for every $w \in$ Vocabulary.[1]

Recall from lectures that, for a single pair of words c and o, the loss is given by:

$$\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log P(O = o|C = c). \tag{2}$$

Another way to view this loss is as the cross-entropy[2] between the true distribution $\boldsymbol{y}$ and the predicted distribution $\hat{\boldsymbol{y}}$. Here, both $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are vectors with length equal to the number of words in the vocabulary. Furthermore, the k^{th} entry in these vectors indicates the conditional probability of the k^{th} word being an 'outside word' for the given c. The true empirical distribution $\boldsymbol{y}$ is a one-hot vector with a 1 for the true outside word o, and 0 everywhere else. The predicted distribution $\hat{\boldsymbol{y}}$ is the probability distribution $P(O|C = c)$ given by our model in equation (1).

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

[1] Assume that every word in our vocabulary is matched to an integer number k. $\boldsymbol{u}_k$ is both the k^{th} column of $\boldsymbol{U}$ and the 'outside' word vector for the word indexed by k. $\boldsymbol{v}_k$ is both the k^{th} column of $\boldsymbol{V}$ and the 'center' word vector for the word indexed by k. **In order to simplify notation we shall interchangeably use k to refer to the word and the index-of-the-word.**

[2] The Cross Entropy Loss between the true (discrete) probability distribution p and another distribution q is $-\sum_i p_i \log(q_i)$.

$$-\sum_{w \in Vocab} y_w \log(\hat{y}_w) = -\log(\hat{y}_o). \tag{3}$$

Your answer should be one line.

(b) (5 points) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{v}_c$. Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{U}$.

(c) (5 points) Compute the partial derivatives of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to each of the 'outside' word vectors, $\boldsymbol{u}_w$'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 $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{v}_c$.

(d) (3 Points) The sigmoid function is given by Equation 4:

$$\sigma(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}}} = \frac{e^{\boldsymbol{x}}}{e^{\boldsymbol{x}} + 1} \tag{4}$$

Please compute the derivative of $\boldsymbol{\sigma}(x)$ with respect to $\boldsymbol{x}$, where $\boldsymbol{x}$ is a vector.

(e) (4 points) 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 $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$. 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:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \tag{5}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.[3]

Please repeat parts (b) and (c), computing the partial derivatives of $\boldsymbol{J}_{\text{neg-sample}}$ with respect to $\boldsymbol{v}_c$, with respect to $\boldsymbol{u}_o$, and with respect to a negative sample $\boldsymbol{u}_k$. Please write your answers in terms of the vectors $\boldsymbol{u}_o$, $\boldsymbol{v}_c$, and $\boldsymbol{u}_k$, where $k \in [1, K]$. 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.

(f) (3 points) Suppose the center word is $c = w_t$ and the context window is $[w_{t-m}, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_{t+m}]$, where m is the context window size. Recall that for the skip-gram version of `word2vec`, the total loss for the context window is:

$$\boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) \tag{6}$$

Here, $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ represents an arbitrary loss term for the center word $c = w_t$ and outside word w_{t+j}. $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ could be $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ or $\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$, depending on your implementation.

Write down three partial derivatives:

(i) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{U}$

(ii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{v}_c$

[3]Note: the loss function here is the negative of what Mikolov et al. had in their original paper, because we are doing a minimization instead of maximization in our assignment code. Ultimately, this is the same objective function.

(iii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U})/\partial \boldsymbol{v}_w$ when $w \neq c$

Write your answers in terms of $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})/\partial \boldsymbol{U}$ and $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})/\partial \boldsymbol{v}_c$. This is very simple – each solution should be one line.

