Word2Vec Model

Serry Sibaee
COOP Student (researcher)

Index

- history and types of word2vec
- simple one word CBOW
- multi words CBOW
- skip gram (preview)
- optimization algorithms
- heiriracy softmax
- negative sampling
- TF-IDF

word2vec model history and types

It was invented in 2013 to represent the words in a dense vector representation in around 100 length in a faster way and it has two types: continuous bag of words and skip gram models where the first one focus on predicting the word from context and the other one is vise versa

CBOW one word context

$$\mathbf{h} = \mathbf{W}^T \mathbf{x} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I},$$

$$u_j = {\mathbf{v}'_{w_j}}^T \mathbf{h},$$

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})},$$

we obtain

$$p(w_j|w_I) = \frac{\exp\left({\mathbf{v}'_{w_j}}^T \mathbf{v}_{w_I}\right)}{\sum_{j'=1}^{V} \exp\left({\mathbf{v}'_{w_{j'}}}^T \mathbf{v}_{w_I}\right)}$$

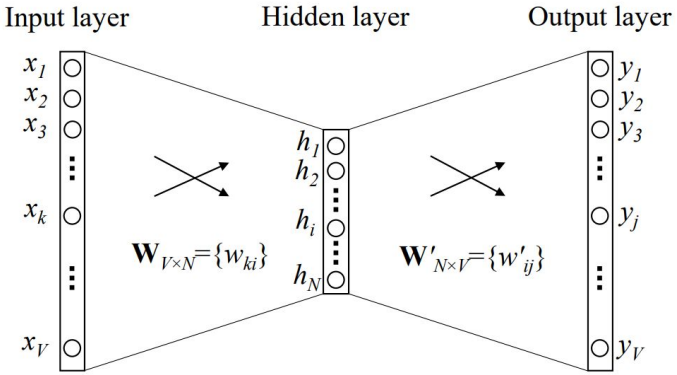


Figure 1: A simple CBOW model with only one word in the context

From the input layer to hidden called “input vector”
From Hidden to output “output vector”

Loss for one word CBOW

Update equation for hidden→output weights

$$\begin{aligned}\max p(w_O|w_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E,\end{aligned}$$

where $E = -\log p(w_O|w_I)$ is our loss function (we want to minimize E), and j^* is the index of the actual output word in the output layer. Note that this loss function can be understood as a special case of the cross-entropy measurement between two probabilistic distributions.

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j \quad \frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i$$

Therefore, using stochastic gradient descent, we obtain the weight updating equation for hidden→output weights:

$$w'^{(\text{new})}_{ij} = w'^{(\text{old})}_{ij} - \eta \cdot e_j \cdot h_i. \tag{10}$$

or

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V. \tag{11}$$

Loss for one word CBOW

Update equation for input→hidden weights

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathrm{EH}_i$$

Next we should take the derivative of E on $\mathbf{W}$. First, recall that the hidden layer performs a linear computation on the values from the input layer. Expanding the vector notation in (1) we get

$$h_i = \sum_{k=1}^{V} x_k \cdot w_{ki} \tag{13}$$

Now we can take the derivative of E with regard to each element of $\mathbf{W}$, obtaining

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \mathrm{EH}_i \cdot x_k \tag{14}$$

This is equivalent to the tensor product of $\mathbf{x}$ and EH, i.e.,

$$\frac{\partial E}{\partial \mathbf{W}} = \mathbf{x} \otimes \mathrm{EH} = \mathbf{x}\mathrm{EH}^T \tag{15}$$

from which we obtain a $V \times N$ matrix. Since only one component of $\mathbf{x}$ is non-zero, only one row of $\frac{\partial E}{\partial \mathbf{W}}$ is non-zero, and the value of that row is EH^T, an N-dim vector. We obtain the update equation of $\mathbf{W}$ as

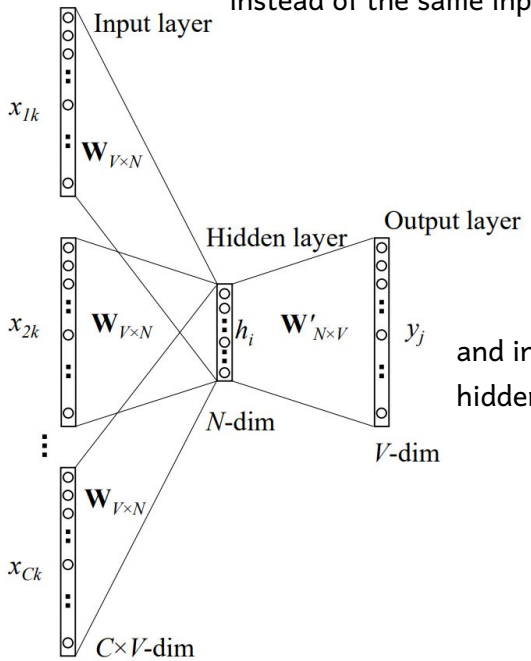
$$\mathbf{v}_{w_I}^{(\mathrm{new})} = \mathbf{v}_{w_I}^{(\mathrm{old})} - \eta \mathrm{EH}^T \tag{16}$$

