Deep Learning & Engineering Applications

9. Word Embedding

JIDONG J. YANG

COLLEGE OF ENGINEERING

UNIVERSITY OF GEORGIA

NLP - Language Modeling

Text data sources, e.g., books, articles, Wikipedia content, etc.

- Abundance
- Discrete

Exemplar applications:

- Autocompletion
- Next-word prediction
- Sentiment analysis
- Text tagging
- Question answering
- Machine translation
- etc.

Word to Vector (word2vec)

Words are basic unit of meaning in natural language.

Vectors are used to represent words, which can also be considered as <u>feature vectors</u> or <u>representations of words</u>.

The technique of mapping words to real vectors is called word embedding.

Different Representations

We could use one-hot vectors to represent words.

Given a vocabulary size of a dictionary = N (i.e., N unique words in the dictionary), for one-hot vector, we will use a N-length long vector to represent each word in the dictionary.

Problems with one-hot word vectors?

- Sparse and inefficient representation.
- Cannot accurately express the similarity between different words (e.g., cosine similarity) $\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \in [-1, 1]$

The word2vec was introduced to address the above issues.

- Map each word to a fixed-length vector (dense representation) to better express the similarity among different words.
- Two popular models: skip-gram and continuous bag of words (CBOW)

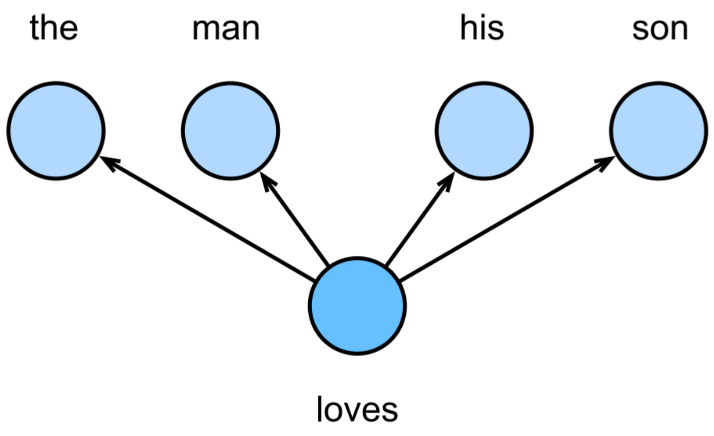
Skip-Gram and CBOW

Both are self-supervised models.

Predict some words using their surrounding words in corpora.

Use conditional probabilities for semantically meaningful representations.

The Skip-Gram Model



Assume a word can be used to generate its surrounding words in a text sequence.

The man loves his son.

Let us choose “loves” as the center word and set the context window size to 2. The conditional probability for generating the *context words*:

$$P(\text{"the"}, \text{"man"}, \text{"his"}, \text{"son"} \mid \text{"loves"})$$

Assume that the context words are independently generated given the center word, the conditional probability can be calculated as:

$$P(\text{"the"} \mid \text{"loves"}) \cdot P(\text{"man"} \mid \text{"loves"}) \cdot P(\text{"his"} \mid \text{"loves"}) \cdot P(\text{"son"} \mid \text{"loves"})$$

The Skip-Gram Model

Each word is represented as a d-dimension vector (R^d).

Assume a center word is represented as $v_c \in R^d$ and a context word is represented as $u_o \in R^d$, we can write the probability of observing the context word (w_o) given the center word (w_c) as:

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

where, $\mathcal{V} = \{0, 1, \ldots, |\mathcal{V}|-1\}$ is the vocabulary index set.

Given a context window size of *m*, the likelihood function becomes the joint probability of generating all the context words given any center word:

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} P(w^{(t+j)} \mid w^{(t)})$$

How to train a Skip-Gram model?

