Assignment 4

Written by Shihan Ran - 15307130424.

1 word2vec

Notice! My vector are all stored by columns.

(a) Derive the gradients with respect to v_c .

The objective function is

$$J'(\theta)= \prod_{t=1}^{T} \prod_{-m\le j\le m} p(w_{t+j} \mid w_t)$$

$$J(\theta)= -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m\le j\le m} \log p(w_{t+j} \mid w_t)$$

where θ represents all variables we optimize and

$$p(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

We need to either maximize the log probability $J'(\theta)$ or minimize the negative log likelihood $J(\theta)$.

Now we take derivatives to work out minimum $J(\theta)$.

$$\frac{\partial}{\partial v_c} \log(p(o \mid c))= \frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)} = \frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) - \frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_w^T v_c)$$

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

$$\begin{aligned}
\frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_w^T v_c)&= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \frac{\partial}{\partial v_c} \sum_{w=1}^{V} \exp(u_w^T v_c) \\
&= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \sum_{w=1}^{V} \frac{\partial}{\partial v_c} \exp(u_w^T v_c) \\
&= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \sum_{w=1}^{V} \exp(u_w^T v_c) \frac{\partial}{\partial v_c} u_w^T v_c \\
&= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \sum_{w=1}^{V} \exp(u_w^T v_c) u_w^T
\end{aligned}$$

Hence, we got

$$\begin{aligned}\frac{\partial}{\partial v_c}\log(p(o\mid c)) &= u_o - \frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)}\sum_{w=1}^{V}\exp(u_w^T v_c)u_w^T \\ &= u_o - \sum_{x=1}^{V}\frac{\exp(u_x^T v_c)}{\sum_{w=1}^{V}\exp(u_w^T v_c)}u_x \\ &= u_o - \sum_{x=1}^{V}p(x\mid c)u_x\end{aligned}$$

And for $J_{softmax-CE}$, we got

$$\begin{aligned}\frac{\partial}{\partial v_c}J_{softmax-CE} &= -\frac{\partial}{\partial v_c}\log(p(o\mid c)) \\ &= -u_o + \sum_{x=1}^{V}p(x\mid c)u_x\end{aligned}$$

(b) Derive the gradients with respect to u_w .

for u_w , we take derivatives again:

$$\frac{\partial}{\partial u_w}\log\frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V}\exp(u_w^T v_c)} = \frac{\partial}{\partial u_w}\log\exp(u_o^T v_c) - \frac{\partial}{\partial u_w}\log\sum_{w=1}^{V}\exp(u_w^T v_c)$$

$$\frac{\partial}{\partial u_w}\log\exp(u_o^T v_c) = \frac{\partial}{\partial u_w}u_o^T v_c = 0$$

$$\begin{aligned}\frac{\partial}{\partial u_w}\log\sum_{w=1}^{V}\exp(u_w^T v_c) &= \frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)}\frac{\partial}{\partial u_w}\sum_{w=1}^{V}\exp(u_w^T v_c) \\ &= \frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)}\frac{\partial}{\partial u_w}\exp(u_w^T v_c) \\ &= \frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)}\exp(u_w^T v_c)\frac{\partial}{\partial u_w}u_w^T v_c \\ &= \frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)}\exp(u_w^T v_c)v_c\end{aligned}$$

Hence, we got

$$\begin{aligned}\frac{\partial}{\partial u_w}\log(p(o\mid c)) &= 0 - \frac{1}{\sum_{w=1}^{V}\exp(u_w^T v_c)}\exp(u_w^T v_c)v_c \\ &= -\frac{\exp(u_w^T v_c)v_c}{\sum_{i=1}^{V}\exp(u_i^T v_c)}\end{aligned}$$

And for $J_{softmax-CE}$, we got

$$\begin{aligned}\frac{\partial}{\partial u_w}J_{softmax-CE} &= -\frac{\partial}{\partial u_w}\log(p(o\mid c)) \\ &= \frac{\exp(u_w^T v_c)v_c}{\sum_{i=1}^{V}\exp(u_i^T v_c)}\ (w\neq o)\end{aligned}$$

Especially, for u_o , we have

$$\frac{\partial}{\partial u_o} J_{softmax-CE} = -\frac{\partial}{\partial u_o} \log(p(o \mid c)) = \frac{\exp(u_o^T v_c) v_c}{\sum_{i=1}^{V} \exp(u_i^T v_c)} - v_c$$

(c)

(c.1) Repeat part (a) and (b) assuming we are using the negative sampling loss.

The objective function now becomes

$$J(\theta) = -\log(\sigma(u_0^T v_c)) - \sum_{K=1}^{K} \log(\sigma(-u_k^T v_c))$$

where $\sigma(\cdot)$ is the sigmoid function.

