CS 482/682: Machine Learning: Deep Learning

Lab #2 Note: Self-supervised Learning and Word Embeddings

TA: Shijie Wu

1 Introduction

In deep learning, we want to represent the object of interest as a vector. For computer vision, it is straightforward to represent an image as a vector since an image is composed of RGB values at each pixel. However, not all objects of interest are naturally represented by a number, for example, words in natural language and users in the recommendation system. Indeed, both of them are discrete items with a limited total number. To solve it, a vector is associated with each *unique* item and the vector is called **embedding**. In practice, the embedding of all unique items is usually stored as a matrix (**embeddings matrix**), where each row of the matrix represents the embedding of an item. The number of rows of the embeddings matrix is the total number of items.

Take natural language as an example, the objects of interest are words. Assuming we have a list of toy vocabulary $\{\text{apple}, \text{orange}, \text{berry}\}$, each word in the vocabulary has an implicit index, which is the same as the order of word in the list. So the indexes of apple, orange and berry are 0, 1 and 2, respectively. Assuming we associate each word with a 4 dimension vector, the embeddings matrix has the shape of 3-by-4. The first, second and third rows of the matrix are the word embeddings of apple, orange and berry, respectively. In the following note, we will represent a word as an index.

How do we learn word embeddings? We want to train the model to predict a word given its context. Continuous bag-of-words (CBOW) follows the same idea. Take the sentence "Yesterday President Obama won re-election" as an example. Given window size of 1, CBOW is trained to have higher similarity between the context bag-of-word embeddings, the sum of word embeddings of "President" and "won", and word embedding of "Obama" compared to other word embeddings. Namely, CBOW is trained to predict the word "Obama" given the context of "President" and "won". Note the CBOW ignores the ordering of the context words.

CBOW is a form of **self-supervised learning**. The idea of self-supervised learning is to learn vectors (usually called **representation**) for each element in the data based on data alone without any manual label or supervision. Self-supervised learning is appealing since data without labels like images and sentences is widely available on the internet. One goal of self-supervised learning is that we want similar items to have vectors with high similarity so that we can learn a model to performs tasks of interest. Indeed, in the previous example, "Bush" would also be a good solution. As a result, the word embedding of "Bush" and "Obama" would be similar. Self-supervised learning has been driving recent breakthroughs in natural language processing with BERT and computer vision with MoCo. BERT follows the same idea as CBOW with clever implementation, deeper neural network, and more data.

2 CBOW

In this section, we discuss the formulation of CBOW and its implementation.

2.1 Prelimary

Let the list of the vocabulary be $\mathcal{V}$. We associate each word w in the vocabulary with two vectors: input embeddings v_w and output embeddings u_w. The input embeddings are used to computed context bag-of-word embeddings and the output embeddings are used for computing the probability distribution over vocabulary given context bag-of-word embeddings. Let the input embeddings matrix be V and the output embeddings matrix is U. U and V has the same shape of $|\mathcal{V}| \times D$, where D is the dimension of input embeddings and output embeddings. Recall that word w is an index, so the w^{th} row of U is u_w and w^{th} row of V is v_w.

2.2 CBOW Loss

Assuming the window size is 2 and we have a slice of words from a sentence w_{-2}, w_{-1}, w_c, w_1, w_2. CBOW is trained to predict the center word w_c given the context words $\{w_{-2}, w_{-1}, w_1, w_2\}$. The context bag-of-word (bow) embeddings is the element-wise sum of all input embeddings of the context words.

$$v_{bow} = v_{w_{-2}} + v_{w_{-1}} + v_{w_1} + v_{w_2} \tag{2.1}$$

For all $w \in \mathcal{V}$, we can measure the similarity between v_{bow} and u_w using dot-product $s_{\{v_{bow}, u_w\}} = v_{bow} \cdot u_{w_c}$. Then, we can use softmax function the normalize the similarity to get a probability distribution over vocabulary given v_{bow}

$$p(w_c|v_{bow}) = \frac{\exp(s_{\{v_{bow}, u_{w_c}\}})}{\sum_{w_j \in \mathcal{V}} \exp(s_{\{v_{bow}, u_{w_j}\}})} \tag{2.2}$$

$$= \frac{\exp(v_{bow} \cdot u_{w_c})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \tag{2.3}$$

Finally, to train CBOW to predict the center word w_c given the context words $\{w_{-2}, w_{-1}, w_1, w_2\}$, we want to minimize the negative log likelihood of the observed data (loss $\mathcal{L}_{w_c}$) so that $p(w_c|v_{bow})$ would be high.

$$\mathcal{L}_{w_c} = -\log p(w_c|v_{bow}) \tag{2.4}$$

In practice, we take the average over $\mathcal{L}_{w_c}$ of a batch of example, as loss of the batch ($\mathcal{L}$).