Once you're done: *Given that you computed the derivatives of $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ with respect to all the model parameters $\boldsymbol{U}$ and $\boldsymbol{V}$ in parts (a) to (c), you have now computed the derivatives of the full loss function $\boldsymbol{J}_{\text{skip-gram}}$ with respect to all parameters. You're ready to implement* `word2vec`*!*

2 Coding: Implementing word2vec (20 points)

In this part you will implement the word2vec model and train your own word vectors with stochastic gradient descent (SGD). Before you begin, first run the following commands within the assignment directory in order to create the appropriate conda virtual environment. This guarantees that you have all the necessary packages to complete the assignment.

```
conda env create -f env.yml
conda activate a2
```

Once you are done with the assignment you can deactivate this environment by running:

```
conda deactivate
```

(a) (12 points) First, implement the `sigmoid` function in `word2vec.py` to apply the sigmoid function to an input vector. In the same file, fill in the implementation for the softmax and negative sampling loss and gradient functions. Then, fill in the implementation of the loss and gradient functions for the skip-gram model. When you are done, test your implementation by running `python word2vec.py`.

(b) (4 points) Complete the implementation for your SGD optimizer in `sgd.py`. Test your implementation by running `python sgd.py`.

(c) (4 points) Show time! Now we are going to load some real data and train word vectors with everything you just implemented! We are going to use the Stanford Sentiment Treebank (SST) dataset to train word vectors, and later apply them to a simple sentiment analysis task. You will need to fetch the datasets first. To do this, run `sh get_datasets.sh`. There is no additional code to write for this part; just run `python run.py`.

Note: The training process may take a long time depending on the efficiency of your implementation ***(an efficient implementation takes approximately an hour)****. Plan accordingly!*

After 40,000 iterations, the script will finish and a visualization for your word vectors will appear. It will also be saved as `word_vectors.png` in your project directory. **Include the plot in your homework write up.** Briefly explain in at most three sentences what you see in the plot.

3 Submission Instructions

You shall submit this assignment on GradeScope as two submissions – one for "Assignment 2 [coding]" and another for 'Assignment 2 [written]":

(a) Run the `collect_submission.sh` script to produce your `assignment2.zip` file.

(b) Upload your `assignment2.zip` file to GradeScope to "Assignment 2 [coding]".

(c) Upload your written solutions to GradeScope to "Assignment 2 [written]".

$$\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log P(O = o|C = c). \tag{2}$$

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

$$-\sum_{w \in Vocab} y_w \log(\hat{y}_w) = -\log(\hat{y}_o).$$

Your answer should be one line.

先来理清一下思路，对于skip-gram模型：

输入：随机化的U、V.　　输出：结果U、V

对某个单词C：$P(O=o|C=c) = \dfrac{\exp(u_o^T v_c)}{\sum_{w \in Vocab} \exp(u_w^T v_c)}$

对词表中每个单词的$P(O=o|C=c)$组成3个.

真正的y应为一个只有outside word为1，其余为0的向量.

那么对一对单词C和O：

$J_{naive-softmax}(v_c, O, U) = -\log P(O=o|C=c)$

$J_{cross-entropy}(v_c, O, U) = -\sum_{O \in Vocab} y \log P(O=o|C=c)$

只有当O是真实的outside word时 $y=1$，其余情况 $y=0$.

∴ $J_{cross-entropy}(v_c, O, U) = -\log P(O=o|C=c)$

$= J_{naive-softmax}(v_c, O, U)$

∵对每对单词C和O两者都相等，　∴两个loss function相等

$$\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log P(O = o|C = c). \tag{2}$$

(b) (5 points) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{v}_c$. Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{U}$.

对一对单词 o 和 c：

$\frac{\partial J_{naive\text{-}softmax}(v_c, o, U)}{\partial v_c}$ 在课上推导过，在这里直接给出结论：

$$\frac{\partial J_{naive\text{-}softmax}(v_c, o, U)}{\partial v_c} = -u_o + \sum_{x \in V} P(x|c) \cdot u_x$$

$$= -u_o + \sum_{x \in V} \hat{y}_x \cdot u_x = \underset{d \times n}{U^T} \underset{n \times 1}{(\hat{y} - y)}$$

(c) (5 points) Compute the partial derivatives of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to each of the 'outside' word vectors, $\boldsymbol{u}_w$'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 $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{v}_c$.