Small note

As we iteratively update the model parameters by going through context-target word pairs generated from a training corpus, the effects on the vectors will accumulate. We can imagine that the output vector of a word w is "dragged" back-and-forth by the input vectors of w's co-occurring neighbors, as if there are physical strings between the vector of w and the vectors of its neighbors. Similarly, an input vector can also be considered as being dragged by many output vectors. This interpretation can remind us of gravity, or force-directed graph layout. The equilibrium length of each imaginary string is related to the strength of cooccurrence between the associated pair of words, as well as the learning rate. After many iterations, the relative positions of the input and output vectors will eventually stabilize.

Note about multi-words CBOW

In previous slides we explained one word context CBOW here we will tell the difference with multiwords.



instead of the same input we take the average of input vectors

$$\begin{aligned}\mathbf{h} &= \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1+\mathbf{x}_2+\cdots+\mathbf{x}_C)\\ &= \frac{1}{C}(\mathbf{v}_{w_1}+\mathbf{v}_{w_2}+\cdots+\mathbf{v}_{w_C})^T\end{aligned}$$

where C is the number of words in the context, $w_1,\cdots,w_C$ are the words the in the context, and $\mathbf{v}_w$ is the input vector of a word w. The loss function is

$$E = -\log p(w_O|w_{I,1},\cdots,w_{I,C}) \quad (19)$$

$$= -u_{j^*}+\log\sum_{j'=1}^{V}\exp(u_{j'}) \quad (20)$$

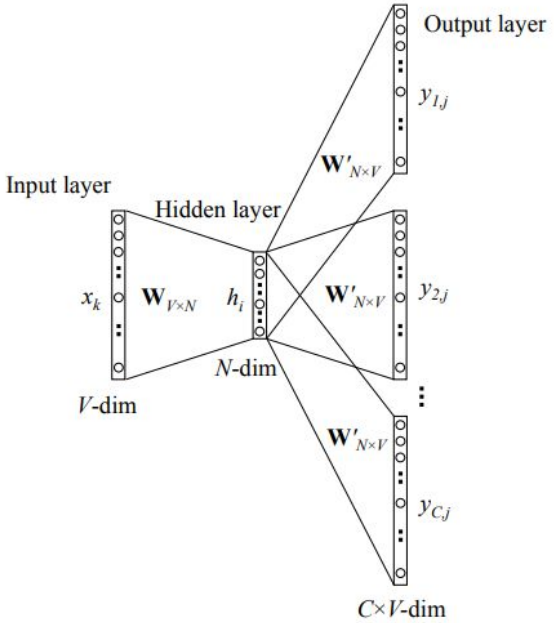
$$= -\mathbf{v}'^{T}_{w_O}\cdot\mathbf{h}+\log\sum_{j'=1}^{V}\exp(\mathbf{v}'^{T}_{w_j}\cdot\mathbf{h}) \quad (21)$$

and in the update in the input to hidden layer we do a change:

The update equation for input→hidden weights is similar to (16), except that now we need to apply the following equation for every word $w_{I,c}$ in the context:

$$\mathbf{v}^{(\text{new})}_{w_{I,c}} = \mathbf{v}^{(\text{old})}_{w_{I,c}} - \frac{1}{C}\cdot\eta\cdot\text{EH}^T \qquad \text{for } c=1,2,\cdots,C. \quad (23)$$

where $\mathbf{v}_{w_{I,c}}$ is the input vector of the c-th word in the input context; η is a positive learning rate; and $\text{EH}=\frac{\partial E}{\partial h_i}$ is given by (12). The intuitive understanding of this update equation is the same as that for (16).