Now we take derivatives with respect to v_c to work out the optimum $J(\theta)$.

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

$$\begin{aligned}
\frac{\partial}{\partial v_c} \log(\sigma(u_o^T v_c)) &= \frac{1}{\sigma(u_o^T v_c)} \frac{\partial}{\partial v_c} \sigma(u_o^T v_c) \\
&= \frac{1}{\sigma(u_o^T v_c)} \sigma(u_o^T v_c)(1 - \sigma(u_o^T v_c)) \frac{\partial}{\partial v_c} u_o^T v_c \\
&= (1 - \sigma(u_o^T v_c)) \frac{\partial}{\partial v_c} u_o^T v_c \\
&= (1 - \sigma(u_o^T v_c)) u_o \\
\frac{\partial}{\partial v_c} \sum_{K=1}^{K} \log(\sigma(-u_k^T v_c)) &= \sum_{K=1}^{K} \frac{\partial}{\partial v_c} \log(\sigma(-u_k^T v_c)) \\
&= -\sum_{K=1}^{K} (1 - \sigma(-u_k^T v_c)) u_k
\end{aligned}$$

Hence, we got

$$\frac{\partial}{\partial v_c} J(\theta) = (\sigma(u_o^T v_c) - 1) u_o + \sum_{K=1}^{K} (1 - \sigma(-u_k^T v_c)) u_k$$

The derivatives with respect to u_k is as followings:

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

$$\frac{\partial}{\partial u_k}\log(\sigma(u_o^T v_c)) = 0$$

$$\begin{aligned}\frac{\partial}{\partial u_k}\sum_{K=1}^{K}\log(\sigma(-u_k^T v_c)) &= \frac{1}{\sigma(-u_k^T v_c)}\frac{\partial}{\partial u_k}\sigma(-u_k^T v_c)\\ &= \frac{1}{\sigma(-u_k^T v_c)}\sigma(-u_k^T v_c)(1-\sigma(-u_k^T v_c))\frac{\partial}{\partial u_k}(-u_k^T v_c)\\ &= (1-\sigma(u_o^T v_c))\frac{\partial}{\partial u_k}(-u_k^T v_c)\\ &= (\sigma(u_o^T v_c)-1)\,v_c\end{aligned}$$

Hence, we got

$$\frac{\partial}{\partial u_k}J(\theta) = (\sigma(u_o^T v_c)-1)\,v_c$$

Especially, for u_o, we have

$$\begin{aligned}\frac{\partial}{\partial u_o}J(\theta) &= -\frac{\partial}{\partial u_o}\log(p(o \mid c))\\ &= (\sigma(u_o^T v_c)-1)\,v_c + (\sigma(u_o^T v_c)-1)\,v_c\\ &= [(\sigma(u_o^T v_c)-1)+1]\,v_c\end{aligned}$$

(c.2) Describe why this cost function is much more efficient

Training a neural network means taking a training example and adjusting **all of the neuron weights** slightly so that it predicts that training sample more accurately. In other words, each training sample will tweak all of the weights in the neural network.

In the softmax-CE loss, the size of our word vocabulary means that our skip-gram neural network has a tremendous number of weights, all of which would be updated slightly by every one of our billions of training samples.

Negative sampling addresses this by having **each training sample only modify a small percentage of the weights**, rather than all of them.

(d) Derive gradients for all of the word vectors for skip-gram.

Recall that $J_{skip-gram} = \sum_{-m\le j\le m} F(w_{c+j}, v_c)$,
We can derive gradients as followings:

$$\frac{\partial}{\partial v_c} J(\theta) = \frac{\partial}{\partial v_c} \sum_{-m \le j \le m} F(w_{c+j}, v_c)$$
$$= \sum_{-m \le j \le m} \frac{\partial}{\partial v_c} F(w_{c+j}, v_c)$$
$$\frac{\partial}{\partial u_{w_{c+j}}} J(\theta) = \frac{\partial}{\partial u_{w_{c+j}}} \sum_{-m \le j \le m} F(w_{c+j}, v_c)$$
$$= \sum_{-m \le j \le m} \frac{\partial}{\partial u_{w_{c+j}}} F(w_{c+j}, v_c)$$

we can get the expression of $\frac{\partial}{\partial v_c} F(w_{c+j}, v_c)$ and $\frac{\partial}{\partial u_{w_{c+j}}} F(w_{c+j}, v_c)$ in (a)~(c).

(e) Implement word2vec models and train word vectors with SGD.

First, write a helper function to normalize rows of a matrix in word2vec.py.

```
1 return sklearn.preprocessing.normalize(x, norm='l2')
```