当 w=o 时：

$$\frac{\partial J_{naive\text{-}softmax}(v_c, o, U)}{\partial u_o} = \frac{\partial -\log P(O=o|C=c)}{\partial u_o}$$

$$= \frac{\partial -\log \frac{\exp(u_o^T v_c)}{\sum_{w \in Vocab} \exp(u_w^T v_c)}}{\partial u_o}$$

$$= \frac{-\partial \log \exp(u_o^T v_c) + \partial \boxed{\log \sum_{w \in Vocab} \exp(u_w^T v_c)}}{\partial u_o}$$

这个数值只有一项与 u_o 有关，其余看作常数，记为 A

$$= -\frac{\partial u_o^T v_c}{u_o} + \frac{\partial \log(A + \exp(u_o^T v_c))}{\partial u_o}$$

$$= -v_c^T + \frac{1}{A + \exp(u_o^T v_c)} \cdot \frac{\partial (A + \exp(u_o^T v_c))}{\partial u_o}$$

$$= -v_c^T + \frac{1}{A + \exp(u_o^T v_c)} \cdot \frac{\partial \exp(u_o^T v_c)}{\partial u_o}$$

$$= -v_c^T + \frac{1}{A + \exp(u_o^T v_c)} \cdot \exp(u_o^T v_c) \cdot v_c^T$$

$$= -v_c^T + \frac{\exp(u_o^T v_c)}{\sum_{w \in Vocab} \exp(u_w^T v_c)} \cdot v_c^T$$

$$= -v_c^T + \hat{y}_o \cdot v_c^T = (\hat{y}_o - 1) v_c^T$$

当 $w \neq o$ 时：

$$\frac{\partial J_{naive\text{-}softmax}(v_c, o, U)}{\partial u_x} = \frac{\partial -\log p(O=o \mid C=c)}{\partial u_x}$$

常数

$$= \frac{\boxed{-\partial \log \exp(u_o^T v_c)} + \boxed{\partial \log \sum_{w \in Vocab} \exp(u_w^T v_c)}}{\partial u_x}$$

只有一项不是常数，其他常数记为 A.

$$= \frac{\partial \log(A + \exp(u_x^T v_c))}{\partial u_x} = \frac{1}{A + \exp(u_x^T v_c)} \cdot \frac{\partial (A + \exp(u_x^T v_c))}{\partial u_x}$$

$$= \frac{1}{\sum_{w \in Vocab} \exp(u_w^T v_c)} \cdot \frac{\partial \exp(u_x^T v_c)}{\partial u_x}$$

$$= \frac{1}{\sum_{w \in Vocab} \exp(u_w^T v_c)} \cdot \exp(u_x^T v_c) \cdot v_c^T$$

$$= \hat{y}_x \cdot v_c^T = \hat{y}_w \cdot v_c^T$$

综上：

$$\frac{\partial J_{naive\text{-}softmax}(v_c, o, U)}{\partial U}$$

$$= (\hat{y} - y)^T v_c$$

$1 \times n \quad 1 \times d$

求导：

(1) $\frac{\partial u^T v}{\partial u} = v^T$

(2) $\frac{\partial u^T v}{\partial v} = u$

(d) (3 Points) The sigmoid function is given by Equation 4:

$$\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{x}}} = \frac{e^{\boldsymbol{x}}}{e^{\boldsymbol{x}}+1} \tag{4}$$

Please compute the derivative of $\boldsymbol{\sigma}(x)$ with respect to $\boldsymbol{x}$, where $\boldsymbol{x}$ is a vector.

