Problem Sets 1

Sung uk Jung / 20215338

May 4, 2022

Submission Instructions: You shall submit this assignment on BlackBoard as two submission files – your code as assignment2coding.zip and write up for assignment2.pdf; Run the collect_submission.sh script to produce your assignment2.zip file. It is your responsibility to make sure your code is runnable. If your code is not runnable, you will get no point.

1 Written: Understanding word2vec (60 points)

Recall that the key insight behind word2vec is that a word is known by the company it keeps'. Concretely, consider a 'center' word c surrounded before and after by a context of a certain length. We term words in this contextual window 'outside words' (O). For example, in Figure 1, the context window length is 2, the center word c is 'banking', and the outside words are 'turning', 'into', 'crises', and 'as':

Skip-gram word2vec aims to learn the probability distribution $P(O|C)$. Specifically, given a specific word o and a specific word c, we want to predict $P(O = o|C = c)$: the probability that word o is an 'outside' word for c (i.e., that it falls within the contextual window of c). We model this probability by taking the softmax function over a series of vector dot-products:

$$P(O = o|C = c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in \text{Vocab}} \exp(u_w^T v_c)}$$

For each word, we learn vectors u and v, where u_o is the 'outside' vector representing outside word o, and v_c is the 'center' vector representing center

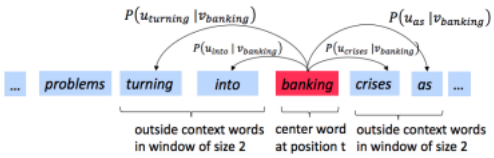

Figure 1: The word2vec skip-gram prediction model with window size 2

word c. We store these parameters in two matrices, U and V. The columns of U are all the 'outside' vectors u_w; the columns of V are all of the 'center' vectors v_w. Both U and V contain a vector for every $w \in$ Vocabulary. Recall, for a single pair of words c and o, the loss is given by:

$$J_{\text{niave}-\text{softmax}} = -\log P(O = o|C = c).$$

we can view this loss as the cross-entropy between the true distribution y and the predicted distribution $\hat{y}$,for a particular center word c and a particular outside word o. Here, both y and $\hat{y}$ 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 y is a one-hot vector with a 1 for the true outside word o, and 0 everywhere else, for this particular example of center word c and outside word o^3 . The predicted distribution $\hat{y}$ is the probability distribution $P(O|C = c)$ given by our model in equation 1.

(a) (6 points) Prove that the naive-softmax loss (Equation 2) is the same as the cross-entropy loss between y and $\hat{y}$, i.e. (note that y, $\hat{y}$ are vectors and $\hat{y}_o$ is a scalar):

$$-\sum_{w \in \text{Vocab}} y_w \log(\hat{y_w}) = -\log(\hat{y_o}).$$

Your answer should be one line. You may describe your answer in words.
Answer:

$$\sum_{w \in \text{Vocab}} y_w \log(\hat{y}_w) = -y_o \log(\hat{y_o}) - \sum_{w \in \text{Vocab}, \text{w} \neq \text{o}} y_w \log(\hat{y}_w) = -\log(\hat{y_o})$$

(b) (10 points) Compute the partial derivative of $J_{niave-softmax}(v_c, o, U)$ with respect to v_c. Please write your answer in terms of $y, \hat{h}$ and U.
Answer:

$$\begin{aligned}
\frac{\delta J_{\text{naive}-\text{softmax}(v_c,o,U)}}{\delta v_c} &= -\frac{\delta}{\delta v_c} \log \frac{e^{u_o^T v_c}}{\sum_{w \in Vocab} e^{u_w^T v_c}} \\
&= -\frac{\delta}{\delta v_c} \left(\log(e^{u_o^T v_c}) - \log \sum_{w \in V} e^{u_w^T v_c} \right) \\
&= -\frac{\delta}{\delta v_c} \left(u_o^T v_c - \log \sum_{w \in V} e^{u_w^T v_c} \right) \\
&= -u_o + \sum_{w \in V} \frac{u_w e^{u_w^T v_c}}{\sum_{w \in V} p(O = w|C = c) u_w} \\
&= -u_o + \sum_{w \in V} \hat{y}_w u_w \\
&= U(\hat{y} - y)
\end{aligned}$$

Additionally, answer the following two questions with one sentence each:

(1) When is the gradient zero?

Answer: y and $\hat{y}$ are same.

(2) Why does subtracting this gradient, in the general case when it is nonzero, make v_c a more desirable vector(namely, a vector closer to outside word vectors in its window)?

Answer: To update weights of back layers. The mean of zero gradient is that the model decide perfectly correct answer.

