Assignment 2

Written : Understanding Word2vec

a.

$- \sum_{w \in Vocab} y_w.log(y_w^{hat}) = -log(y_0^{hat})$ because y_w is indeed one-hot coding vector.

b.

i.

Compute the partial derivative of J with respect to v_c

$$J_{naive-softmax}(v_c, o, U) = -\log P(O = o|C = c) = -u_0^T v_c + log(\sum_{w \in Vocab} e^{u_w^T.v_c})$$

The derivative is : $-u_0 + \sum u_w.P(O = w|C = c)$ = $U^T.(\hat{y} - y)$ $w \in$ Vocab

ii.

The derivative is zero when $\hat{y}$ is close to y

c.

- $u_w = u_o$, the derivative with respect to u_w is : $-v_c + P(O = o).v_c = (P(O = o) - 1).v_c = [(\hat{y} - y).y].v_c$
- $u_w = u_k$ # u_o, the derivative with respect to u_w is : $P(O = w).v_c = \hat{y}_k.v_c$

$\hat{y}_k$ is the probability distribution value at the $k^{th}$of $\hat{y}$ for the possibility the context word is u_k

d.

The derivative of J with respect to U is :

$[\hat{y_1}.v_c, \;\; \hat{y_2}.v_c, .(y \hat{-} y).y.v_c.. , y_{\hat{vocab}}.v_c]$

g. Negative sampling loss

$Jneg - sample(vc, o, U) = -log(\sigma(u_o^\top.v_c)) - \sum_{s=1}^{K} log(\sigma(-u_{w_s}^\top.v_c))$

The derivative with respect to v_c is : $-u_o.(1 - \delta(u_o^T.v_c)) + \sum u_{w_s}.(1 - \delta(-u_{w_s}^T.v_c))$

δ is the sigmoid function.

The derivative with respect to u_o is : $-v_c.(1 - \delta(u_0^T.v_c))$

The derivative with respect to u_{w_s} is : $v_c.(1 - \delta(-u_{w_s}^T.v_c))$

h.

i. $\frac{dJ(v_c, w_{t-m}, ...w_{t+m}, U)}{dU} = \sum \frac{dJ(v_c, w_{t+j}, U)}{dU}$

ii. $\frac{dJ(v_c, w_{t-m}, ...w_{t+m}, U)}{dv_c} = \sum \frac{dJ(v_c, w_{t+j}, U)}{dv_c}$

iii. $\frac{dJ(v_c, w_{t-m}, ...w_{t+m}, U)}{dv_w} = \sum \frac{dJ(v_c, w_{t+j}, U)}{dv_w} = 0$ **with all** $v_w \# v_c$

Coding results

- Run with method (loss function) : NaiveSoftmax

```
iter 39970: 9.776979
iter 39980: 9.813174
iter 39990: 9.854022
iter 40000: 9.812206
→ sanity check: cost at convergence should be around or below 10
→ training took 94730 seconds (40000 iterations)
```

- Run with method (loss function) : NegSampling

  ```
  iter 39950: 9.730395
  iter 39960: 9.721694
  iter 39970: 9.668252
  iter 39980: 9.610189
  iter 39990: 9.573013
  iter 40000: 9.626349
  ```

  → sanity check: cost at convergence should be around or below 10
  → training took 7519 seconds (35000→40000 iterations)