Maximize the likelihood function:

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} P(w^{(t+j)} \mid w^{(t)})$$

Minimize the following loss function:

$$-\sum_{t=1}^{T} \sum_{-m \leq j \leq m,\ j \neq 0} \log P(w^{(t+j)} \mid w^{(t)})$$

Skip-Gram Model – Computing gradients

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

$$\log P(w_o \mid w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log\left(\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)\right)$$

$$\frac{\partial \log P(w_o \mid w_c)}{\partial \mathbf{v}_c} = \mathbf{u}_o - \frac{\sum_{j\in\mathcal{V}} \exp(\mathbf{u}_j^\top \mathbf{v}_c)\mathbf{u}_j}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

$$= \mathbf{u}_o - \sum_{j\in\mathcal{V}} \left(\frac{\exp(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}\right)\mathbf{u}_j$$

$$= \mathbf{u}_o - \sum_{j\in\mathcal{V}} P(w_j \mid w_c)\mathbf{u}_j$$

After training, we obtain $\boldsymbol{v}_i$ (as the center word) and $\boldsymbol{u}_i$ (as the context word) for all words in the dictionary (i.e., two embedding matrices).

In NLP applications, the center word vectors are typically used as representations.

The Continuous Bag of Words (CBOW) Model

The conditional probability of generating any center word w_c given its surrounding context words $w_{o1}, \ldots, w_{o2m}$ can be modeled by:

$$P(w_c \mid w_{o_1}, \ldots, w_{o_{2m}}) = \frac{\exp\left(\frac{1}{2m}\mathbf{u}_c^\top(\mathbf{v}_{o_1} + \ldots, + \mathbf{v}_{o_{2m}})\right)}{\sum_{i \in \mathcal{V}} \exp\left(\frac{1}{2m}\mathbf{u}_i^\top(\mathbf{v}_{o_1} + \ldots, + \mathbf{v}_{o_{2m}})\right)}$$

$$\mathcal{W}_o = \{w_{o_1}, \ldots, w_{o_{2m}}\} \qquad \bar{\mathbf{v}}_o = \frac{(\mathbf{v}_{o_1} +, \ldots, + \mathbf{v}_{o_{2m}})}{2m}$$

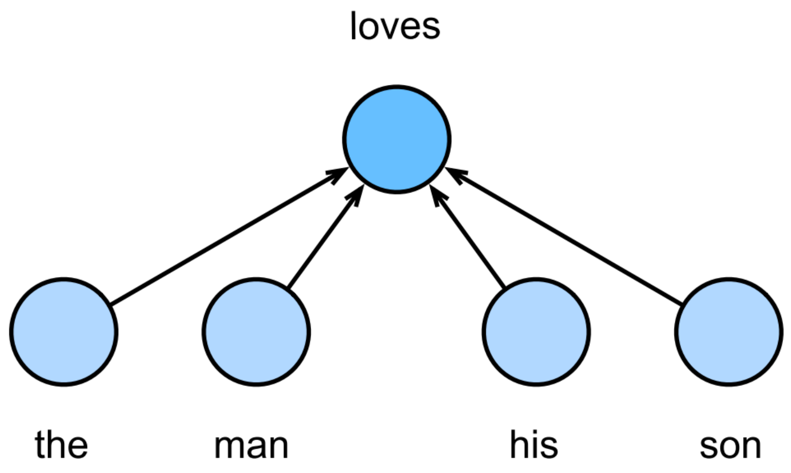
$$P(w_c \mid \mathcal{W}_o) = \frac{\exp\left(\mathbf{u}_c^\top \bar{\mathbf{v}}_o\right)}{\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^\top \bar{\mathbf{v}}_o\right)}$$

Given a text sequence of length *T* and a context window size of m, The likelihood function of the CBOW model is:

$$\prod_{t=1}^{T} P(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$$

The man loves his son.

$$P(\text{"loves"} \mid \text{"the"}, \text{"man"}, \text{"his"}, \text{"son"})$$