我们讨论向量x中的一个元素x_i：

$$\frac{\partial \sigma(x_i)}{\partial x_i} = \frac{\partial \frac{e^{x_i}}{e^{x_i}+1}}{\partial x_i} = \frac{\partial \frac{e^{x_i}}{e^{x_i}+1}}{\partial e^{x_i}} \cdot \frac{de^{x_i}}{\partial x_i} = \frac{\partial (1-\frac{1}{e^{x_i}+1})}{\partial e^{x_i}} \cdot e^{x_i}$$

令 $e^{x_i}=u$ $\quad \frac{\partial (1-\frac{1}{u+1})}{\partial u} \cdot u = \frac{1}{(u+1)^2} \cdot u = \frac{1}{(1+e^{x_i})^2} \cdot e^{x_i}$

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

(e) (4 points) 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 $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$. 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:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \tag{5}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.[3]

Please repeat parts (b) and (c), computing the partial derivatives of $\boldsymbol{J}_{\text{neg-sample}}$ with respect to $\boldsymbol{v}_c$, with respect to $\boldsymbol{u}_o$, and with respect to a negative sample $\boldsymbol{u}_k$. Please write your answers in terms of the vectors $\boldsymbol{u}_o$, $\boldsymbol{v}_c$, and $\boldsymbol{u}_k$, where $k \in [1, K]$. 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.

(1) $$\frac{\partial J_{\text{neg-sample}}(v_c, o, U)}{\partial v_c} = \frac{\partial -\log(\sigma(u_o^T v_c))}{\partial v_c} - \frac{\partial \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))}{\partial v_c}$$

$$= \frac{-1}{\sigma(u_o^T v_c)} \cdot \frac{\partial \sigma(u_o^T v_c)}{\partial v_c} - \sum_{k=1}^{K} \frac{\partial \log(\sigma(-u_k^T v_c))}{\partial v_c}$$

$$= -\frac{1}{\sigma(u_o^T v_c)} \cdot \sigma(u_o^T v_c) \cdot [1-\sigma(u_o^T v_c)] \cdot \frac{\partial u_o^T v_c}{\partial v_c}$$

$$- \sum_{k=1}^{K} \frac{1}{\sigma(-u_k^T v_c)} \cdot \frac{\partial \sigma(-u_k^T v_c)}{\partial v_c}$$

$$= [\sigma(u_o^T v_c) - 1] \cdot u_o - \sum_{k=1}^{K} \frac{1}{\sigma(-u_k^T v_c)} \cdot \sigma(-u_k^T v_c) \cdot [1 - \sigma(-u_k^T v_c)] \cdot \frac{\partial(-u_k^T v_c)}{\partial v_c}$$

$$= [\sigma(u_o^T v_c) - 1] \cdot u_o - \sum_{k=1}^{K} [1 - \sigma(-u_k^T v_c)] \cdot (-u_k)$$

$$= [\sigma(u_o^T v_c) - 1] \cdot u_o + \sum_{k=1}^{K} [1 - \sigma(-u_k^T v_c)] \cdot u_k$$

(2) $$\frac{\partial J_{neg\text{-}sample}(v_c, o, U)}{\partial u_o} = \frac{\partial -\log(\sigma(u_o^T v_c))}{\partial u_o} - \frac{\partial \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))}{\partial u_o}$$

常数

$$= \frac{\partial -\log(\sigma(u_o^T v_c))}{\partial u_o} = \frac{-1}{\sigma(u_o^T v_c)} \cdot \sigma(u_o^T v_c) \cdot [1 - \sigma(u_o^T v_c)] \cdot v_c$$

$$= [\sigma(u_o^T v_c) - 1] \cdot v_c$$

(3) $$\frac{\partial J_{neg\text{-}sample}(v_c, o, U)}{\partial u_k} = \frac{\partial -\log(\sigma(u_o^T v_c))}{\partial u_k} - \frac{\partial \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))}{\partial u_k}$$

常数　　除了u_k其余项常数

$$= - \frac{\partial \log(\sigma(-u_k^T v_c))}{\partial u_k} = \frac{1}{\sigma(-u_k^T v_c)} \cdot \sigma(-u_k^T v_c) \cdot$$

$$[1 - \sigma(-u_k^T v_c)] \cdot v_c^T$$

$$= [1 - \sigma(-u_k^T v_c)] \cdot v_c^T$$

注意到这3问的结果中存在大量可以复用的项，这使得在计算梯度时，更加高效.

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。(同时也可以提升词向量的效果).

