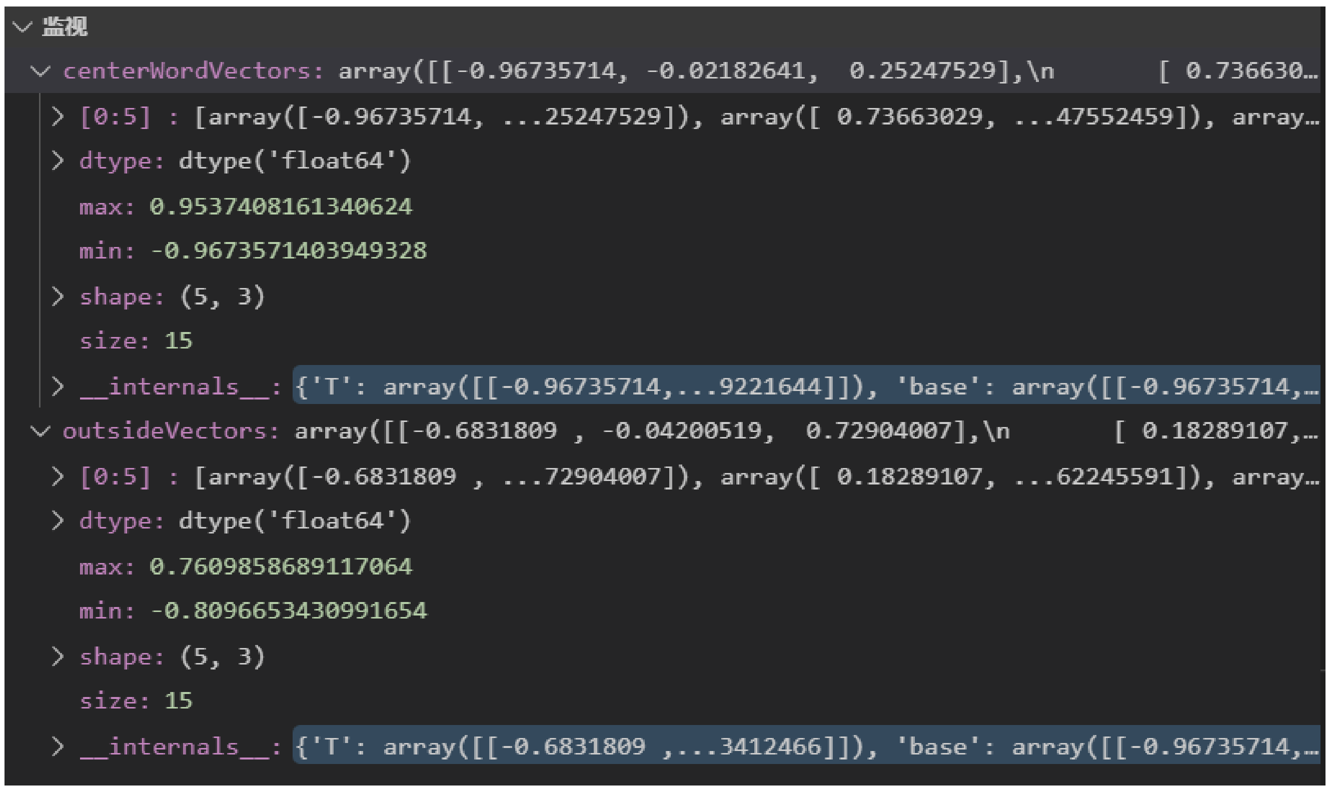
Assignment2 [written]

相关参数，详见 notes

- n 向量维度，$|V|$ 词汇表大小
- w_i 词汇表中的单词i
- $V \in R^{|V|*n}$ 输入词矩阵
- $v_i \in R^{1*n}$ V的第 i行，单词 w_i的输入向量表示
- $U \in R^{|V|*n}$ 输出词矩阵
- $u_i \in R^{1*n}$ U的第 i行，单词 w_i的输出向量表示
- $y \in R^{1*|V|}$ one-hot 行向量，the true probabilities
- $\hat{y} \in R^{1*|V|}$ 行向量，the predicted probabilities

p.s. 注意，此处的输入矩阵 V（即作业程序中的 centerWordVectors）与 notes中的矩阵维度描述相反。另外，做这个作业我的小心得是，搞清楚**相关参数维度**以及整个**运行过程**，对分析偏导和理解程序，很重要。可以 debug查看V、U矩阵相关参数如下