Skip gram

Skip gram in notice is the opposite of the multi words model

$$\mathbf{h} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I}, \tag{24}$$

On the output layer, instead of outputing one multinomial distribution, we are outputing C multinomial distributions. Each output is computed using the same hidden→output matrix:

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V}\exp(u_{j'})} \tag{25}$$

$$\begin{aligned} E &= -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C}|w_I) && (27)\\ &= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j^*_c})}{\sum_{j'=1}^{V}\exp(u_{j'})} && (28)\\ &= -\sum_{c=1}^{C} u_{j^*_c} + C\cdot\log\sum_{j'=1}^{V}\exp(u_{j'}) && (29) \end{aligned}$$

$$\mathbf{v}^{(\text{new})}_{w_I} = \mathbf{v}^{(\text{old})}_{w_I} - \eta\cdot \text{EH}^T \tag{35}$$

where EH is an N-dim vector, each component of which is defined as

$$\text{EH}_i = \sum_{j=1}^{V}\text{EI}_j\cdot w'_{ij}. \tag{36}$$

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta\cdot\text{EI}_j\cdot\mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

Optimization

The previous method (mentioned in the previous slides) has a high complexity and it needs optimization to reduce the huge computation and there is two ways: 1. heirirecy softmax 2. negative sampling

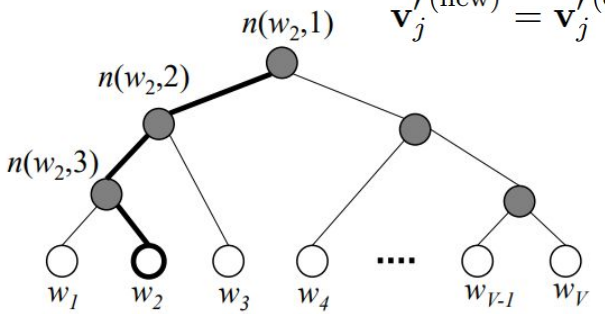
Heiriracy softmax

A binary tree model to represent the words where each edge is the probability from word to other and we use dot product to move. with the Huffman tree based hierarchical softmax requires only about log2(Unigram perplexity(V)).

In the hierarchical softmax model, there is no output vector representation for words. Instead, each of the $V-1$ inner units has an output vector $\mathbf{v}'_{n(w,j)}$. And the probability of a word being the output word is defined as

$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w, j))]\!] \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h}\right) \tag{37}$$

$$\mathbf{v}'^{(\mathrm{new})}_j = \mathbf{v}'^{(\mathrm{old})}_j - \eta\left(\sigma({\mathbf{v}'_j}^T \mathbf{h}) - t_j\right) \cdot \mathbf{h}$$



nodes. The idea behind decomposing the output layer to a binary tree was to reduce the complexity to obtain probability distribution from $O(V)$ to $O(\log(V))$. Below Figure 2 is an example of Hierarchical binary tree (figure is from [8]):

Negative sampling

A method to reduce the complexity by reducing the number of negative samples taken with the positive one (by K)

$$E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h}) \tag{55}$$

in the skip-gram model; $\mathcal{W}_{\text{neg}} = \{w_j | j = 1, \cdots, K\}$ is the set of words that are sampled based on $P_n(w)$, i.e., negative samples.

TF-IDF

word representation model that distinguish between stopwords (and words are similar to them) and the important words in the corpus

$$\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t,d)+1) \tag{6.12}$$

The **inverse document frequency** is a measure of how much information the word provides

Because of the large number of documents in many collections, this measure too is usually squashed with a log function. The resulting definition for inverse document frequency (idf) is thus

$$\mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right) \tag{6.13}$$

Sources

- word2vec Parameter Learning Explained [by Xin Rong]
- Distributed Representations of Words and Phrasesand their Compositionality [Tomas Mikolov and others]
- Efficient Estimation of Word Representations in Vector Space [Tomas Mikolov and others]
- Speech and Language Processing [Dan Jurafsky and James H. Martin]

CBOW

use the product of the input→hidden weight matrix and the average vector as the output.

$$\mathbf{h} = \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) \tag{17}$$

$$= \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C})^T \tag{18}$$

where C is the number of words in the context, $w_1, \cdots, w_C$ are the words the in the context, and $\mathbf{v}_w$ is the input vector of a word w. The loss function is

$$E = = -\log p(w_O|w_{I,1}, \cdots, w_{I,C}) \tag{19}$$

$$= -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \tag{20}$$

$$= -\mathbf{v}'_{w_O}{}^T \cdot \mathbf{h} + \log \sum_{j'=1}^{V} \exp(\mathbf{v}'_{w_j}{}^T \cdot \mathbf{h}) \tag{21}$$

The update equation for the hidden→output weights stay the same as that for the one-word-context model (11). We copy it here:

$$\mathbf{v}'_{w_j}{}^{(\text{new})} = \mathbf{v}'_{w_j}{}^{(\text{old})} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V. \tag{22}$$

Note that we need to apply this to every element of the hidden→output weight matrix for each training instance.

