NLP with Deep Learning - Assignment 2 - CS224N Stanford

LIU Vincent

March 22, 2021

word2vec assignment from CS224N 2019 Winter session

(a)

$$
\begin{aligned}
-\sum_{w\in Vocab} y_w log(\hat{y}_w) &= -y_o log(\hat{y}_o) - \sum_{w\in Vocab, w\neq o} y_w log(\hat{y}_w) \\
&= -1 \times log(\hat{y}_o) - \sum_{w\in Vocab, w\neq o} 0 \times (\hat{y}_w) \\
&= -1 \times log(\hat{y}_o) \\
&= -log(\hat{y}_o)
\end{aligned}
$$

since y is an one-hot representation of the outside word o.

(b)

$$
\begin{aligned}
J_{\text{naive softmax}}(v_c, o, U) &= -log(\hat{y}_o) \\
&= -log\left(\frac{exp(u_o^t v_c)}{\sum_w exp(u_w^t v_c)}\right) \\
&= -u_o^t v_c + log\left(\sum_w exp(u_w^t v_c)\right)
\end{aligned}
$$

$$
\frac{\partial J_{\text{naive softmax}}(v_c, o, U)}{\partial v_c} = -u_o + \sum_x \hat{y_x} u_x
$$

(c)

$$
\begin{aligned}
\frac{\partial J_{\text{naive softmax}}(v_c, o, U)}{\partial u_o} &= v_c(\hat{y_o} - 1) \\
\frac{\partial J_{\text{naive softmax}}(v_c, o, U)}{\partial u_{w\neq o}} &= v_c \hat{y}_{w\neq o}
\end{aligned}
$$

(d) $\sigma' = \sigma(1-\sigma)$

(e)
$J_{\text{neg sample}} = -log(\sigma(u_o^t v_c)) - \sum_k log(\sigma(-u_k^t v_c))$

$$\frac{\partial J_{\text{neg sample}}(v_c, o, U)}{\partial v_c} = (\sigma(u_o^t v_c) - 1)u_o + \sum_k (1 - \sigma(-u_k^t v_c))u_k$$
$$\frac{\partial J_{\text{neg sample}}(v_c, o, U)}{\partial u_o} = (\sigma(u_o^t v_c) - 1)v_c$$
$$\frac{\partial J_{\text{neg sample}}(v_c, o, U)}{\partial u_k} = \#_k \times (1 - \sigma(-u_k^t v_c))v_c$$

Note: The same word may be negatively sampled multiple times. For example if an outside word is sampled twice, you shall have to double count the gradient with respect to this word. Thrice if it was sampled three times, and so forth.

Note 2: Computing $J_{\text{neg sample}}$ is more efficient since it requires only O(K) negative samples while computing $J_{\text{naive softmax}}$ requires normalization, which cost O(Vocab size) operations.

(f) trivial (derivative of sum is sum of derivatives)