(a)

交叉熵损失 $H(\hat{y}, y) = -\sum_{w\in Vocab} y_w log(\hat{y_w})$

其中 y 是one-hot向量($j = o$处为1，其余为0)，$H(\hat{y}, y)$ 可简化，即有

$$H(\hat{y}, y) = -\sum_{w\in Vocab} y_w log(\hat{y_w}) = -\sum_{j=1}^{|V|} y_j log(\hat{y_j}) = -y_o log(\hat{y_o}) = -1 * log(\hat{y_o}) = -log(\hat{y_o})$$

(b)

$$\begin{aligned} J_{naive-softmax}(v_c, o, U) &= -logP(O = o|C = c) = -log(\frac{exp(u_o^T v_c)}{\sum_{w\in Vocab} exp(u_w^T v_c)}) \\ &= -u_o^T v_c + log(\sum_{w\in Vocab} exp(u_w^T v_c)) \end{aligned}$$

逐级求导有

$$\begin{aligned} \frac{\partial J_{naive-softmax}(v_c, o, U)}{\partial v_c} &= -u_o + \frac{1}{\sum_{w\in Vocab} exp(u_w^T v_c)} * \sum_{w\in Vocab} exp(u_w^T v_c) u_w \\ &= -u_o + \sum_{w\in Vocab} \frac{exp(u_w^T v_c)}{\sum_{w\in Vocab} exp(u_w^T v_c)} u_w \\ &= -u_o + \sum_{w\in Vocab} P(O = w|C = c) u_w \\ &= -u_o + \sum_{w\in Vocab} \hat{y_w} u_w \end{aligned}$$

受（a）中启发，考虑将 $\sum$ 项中的 $\hat{y_w}$拆分为 $y_w + (\hat{y_w} - y_w)$，有 $\sum_{w\in Vocab} y_w u_w = u_o$，上式为

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c,o,U)}{\partial v_c} &= -u_o + \sum_{w\in Vocab}\hat{y_w}u_w \\ &= -u_o + \sum_{w\in Vocab}[y_w + (\hat{y_w}-y_w)]u_w \\ &= -u_o + \sum_{w\in Vocab}y_w u_w + \sum_{w\in Vocab}(\hat{y_w}-y_w)u_w \\ &= -u_o + u_o + \sum_{w\in Vocab}(\hat{y_w}-y_w)u_w \\ &= \sum_{w\in Vocab}(\hat{y_w}-y_w)u_w\end{aligned}$$

展开化简有

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c,o,U)}{\partial v_c} &= \sum_{j=1}^{|V|}(\hat{y_w}-y_w)u_w = (\hat{y_1}-y_1)u_1 + (\hat{y_2}-y_2)u_2 + \ldots + (\hat{y_{|V|}}-y_{|V|})u_{|V|} \\ &= \begin{pmatrix}\hat{y_1}-y_1 & \hat{y_2}-y_2 & \cdots & \hat{y_{|V|}}-y_{|V|}\end{pmatrix}\begin{pmatrix}u_1\\u_2\\\vdots\\u_{|V|}\end{pmatrix} = (\hat{y}-y)U\end{aligned}$$

(c)

由（b）中

$$J_{naive-softmax}(v_c,o,U) = -u_o^T v_c + log(\sum_{w\in Vocab}exp(u_w^T v_c))$$

当 $w \neq o$ 时 $y_w = 0$

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c,o,U)}{\partial u_w} &= 0 + \frac{1}{\sum_{w\in Vocab}exp(u_w^T v_c)} * \sum_{w\in Vocab}exp(u_w^T v_c)v_c \\ &= \sum_{w\in Vocab}\frac{exp(u_w^T v_c)}{\sum_{w\in Vocab}exp(u_w^T v_c)}v_c \\ &= \sum_{w\in Vocab}P(O=w|C=c)v_c \\ &= \sum_{w\in Vocab}\hat{y_w}v_c \\ &= \hat{y_1}v_c + \hat{y_2}v_c + \hat{y_{|V|}}v_c \\ &= \hat{y}v_c \Rightarrow (\hat{y_w}-y_w)v_c\end{aligned}$$

当 $w = o$ 时 $y_w = 1$

$$\begin{aligned}\frac{\partial J_{naive-softmax}(v_c,o,U)}{\partial u_w} &= -v_c + \sum_{w\in Vocab}P(O=w|C=c)v_c \\ &= -v_c + \sum_{w\in Vocab}\hat{y_w}v_c \\ &= (\hat{y_o}-1)v_c \Rightarrow (\hat{y_w}-y_w)v_c\end{aligned}$$