CBOW Model - Computing gradients

$$P(w_c \mid \mathcal{W}_o) = \frac{\exp\left(\mathbf{u}_c^\top \bar{\mathbf{v}}_o\right)}{\sum_{i\in\mathcal{V}} \exp\left(\mathbf{u}_i^\top \bar{\mathbf{v}}_o\right)}$$

$$\log P(w_c \mid \mathcal{W}_o) = \mathbf{u}_c^\top \bar{\mathbf{v}}_o - \log\left(\sum_{i\in\mathcal{V}} \exp\left(\mathbf{u}_i^\top \bar{\mathbf{v}}_o\right)\right)$$

$$\frac{\partial \log P(w_c \mid \mathcal{W}_o)}{\partial \mathbf{v}_{o_i}} = \frac{1}{2m}\left(\mathbf{u}_c - \sum_{j\in\mathcal{V}} \frac{\exp(\mathbf{u}_j^\top \bar{\mathbf{v}}_o)\mathbf{u}_j}{\sum_{i\in\mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)}\right)$$

$$= \frac{1}{2m}\left(\mathbf{u}_c - \sum_{j\in\mathcal{V}} P(w_j \mid \mathcal{W}_o)\mathbf{u}_j\right)$$

Problems with training

Recall that the previous formulations use softmax to calculate the conditional probability.

Skip-Gram: $$\frac{\partial \log P(w_o \mid w_c)}{\partial \mathbf{v}_c} = \mathbf{u}_o - \sum_{j \in \mathcal{V}} \left(\frac{\exp(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \right) \mathbf{u}_j$$

CBOW: $$\frac{\partial \log P(w_c \mid \mathcal{W}_o)}{\partial \mathbf{v}_{o_i}} = \frac{1}{2m} \left(\mathbf{u}_c - \sum_{j \in \mathcal{V}} \frac{\exp(\mathbf{u}_j^\top \bar{\mathbf{v}}_o)\mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)} \right)$$

The computation of the gradients that sum over a large dictionary (often with millions of words) is costly.

Negative sampling

Modify the original objective function.

Define an event that the context word w_o comes from context window of the central word w_c, the probability of this event can be modeled as:

$$P(D = 1 \mid w_c, w_o) = \sigma(\mathbf{u}_o^\top \mathbf{v}_c) \qquad \sigma(x) = \frac{1}{1 + \exp(-x)}$$

What if we maximize the joint probability of the events for a given text sequence of length T?

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} P(D = 1 \mid w^{(t)}, w^{(t+j)})$$

Permit negative events

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} P(w^{(t+j)} \mid w^{(t)})$$

Negative sampling

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\ j \neq 0} P(w^{(t+j)} \mid w^{(t)})$$

The gradient computation in each step of the training is no longer related to the dictionary size, but linearly related to K.

Approximate by $P(w^{(t+j)} \mid w^{(t)}) = P(D=1 \mid w^{(t)}, w^{(t+j)}) \prod_{k=1,\ w_k \sim P(w)}^{K} P(D=0 \mid w^{(t)}, w_k)$

$$-\log P(w^{(t+j)} \mid w^{(t)}) = -\log P(D=1 \mid w^{(t)}, w^{(t+j)}) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log P(D=0 \mid w^{(t)}, w_k)$$

$$= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^\top \mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log\left(1 - \sigma\left(\mathbf{u}_{h_k}^\top \mathbf{v}_{i_t}\right)\right)$$

$$= -\log \sigma\left(\mathbf{u}_{i_{t+j}}^\top \mathbf{v}_{i_t}\right) - \sum_{k=1,\ w_k \sim P(w)}^{K} \log \sigma\left(-\mathbf{u}_{h_k}^\top \mathbf{v}_{i_t}\right)$$