(c) (10 points) Compute the partial derivatives of $J_{native-sfotmax}(v_c, o, U)$ with respect to each of the 'outside' word vectors, 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 your answer in terms of $y, \hat{y}$ and v_c. In this subpart, you may use specific elements within these terms as well (such as $y_1, y_2, ...$). Note that u_w is a vector while $y_1, y_2, ...$ are scalars.

Answer:

$$\begin{aligned}\frac{\delta J(v_c, o, U)}{\delta u_w} &= -\frac{\delta}{\delta u_w}\left(u_o^T v_c - \log \sum_{w \in Vocab} e^{u_w^T v_c}\right) \\ &= -\frac{\delta u_o^T v_c}{\delta u_w} + \frac{\delta(\log(\sum_{w \in Vocab} e^{u_w^T v_c}))}{\delta u_w}\end{aligned}$$

when w is o then, we can get

$$\begin{aligned}\frac{\delta J(v_c, o, U)}{\delta u_w} &= -v_c + p(O = w|C = c)v_c \\ &= \hat{y}_w v_c - v_c \\ &= (\hat{y}_w - 1)v_c\end{aligned}$$

$$\frac{\delta J(v_c, o, U)}{\delta U} = v_c(\hat{y} - y)^T$$

And the other case w is not equal to o then,

$$\begin{aligned}\frac{\delta J(v_c, o, U)}{\delta u_w} &= 0 + p(O = w|C = c)v_c \\ &= \hat{y}_w v_c\end{aligned}$$

(d) (2 point) Write down the partial derivative of $J_{\text{naive}-\text{softmax}}(v_c, o, U)$ with respect to U. Please break down your answer in terms of $\frac{\delta J(v_c,o,U)}{\delta u_1}, \frac{\delta J(v_c,o,U)}{\delta u_2}, ..., \frac{\delta J(v_c,o,U)}{\delta u_{|\text{Vocab}|}}$. The solution should be one or two lines long.

Answer: $\frac{\delta J(v_c,o,U)}{\delta u_1} = -(\hat{y}_1 - 1)v_c, \frac{\delta J(v_c,o,U)}{\delta u_2} = -(\hat{y}_2 - 1)v_c, ..., \frac{\delta J(v_c,o,U)}{\delta u_{|\text{Vocab}|}} = -(\hat{y}_{\text{vocab}} - 1)v_c$

(e) (4 points) The ReLU (Rectified Linear Unit) activation function is given by Equation 4:

$$f(x) = max(0, x)$$

Please compute the derivative of $f(x)$ with respect to x, where x is a scalar. You may ignore the case that the derivative is not defined at 0.

Answer : $\frac{d}{dx}f(x) = 1$

(f) (6 points) The sigmoid function is given by Equation 5:

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

Please compute the derivative of $\sigma(x)$ with respect to x, where x is a scalar. Hint: you may want to write your answer in terms of $\sigma(x)$.

Answer:

$$\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{d}{dx}\left(1+e^{-x}\right)^{-1} \\
&= \frac{-e^{-x}}{-(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}}\frac{e^{-x}+(1-1)}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}}\frac{(e^{-x}+1)-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right) \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}$$

$$\frac{\delta\sigma(x)}{\delta x} =$$

$$\begin{vmatrix}
\sigma'(x_1) & 0 & 0 & ... & 0 \\
0 & \sigma'(x_2) & 0 & ... & 0 \\
... & ... & ... & ... & ... \\
0 & 0 & 0 & ... & \sigma'(x_d)
\end{vmatrix} = diag(\sigma'(x_d))$$

(g) (12 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 $w_1, W_2, ..., w_k$ and their outside vectors as $u_{w_1}, u_{w_2}, ..., u_{w_k}$. For this question, assume that the K negative samples are distinct. . In other words, $i \neq j$ implies $w_i \neq w_j$ for $i, j \in 1, ..., K$. Note that $o \notin \{w_1, w_2, ..., w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

$$J_{neg-sample}(v_c, o, U) = -\log(\sigma(u_o^T v_c)) - \sum_{s=1}^{K}(\sigma(-u_{w_s}^T v_c))$$

for a sample $w_1, w_2, ..., w_k$ where $sigma(\cdot)$ is the sigmoid function.

(i) Please repeat parts (b) and (c), computing the partial derivatives of $J_{neg-smaple}$ with respect to v_C, with respect to u_o, and with respect to the s^{th} negative sample u_{ws}. Please write your answers in terms of the vectors v_c, u_o and u_{w_s}, where $s \in [1, K]$. Note: you should be able to use your solution to part (f) to help compute the necessary gradients here.