另外，考虑到维度问题， $y,\hat{y} \in R^{1*|V|}, v_c \in R^{1*n}, \frac{\partial J_{naive-softmax}(v_c,o,U)}{\partial u_w} \in R^{|V|*n}$，综上有$\frac{\partial J_{naive-softmax}(v_c,o,U)}{\partial u_w} = (\hat{y}-y)^T v_c$

(d)

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

(e)

$$\begin{aligned}\frac{\partial J_{neg-sample}(v_c,o,U)}{\partial v_c} &= -\frac{(1-\sigma(u_o^T v_c))*\sigma(u_o^T v_c)*u_o}{\sigma(u_o^T v_c)} - \sum_{k=1}^{K}\frac{(1-\sigma(-u_k^T v_c))*\sigma(-u_k^T v_c)*(-u_k)}{\sigma(-u_k^T v_c)} \\ &= -(1-\sigma(u_o^T v_c))*u_o - \sum_{k=1}^{K}(1-\sigma(-u_k^T v_c))*(-u_k) \\ &= -(1-\sigma(u_o^T v_c))*u_o + \sum_{k=1}^{K}(1-\sigma(-u_k^T v_c))*u_k\end{aligned}$$

$$\begin{aligned}\frac{\partial J_{neg-sample}(v_c,o,U)}{\partial u_k} &= 0 - \sum_{k=1}^{K}\frac{1}{\sigma(-u_k^T v_c)} * (1-\sigma(-u_k^T v_c))*\sigma(-u_k^T v_c)*(-v_c) \\ &= -\sum_{k=1}^{K}(1-\sigma(-u_k^T v_c))*(-v_c) \\ &= \sum_{k=1}^{K}(1-\sigma(-u_k^T v_c))*v_c\end{aligned}$$

$$\begin{aligned}\frac{\partial J_{neg-sample}(v_c,o,U)}{\partial u_o} &= -\frac{(1-\sigma(u_o^T v_c))*\sigma(u_o^T v_c)*v_c}{\sigma(u_o^T v_c)} \\ &= -(1-\sigma(u_o^T v_c))*v_c\end{aligned}$$

(f)

$$(i)\ \frac{\partial J_{skip-gram}(v_c, w_{t-m}, \dots w_{t+m}, U)}{\partial U} = \sum_{-m\le j\le m, j\ne 0} \frac{\partial J(v_c, w_{t+j}, U)}{\partial U}$$

$$(ii)\ \frac{\partial J_{skip-gram}(v_c, w_{t-m}, \dots w_{t+m}, U)}{\partial v_c} = \sum_{-m\le j\le m, j\ne 0} \frac{\partial J(v_c, w_{t+j}, U)}{\partial v_c}$$

$$(iii)\ \frac{\partial J_{skip-gram}(v_c, w_{t-m}, \dots w_{t+m}, U)}{\partial v_w} = 0 (w \ne c)$$

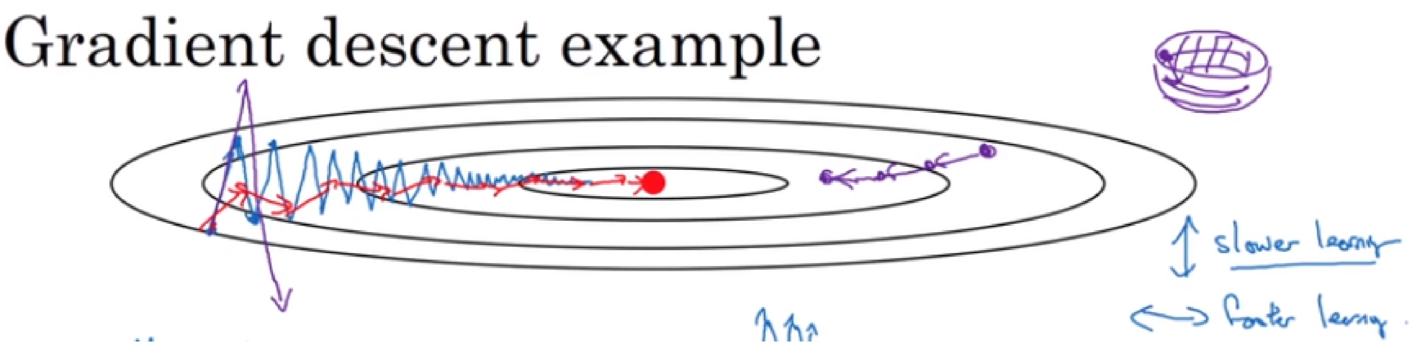
> p.s. 这块我花了不少时间，学到这儿再回去看 Chris Manning 第一个video最后的推导，思路会清晰很多，部分式子理解可参考此网站

Assignment3 [written]

1.