2.3 CBOW Gradient

For the previous example of w_{-2}, w_{-1}, w_c, w_1, w_2, we first need the work out the gradient of $-\log p(w_c|v_{bow})$ with respect to u_{w_c}, u_w where $\forall w \in \mathcal{V}, w \neq w_c$ and v_w where $w \in \{w_{-2}, w_{-1}, w_1, w_2\}$.

Problem Try to work the gradient of CBOW

$$\frac{\partial}{\partial u_{w_c}} - \log p(w_c|v_{bow}) = \frac{\partial}{\partial u_{w_c}} - \log \frac{\exp(v_{bow} \cdot u_{w_c})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \tag{2.5}$$

$$\frac{\partial}{\partial u_{w}} - \log p(w_c|v_{bow}) = \frac{\partial}{\partial u_{w}} - \log \frac{\exp(v_{bow} \cdot u_{w_c})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \tag{2.6}$$

$$\frac{\partial}{\partial v_{w}} - \log p(w_c|v_{bow}) = \frac{\partial}{\partial v_{w}} - \log \frac{\exp(v_{bow} \cdot u_{w_c})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \tag{2.7}$$

Hint: Some useful equations:

$$\begin{aligned}
\frac{\partial}{\partial v} u \cdot v &= u \\
\frac{\partial}{\partial u} u \cdot v &= v \\
\frac{\partial}{\partial x} \exp f(x) &= \exp \frac{\partial}{\partial x} f(x) \\
\frac{\partial}{\partial x} \log f(x) &= \frac{1}{f(x)} \frac{\partial}{\partial x} f(x)
\end{aligned}$$

To train CBOW, we need to compute the gradient of $\mathcal{L}_{w_c}$ with respect to U and V. The gradient matrix of U and V has the same shape as U and V, namely $|\mathcal{V}| \times D$. The initial value of gradient matrix is 0.

- The ${w_c}^{th}$ row of the gradient matrix of U is $\frac{\partial}{\partial u_{w_c}} - \log p(w_c|v_{bow})$
- $\forall w \in \mathcal{V}, w \neq w_c$, the w^{th} row of the gradient matrix of U is $\frac{\partial}{\partial u_{w}} - \log p(w_c|v_{bow})$
- $\forall w \in \{w_{-2}, w_{-1}, w_1, w_2\}$, the w^{th} row of the gradient matrix of V is $\frac{\partial}{\partial v_{w}} - \log p(w_c|v_{bow})$

Finally, the gradient of $\mathcal{L}$ with respect to U and V are the average of each gradient matrix of $\mathcal{L}_{w_c}$ with respect to U and V, respectively.

2.4 Implementation

Variable Description Let WS be window size. In the notebook, `batch_x` is a matrix containing a batch of context words (represented as indexes). `batch_y` is a vector containing a batch of center word (also represented as indexes). The i^{th} row of the matrix `batch_x` is a vector of context words ($\{w_{-WS}, \cdots, w_{-2}, w_{-1}, w_1, w_2, \cdots, w_{WS}\}$) while the i^{th} element of the vector `batch_y` is the corresponding center word (w_c).

The `input_embed` is V and `output_embed` is U. The gradient matrix of V is `input_embed_grad` and the gradient matrix of U is `output_embed_grad`. In the function `forward_and_backward`, the forward pass refer to the computation of CBOW loss while the backward pass refer to the computation of CBOW gradient.