The update equation for input→hidden weights is similar to (16), except that now we need to apply the following equation for every word $w_{I,c}$ in the context:

$$\mathbf{v}_{w_{I,c}}^{(\text{new})} = \mathbf{v}_{w_{I,c}}^{(\text{old})} - \frac{1}{C} \cdot \eta \cdot \text{EH}^T \qquad \text{for } c = 1, 2, \cdots, C. \tag{23}$$

where $\mathbf{v}_{w_{I,c}}$ is the input vector of the c-th word in the input context; η is a positive learning rate; and $\text{EH} = \frac{\partial E}{\partial h_i}$ is given by (12). The intuitive understanding of this update equation is the same as that for (16).

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \text{EH}_i$$

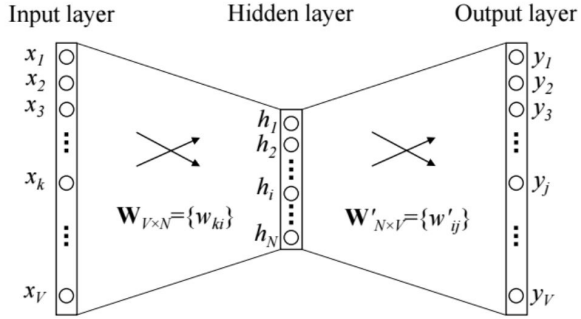


Figure 1: A simple CBOW model with only one word in the context

explain the loss in a simple example of CBOW

$$= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E, \tag{7}$$

where $E = -\log p(w_O|w_I)$ is our loss function (we want to minimize E), and j^* is the index of the actual output word in the output layer. Note that this loss function can be understood as a special case of the cross-entropy measurement between two probabilistic distributions.

$$p(w_j|w_I) = \frac{\exp\left({\mathbf{v}'_{w_j}}^T \mathbf{v}_{w_I}\right)}{\sum_{j'=1}^{V} \exp\left({\mathbf{v}'_{w_{j'}}}^T \mathbf{v}_{w_I}\right)}$$

$$CELoss = -x_i + \log \sum_{c=1}^{N} e^{x_c}$$

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V. \tag{11}$$

$$\mathbf{v}^{(\text{new})}_{w_I} = \mathbf{v}^{(\text{old})}_{w_I} - \eta \mathrm{E}\mathrm{H}^T \tag{16}$$

CBOW Multiwords

use the product of the input→hidden weight matrix and the average vector as the output.

$$\mathbf{h} = \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) \tag{17}$$

$$= \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C})^T \tag{18}$$

where C is the number of words in the context, $w_1, \cdots, w_C$ are the words the in the context, and $\mathbf{v}_w$ is the input vector of a word w. The loss function is

$$E = = -\log p(w_O|w_{I,1}, \cdots, w_{I,C}) \tag{19}$$

$$= -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \tag{20}$$

$$= -\mathbf{v}'_{w_O}{}^T \cdot \mathbf{h} + \log \sum_{j'=1}^{V} \exp(\mathbf{v}'_{w_j}{}^T \cdot \mathbf{h}) \tag{21}$$

The update equation for the hidden→output weights stay the same as that for the one-word-context model (11). We copy it here:

$$\mathbf{v}'_{w_j}{}^{(\text{new})} = \mathbf{v}'_{w_j}{}^{(\text{old})} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V. \tag{22}$$

Note that we need to apply this to every element of the hidden→output weight matrix for each training instance.

The update equation for input→hidden weights is similar to (16), except that now we need to apply the following equation for every word $w_{I,c}$ in the context:

$$\mathbf{v}_{w_{I,c}}^{(\text{new})} = \mathbf{v}_{w_{I,c}}^{(\text{old})} - \frac{1}{C} \cdot \eta \cdot \text{EH}^T \qquad \text{for } c = 1, 2, \cdots, C. \tag{23}$$

where $\mathbf{v}_{w_{I,c}}$ is the input vector of the c-th word in the input context; η is a positive learning rate; and $\text{EH} = \frac{\partial E}{\partial h_i}$ is given by (12). The intuitive understanding of this update equation is the same as that for (16).

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \text{EH}_i$$

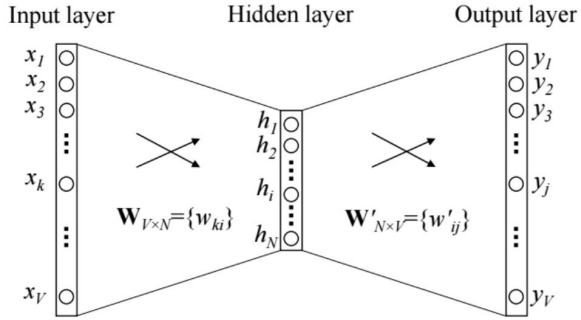


Figure 1: A simple CBOW model with only one word in the context