(a)

i. 当横向与纵向的梯度变化差别很大时，SGD 会朝最小值振荡前进，梯度下降的速度很慢且我们不能使用大的学习率。Adam 计算梯度的指数加权平均，然后用这个梯度来更新权重。将 m看作速度，β_1看作摩擦，后边的倒数项看成加速度，指数加权平均大的方向得到更大的动量，可以更快的朝着最小值方向移动。详见吴恩达视频



ii. 学习率

(b)

i. $E_{P_{drop}}[h_{drop}]_i = E_{P_{drop}}[\gamma dh]_i = \gamma E_{P_{drop}}[dh]_i = \gamma(1-P_{drop})h_i = h_i$

则有 $\gamma = \frac{1}{1-P_{drop}}$

ii. 训练时随机失活可以减少数据过拟合，而测试评估时使用 dropout会随机失活部分cell，从而可能产生不同的结果，我们不希望得到随机的预测结果，所以在评估时不用dropout

2.

(a)

Stack	Buffer	New dependency	Transition
[ROOT]	[I,parsed,this,sentence,correctly]		Initial Configuration
[ROOT,I]	[parsed,this,sentence,correctly]		SHIFT
[ROOT,I,parsed]	[this,sentence,correctly]		SHIFT
[ROOT,parsed]	[this,sentence,correctly]	parsed $\rightarrow$I	LEFT-ARC
[ROOT,parsed,this]	[sentence,correctly]		SHIFT
[ROOT,parsed,this,sentence]	[correctly]		SHIFT
[ROOT,parsed,sentence]	[correctly]	sentence $\rightarrow$this	LEFT-ARC
[ROOT,parsed]	[correctly]	parsed$\rightarrow$sentence	RIGHT-ARC
[ROOT,parsed,correctly]	[]		SHIFT
[ROOT,parsed]	[]	parsed$\rightarrow$correctly	RIGHT-ARC
[ROOT]	[]	$ROOT$ $\rightarrow$parsed	RIGHT-ARC

(b)

2n
将所有的 n个词移进$\rightarrow$ n steps + 每个词被且仅被指向一次$\rightarrow$ n steps

(c)

```
(cs) PS C:\Users\Lucky\Downloads\a3\a3> python parser_transitions.py part_c
SHIFT test passed!
LEFT-ARC test passed!
RIGHT-ARC test passed!
parse test passed!
(cs) PS C:\Users\Lucky\Downloads\a3\a3> 
```

(d)

```
(cs) PS C:\Users\Lucky\Downloads\a3\a3> python parser_transitions.py part_d
minibatch_parse test passed!
(cs) PS C:\Users\Lucky\Downloads\a3\a3> 
```

(e)

```
100%|
| 48/48 [00:08<00:00,  5.56it/s]
Average Train Loss: 0.14847215808307132
Evaluating on dev set
125250it [00:00, 7848927.64it/s]
- dev UAS: 73.26
New best dev UAS! Saving model.

(cs) PS C:\Users\Lucky\Downloads\a3\a3> 
```

> p.s. 因为有随机 dropout，所以每次运行得到的数据不一定相同

```
Epoch 10 out of 10
100%|
Average Train Loss: 0.06733619525536766
Evaluating on dev set
1445850it [00:00, 29196863.04it/s]
- dev UAS: 88.72
New best dev UAS! Saving model.
```

```
TESTING
================================================================================
Restoring the best model weights found on the dev set
Final evaluation on test set
2919736it [00:00, 34301809.12it/s]

- test UAS: 89.02
Done!
```

(f)

i. Verb Phrase Attachment Error; wedding$\rightarrow$fearing; heading$\rightarrow$fearing
ii. Coordination Attachment Error; makes$\rightarrow$rescue;rush$\rightarrow$rescue
iii. Prepositional Phrase Attachment Error; named$\rightarrow$Midland; guy$\rightarrow$Midland
iv. Modifier Attachment Error; elements$\rightarrow$most; crucial$\rightarrow$most