(f) (3 points) Suppose the center word is $c = w_t$ and the context window is $[w_{t-m}, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_{t+m}]$, where m is the context window size. Recall that for the skip-gram version of `word2vec`, the total loss for the context window is:

$$\boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) \tag{6}$$

Here, $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ represents an arbitrary loss term for the center word $c = w_t$ and outside word w_{t+j}. $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ could be $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ or $\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$, depending on your implementation.

Write down three partial derivatives:

(i) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{U}$

(ii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{v}_c$

(iii) $\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) / \partial \boldsymbol{v}_w$ when $w \neq c$

Write your answers in terms of $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) / \partial \boldsymbol{U}$ and $\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) / \partial \boldsymbol{v}_c$. This is very simple – each solution should be one line.

Once you're done: *Given that you computed the derivatives of $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ with respect to all the model parameters $\boldsymbol{U}$ and $\boldsymbol{V}$ in parts (a) to (c), you have now computed the derivatives of the full loss function $\boldsymbol{J}_{\text{skip-gram}}$ with respect to all parameters. You're ready to implement* `word2vec`*!*

(i) $\sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial J(v_c, w_{t+j}, U)}{\partial U}$

(ii) $\sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial J(v_c, w_{t+j}, U)}{\partial v_c}$

(iii) $\sum_{\substack{-m \leq j \leq m \\ j \neq 0 \\ w \neq c}} \frac{\partial J(v_c, w_{t+j}, U)}{\partial v_w} = 0$

2. Coding

按照注释U中的每一行为一个outside vector，假设一共有n个单词.

$U = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$ 其中u_i为d维向量（1×d） U的shape：(n×d)

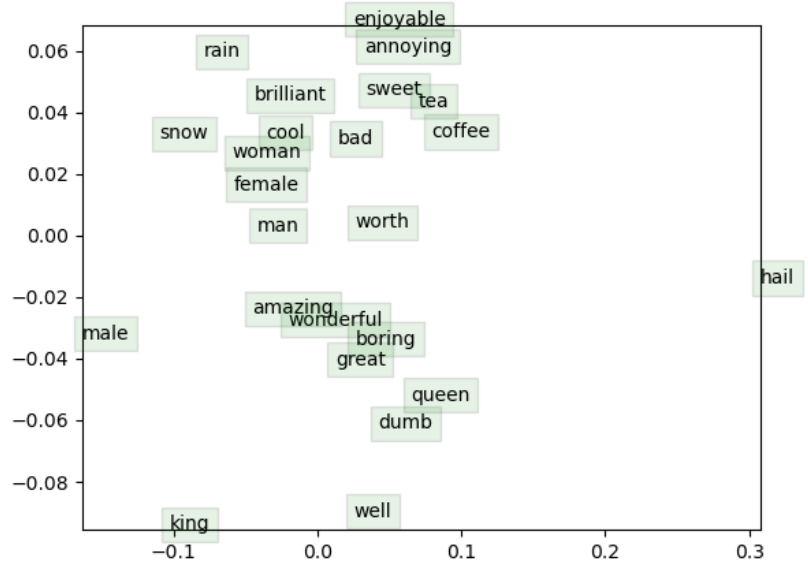
$\frac{\partial J}{\partial U} = \frac{\partial J}{\partial \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}} = \left(\frac{\partial J}{\partial u_1}, \frac{\partial J}{\partial u_2} \cdots \frac{\partial J}{\partial u_n}\right)$ dxn维矩阵

$\frac{\partial J}{\partial u_i}$ 为d×1维

在前面的writing部分，我们已经求出了$\frac{\partial J}{\partial u_i}$，现在只要代入就可以完成。

总结：在矩阵求导和书写代码的过程中，可以领悟到在深度学习中，得到的结果是n×K，还是K×n真的影响不大，主要是看问题/输出想要什么维度，只要保证乘法时维度能match上就可以了.

coding部分的结果：



从图上来看，"woman"和"female"很接近，这点是符合认知的（除此还有，"tea"，"coffee"；"amazing"，"wonderful"等等.）

male - King，female - queen有对应关系.