A nice tutorial: https://jalammar.github.io/illustrated-word2vec/

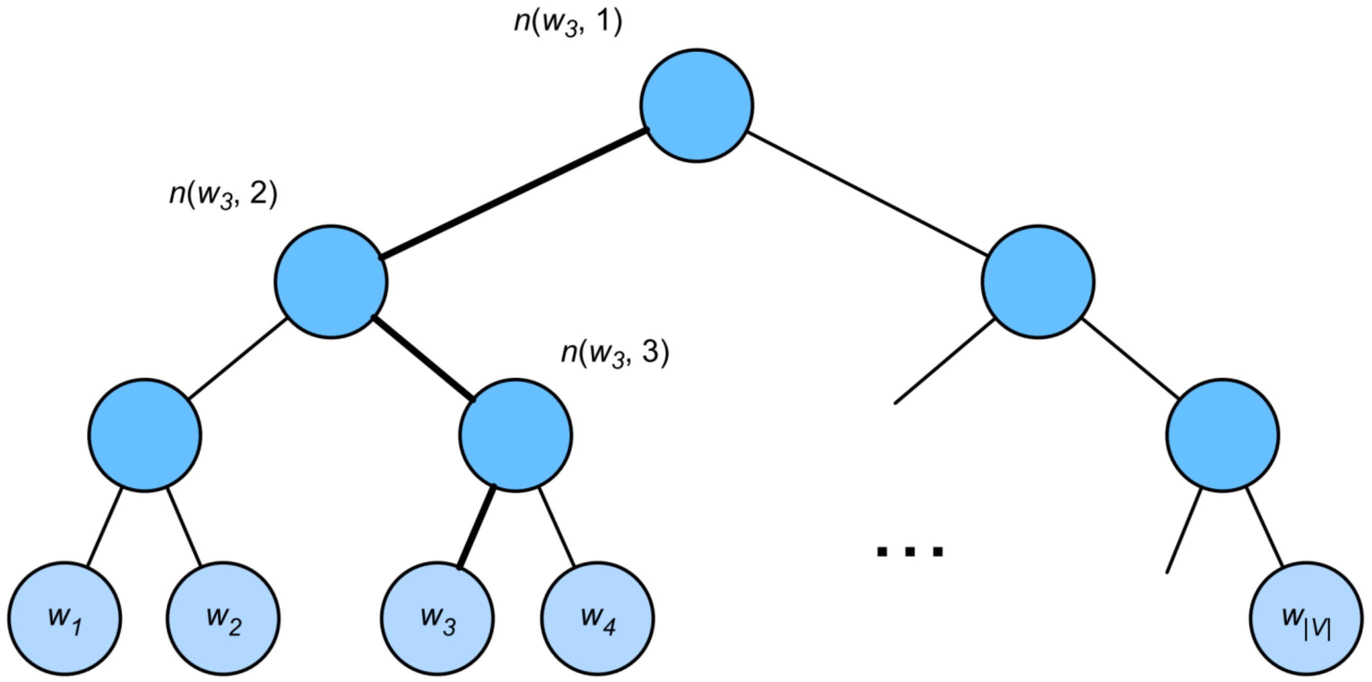
Hierarchical Softmax (H-softmax)

Use the binary tree.

Each leaf node of the tree represents a word in the dictionary.

The gradient computational overhead for each step in the training process is related to the logarithm of the dictionary size.