Numerical Stability of Softmax and Log Let $X = \{x_1, \cdots, x_N\}$, when x_i is to small, it could cause numerical instability when taking log of softmax. As a result, we can use the following tricks to compute sotfmax and log jointly. Let $b = \max(X)$,

$$\begin{aligned}
\log p(x_i|X) &= \log \frac{\exp(x_i)}{\sum_j \exp(x_j)} \\
&= \log \frac{\exp(x_i)/\exp(b)}{\sum_j \exp(x_j)/\exp(b)} \\
&= \log \frac{\exp(x_i - b)}{\sum_j \exp(x_j - b)} \\
&= \log \exp(x_i - b) - \log \sum_1^N \exp(x_j - b) \\
&= x_i - (b + \log \sum_1^N \exp(x_j - b))
\end{aligned}$$

SPOILER ALERT:

Please First Try Working Out The Gradient Before Viewing The Solution

3 Solution: Gradient of CBOW

$$\frac{\partial}{\partial u_{w_c}} - \log p(w_c|v_{bow}) = \frac{\partial}{\partial u_{w_c}} - \log \frac{\exp(v_{bow} \cdot u_{w_c})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \tag{3.1}$$

$$= \frac{\partial}{\partial u_{w_c}} - \log \exp(v_{bow} \cdot u_{w_c}) + \log \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.2}$$

$$= -\frac{\partial}{\partial u_{w_c}} v_{bow} \cdot u_{w_c} + \frac{\partial}{\partial u_{w_c}} \log \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.3}$$

$$= -v_{bow} + \frac{1}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial u_{w_c}} \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.4}$$

$$= -v_{bow} + \frac{1}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial u_{w_c}} \exp(v_{bow} \cdot u_{w_c}) \tag{3.5}$$

$$= -v_{bow} + \frac{\exp(v_{bow} \cdot u_{w_c})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial u_{w_c}} v_{bow} \cdot u_{w_c} \tag{3.6}$$

$$= -v_{bow} + p(w_c|v_{bow}) v_{bow} \tag{3.7}$$

$$\frac{\partial}{\partial u_{w}} - \log p(w_c|v_{bow}) = \frac{\partial}{\partial u_{w}} - \log \frac{\exp(v_{bow} \cdot u_{w_c})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \tag{3.8}$$

$$= \frac{\partial}{\partial u_{w}} - \log \exp(v_{bow} \cdot u_{w_c}) + \log \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.9}$$

$$= -\frac{\partial}{\partial u_{w}} v_{bow} \cdot u_{w_c} + \frac{\partial}{\partial u_{w}} \log \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.10}$$

$$= \frac{1}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial u_{w}} \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.11}$$

$$= \frac{1}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial u_{w}} \exp(v_{bow} \cdot u_{w}) \tag{3.12}$$

$$= \frac{\exp(v_{bow} \cdot u_{w})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial u_{w}} v_{bow} \cdot u_{w} \tag{3.13}$$

$$= p(w|v_{bow}) v_{bow} \tag{3.14}$$

Recall $v_{bow} = v_{w_{-2}} + v_{w_{-1}} + v_{w_1} + v_{w_2}$. For each $w \in \{w_{-2}, w_{-1}, w_1, w_2\}$

$$\frac{\partial}{\partial v_w} - \log p(w_c|v_{bow}) = \frac{\partial}{\partial v_w} - \log \frac{\exp(v_{bow} \cdot u_{w_c})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \tag{3.15}$$

$$= \frac{\partial}{\partial v_w} - v_{bow} \cdot u_{w_c} + \log \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.16}$$

$$= -\frac{\partial}{\partial v_w} v_{bow} \cdot u_{w_c} + \frac{\partial}{\partial v_w} \log \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.17}$$

$$= -u_{w_c} + \frac{1}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial v_w} \sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j}) \tag{3.18}$$

$$= -u_{w_c} + \sum_{w_j \in \mathcal{V}} \frac{1}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial v_w} \exp(v_{bow} \cdot u_{w_j}) \tag{3.19}$$

$$= -u_{w_c} + \sum_{w_j \in \mathcal{V}} \frac{\exp(v_{bow} \cdot u_{w_j})}{\sum_{w_j \in \mathcal{V}} \exp(v_{bow} \cdot u_{w_j})} \frac{\partial}{\partial v_w} v_{bow} \cdot u_{w_j} \tag{3.20}$$

$$= -u_{w_c} + \sum_{w_j \in \mathcal{V}} p(w_j|v_{bow}) u_{w_j} \tag{3.21}$$