Word Embeddings

Rohit Amarnath

December 5, 2023

1 Continous Bag of Words

The CBOW model takes a window of surrounding words as input and tries to predict the target word in the center of the window.

Let V be the vocabulary, n be the embedding size, $E = \begin{bmatrix} | & & | \\ e_1 & \dots & e_{|V|} \\ | & & | \end{bmatrix} \in \mathbb{R}^{n \times |V|}$ be the input word matrix and $U = \begin{bmatrix} — & u_1 & — \\ & \vdots & \\ — & u_{|V|} & — \end{bmatrix} \in \mathbb{R}^{|V| \times n}$ be the output word matrix. The input word matrix can alternatively be interpreted as the word embedding layer. Each column in this matrix is a word embedding.

1.1 Methodology

1. The input is a $2m$ size list of words $(x_{c-m}, \dots, x_{c-1}, x_{c+1}, \dots, x_{c+m})$. Each word is a number, and we will one hot encode these vectors before feeding them into our model. The output is $x_c = k$, and it is also oen hot encoded.

2. We get the embedded word vectors

$$(v_{c-m}, \dots, v_{c-1}, v_{c+1}, \dots, v_{c+m}) = (Ex_{c-m}, \dots, Ex_{c-1}, Ex_{c+1}, \dots, Ex_{c+m})$$

   Then, average the embedded vectors.

$$\bar{v} = \frac{v_{c-m} + \cdots + v_{c-1} + v_{c+1} + \cdots + v_{c+m}}{2m}$$

3. Generate a score vector $z = U\bar{v}$ and turn the scores into probabilities with the softmax function. We have $\hat{y} = \text{softmax}(z)$.

4. Cross-entropy loss is used to train the model.

$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

   Since $x_c = k$ as defined in the first step, the loss is $H(\hat{y}, y) = -y_c \log(\hat{y}_c)$. Thus, with gradient descent, we optimize

$$\begin{aligned} H(\hat{y}, y) &= -\log \hat{y}_k \\ &= -\log \frac{\exp(u_k^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\ &= -u_k^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v}) \end{aligned}$$

2 Skip-grams

The skip-gram is the inverse of the CBOW model. In the skip-gram, the input is a word, and the model tries to predict the context window surrounding the word.

As before, let V be the vocabulary, n be the embedding size, $E = \begin{bmatrix} | & & | \\ e_1 & \dots & e_{|V|} \\ | & & | \end{bmatrix} \in \mathbb{R}^{n \times |V|}$ be the input word matrix and $U = \begin{bmatrix} — & u_1 & — \\ & \vdots & \\ — & u_{|V|} & — \end{bmatrix} \in \mathbb{R}^{|V| \times n}$ be the output word matrix.

2.1 Methodology

There is an abuse of notation with the indexing in this section. When we index by i, we actually index by the word index. Let $x_i = k$, which means x_i is the k-th word in the vocabulary. The notation used in this section allows $u_i = u_{x_i}$ and $e_i = e_{x_i}$.

The skip-gram model learns the probability $P(x_{c-m}, \dots, x_{c-1}, x_{c+1}, \dots, x_{c+m} \mid x_c)$. We invoke the naive Bayes assumption and assume conditional independence between context words. That is,

$$P(x_{c-m}, \dots, x_{c-1}, x_{c+1}, \dots, x_{c+m} \mid x_c) = \prod_{i=-m, i \neq 0}^{m} P(x_{c+i} \mid x_c)$$

We want to minimize the negative of the above. The skip-gram model can be thought of as training a hidden representation (the word embedding) by training a softmax regression on $2m - 1$ outputs for one input.

We now go over the model.

1. The input is a word. As before, the word is a number, but we will one hot encode this before feeding it into the model. Let the input be x_c and the output be $x_{c-m}, \dots, x_{c-1}, x_{c+1}, \dots, x_{c+m}$.
2. We retrieve the embedded word vector $v_c = Ex_c$. Now, for each word in the context, we calculate the softmax probability. That is, for the output word x_{c-m}, we calculate

$$\begin{aligned} P(x_{c-m} \mid x_c) &= P(u_{c-m} \mid v_c) \\ &= \frac{\exp\left(u_{c-m}^T v_c\right)}{\sum_{i=1}^{|V|} \exp\left(u_i^T v_c\right)} \end{aligned}$$

   We now have a set of probabilities for every word in the context.