Answer:

$$\frac{dJ_{neg-softmax}(v_c, o, U)}{dv_c} = (\sigma(u_o^T v_c) - 1)u_o + \sum_{k=1}^{K}(1 - \sigma(-u_k^T v_c))u_k$$
$$= (\sigma(u_o^T) - 1)u_o + \sum_{k=1}^{K}\sigma(-u_k^T v_c)u_k$$

$$\frac{\delta J_{neg-sample}}{\delta u_o} = (\sigma(u_o^T v_c) - 1)v_c$$

For u_k:

$$\frac{\delta J_{neg-sample}}{\delta u_k} = (1 - \sigma(u_k^T v_c))v_c = \sigma(u_k^T v_c)v_c$$

(ii) In lecture, we learned that an efficient implementation of backpropagation leverages the reuse of previously-computed partial derivatives. Which quantity could you reuse between the three partial derivatives to minimize duplicate computation? Write your answer in terms of $U_{o\{w_1,...,w_K\}} = [u_o, -u_{w_1}, ..., -u_{w_k}]$, a matrix with the outside vectors stacked as columns, and 1, $a(K+1) \times 1$ vector of 1's.

Answer:

(iii) Describe with one sentence why this loss function is much more efficient to compute than the naive-softmax loss.

Answer: The loss function just choose some of negative sample and this only calculates loss on the negative sample while the softmax calculate loss from all of the words.

Caveat: So far we have looked at re-using quantities and approximating softmax with sampling for faster gradient descent. Do note that some of these optimizations might not be necessary on modern GPUs and are, to some extent, artifacts of the limited compute resources available at the time when these algorithms were developed.

(h) (4 points) Now we will repeat the previous exercise, but without the assumption that the K sampled words are distinct. Assume that K negative samples (words) are drawn from the vocab-ulary. For simplicity of notation we shall refer to them as $w_1, w_2, ..., w_K$ and their outside vectors as $u_{w_1}, ...u_{w_k}$. In this question, you may not assume that the words are distinct. In other words, $w_i = w_j$ may be true when $i \neq j$ is true. Note that $o \notin \{w_1, w_2, ..., w_k\}$. For a center word c and an outside word o, the negative sampling loss function is

given by:

$$J_{neg-sample}(v_c, o, U) = -\log(\sigma(u_o^T v_c)) - \sum_{s=1}^{K} \log(\sigma(-{u_w}_s^T v_c))$$

for a sample $w_1, w_2, ..., w_k$, where $\sigma(\cdot)$ is the sigmoid function. Compute the partial derivative of $J_{neg-sample}$ with respect to a negative sample u_{ws}. Notationwise, you may write 'equal' and 'not equal' conditions below the summation symbols, such as in Equation 8.

Answer:

(i) (6 points) Suppose the center word is c = wt and the context window is $[w_{t-m}, ..., w_{t-1}, w_t, w_{t+1}, w_{t+m}]$, where m is the context window size. Recall that for the skip-gram version of word2vec, the total loss for the context window is:

$$J_{skip-gram}(v_c, w_{t-m}, ..., w_{t+m}, U) = \sum_{-m \le j \le m, j \neq 0} J(v_c, w_{t+j}, U)$$

Here, $J(v_c, w_{t+j}, U)$ represents an arbitrary loss term for the center word $c = w_t$ and outside word w_{t+j}. $J(v_c, w_{t+j}, U)$ could be $J_{naive-softmax}(v_c, w_{i+j}, U)$ or $J_{neg-sample}(v_c, w_{t+j}, U)$, depending on your implementation. Write down three partial derivatives:

$$(1) \frac{\delta J_{skip-gram}(v_c, w_{t-m}, ..., w_{t+m}, U)}{\delta U}$$

$$(2) \frac{\delta J_{skip-gram}(v_c, w_{t-m}, ..., w_{t+m}, U)}{\delta v_c}$$

$$(3) \frac{\delta J_{skip-gram}(v_c, w_{t-m}, ..., w_{t+m}, U)}{\delta v_w} \text{ (when w} \neq \text{ c)}$$

Write your answers in terms of $\frac{\delta J_{skip-gram}(v_c, w_{i+j}, U)}{\delta U}$ and $\frac{\delta J_{skip-gram}(v_c, w_{i+j}, U)}{\delta v_c}$. This is very simple – each solution should be one line. Once you're done: Given that you computed the derivatives of $J(v_C, w_{t+j}, U)$ with respect to all the model parameters U and V in parts (a) to (c), you have now computed the derivatives of the full loss function $J_{skip-gram}$ with respect to all parameters. You're ready to implement word2vec!

Answer:

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

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

$$\frac{\delta J_{skip-gram}}{\delta v_w} = 0 \text{ (when w} \neq \text{ c)}$$