3. With gradient descent, we minimize the negative log of $P(x_{c-m},\dots,x_{c-1},x_{c+1},\dots,x_{c+m}\mid x_c)$.

$$\begin{aligned}
-\log P(x_{c-m},\dots,x_{c-1},x_{c+1},\dots,x_{c+m}\mid x_c) &= -\log \prod_{i=-m,i\neq 0}^{m} P(x_{c+i}\mid x_c)\\
&= -\log \prod_{i=-m,i\neq 0}^{m} P(u_{c+i}\mid v_c)\\
&= -\sum_{i=-m,i\neq 0}^{m} \log P(u_{c+i}\mid v_c)\\
&= -\sum_{i=-m,i\neq 0}^{m} \log \frac{\exp\left(u_{c+i}^T v_c\right)}{\sum_{i=1}^{|V|}\exp\left(u_i^T v_c\right)}\\
&= -\sum_{i=-m,i\neq 0}^{m} u_{c+i}^T v_c + 2m\log\sum_{i=1}^{|V|}\exp\left(u_i^T v_c\right)
\end{aligned}$$

We can also think of this as finding the cross entropy loss with respect to $2m-1$ output vectors, showing the correspondence between cross entropy loss and the maximum likelihood.

3 Negative Sampling

Skip-grams work well, but they require a $O(|V|)$ update per word and context pair. To mitigate this, we reframe the prediction problem from maximizing likelihoods of pairs of context and a word to determining whether a pair of words is a pair of context and a word. Observe that this new paradigm has changed the existing $|V|$ classes classification task to a binary classification task (whether a pair of words is *valid* or not). Negative sampling is a special case of noise contrastive estimation and is a contrastive representation learning technique. We present it now.

Denote $(w,c)\in\mathcal{D}$ be a pair of word and context. Let $P(D=1\mid w,c)$ be the probability that pair (w,c) came from the corpus.

$$P(D=1\mid w,c)=\frac{1}{1+\exp(-u_w^T v_c)}$$

Similarly, $P(D=0\mid w,c)$ is the probability that pair (w,c) did not come from the corpus.

Let $\widetilde{\mathcal{D}}$ be the set of pairs of word and context that are erroneous and did not come from the corpus. With these probabilities, we have the maximum likelihood to be

$$\begin{aligned}
\theta_{\text{MLE}} &= \operatorname*{argmax}_\theta \prod_{(w,c)\in\mathcal{D}} P(D=1\mid w,c)\prod_{(w,c)\in\widetilde{\mathcal{D}}} P(D=0\mid w,c)\\
&= \operatorname*{argmax}_\theta \sum_{(w,c)\in\mathcal{D}} \log P(D=1\mid w,c)+\sum_{(w,c)\in\widetilde{\mathcal{D}}}\log(1-P(D=1\mid w,c))\\
&= \operatorname*{argmax}_\theta \sum_{(w,c)\in\mathcal{D}} \log\frac{1}{1+\exp(-u_w^T v_c)}+\sum_{(w,c)\in\widetilde{\mathcal{D}}}\log\left(1-\frac{1}{1+\exp(-u_w^T v_c)}\right)\\
&= \operatorname*{argmax}_\theta \sum_{(w,c)\in\mathcal{D}} \log\frac{1}{1+\exp(-u_w^T v_c)}+\sum_{(w,c)\in\widetilde{\mathcal{D}}}\log\frac{1}{1+\exp(u_w^T v_c)}
\end{aligned}$$

The objective function ends up being

$$J=-\log\frac{1}{1+\exp(-u_w^T v_c)}-\sum_{i=1}^{K}\log\frac{1}{1+\exp\left(u_i^T v_c\right)}$$

where we take k samples from $\widetilde{\mathcal{D}}$ and $u_1, \ldots, u_k$ are the output embeddings of the *negative samples* and v_c is the input word. Note that $(w, c) \in \mathcal{D}$ and $(i, c) \in \widetilde{\mathcal{D}}$.

4 GloVe: Global Vectors

Denote X to be the co-occurence matrix of vocabulary V. X is a $|V| \times |V|$ matrix, and X_{ij} denotes the count of word i being within the context of word j and vice versa (note that $X_{ij} = X_{ji}$). The GloVe algorithm has two $K \times N$ matrices W and $\widetilde{W}$ which factorize the co-occurence matrix X into a K dimensional subspace.

The GloVe algorithm solves the following prediction problem:

$$W_i^T \widetilde{W}_j + b_i + \tilde{b}_j = \log X_{ij}$$

where W_i^T is the i-th column of W, $\widetilde{W}_j$ is the j-th column of $\widetilde{W}$, and b_i and b_j are bias terms for words i and j, respectively.

The loss function is

$$J = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} f(X_{ij})(W_i^T \widetilde{W}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

The GloVe paper has a more specific derivation of the loss function, which we will not present here. After training, we compute $\frac{W + \widetilde{W}}{2}$ to get the word embeddings.