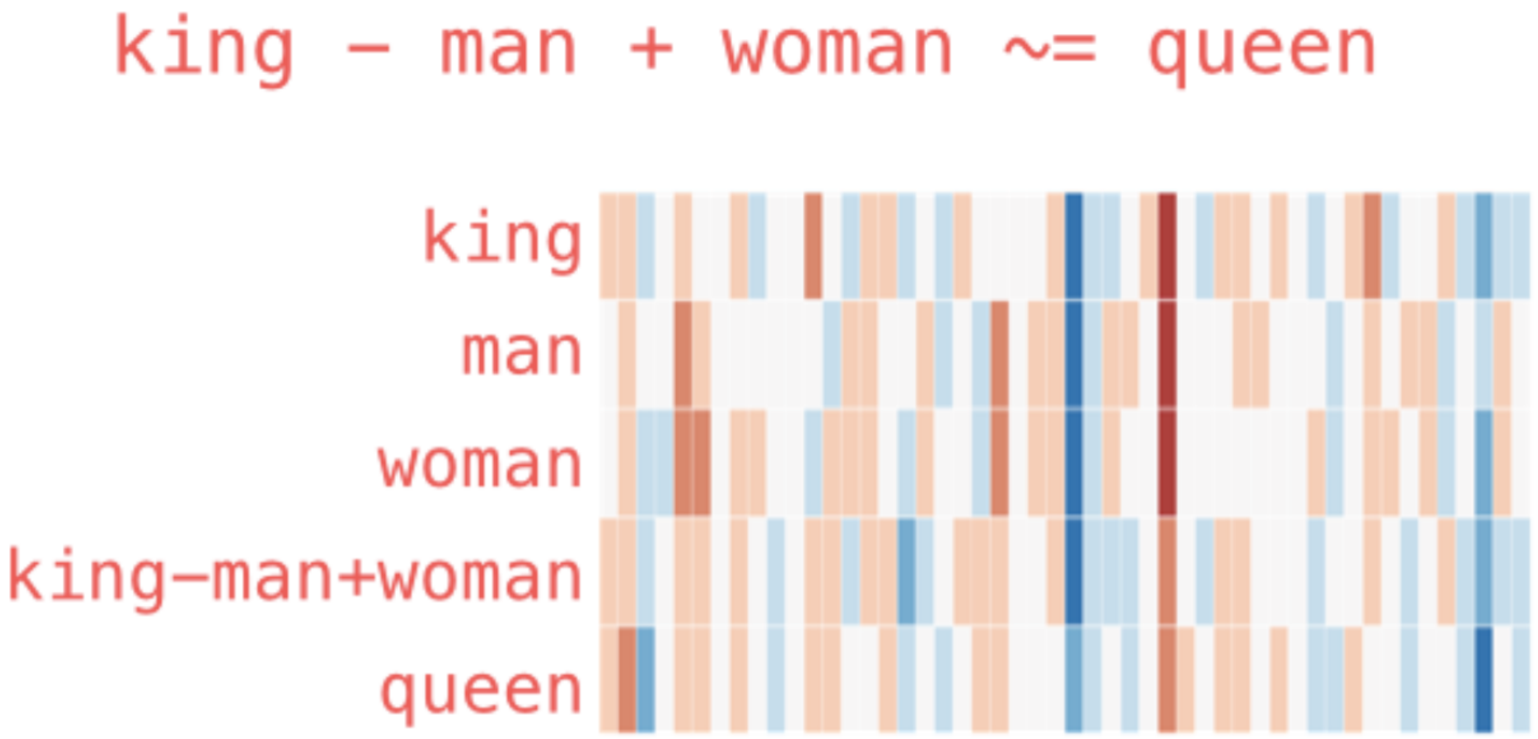
$Path\ length\ = log_2|V|$



$$P(w_3 \mid w_c) = \sigma(\mathbf{u}_{n(w_3,1)}^\top \mathbf{v}_c) \cdot \sigma(-\mathbf{u}_{n(w_3,2)}^\top \mathbf{v}_c) \cdot \sigma(\mathbf{u}_{n(w_3,3)}^\top \mathbf{v}_c)$$

$$\sum_{w \in \mathcal{V}} P(w \mid w_c) = 1$$

Interpretation with vector representation



https://jalammar.github.io/illustrated-word2vec/

Global Vectors (GloVe) model

Explicitly consider the global corpus statistics.

Let *X* be the matrix of word-word co-occurrence counts, where the entries X_{ij} tabulate the number of times that word *j* occurs in the context of word *i*.

Let $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of word *i*.

Let $p_{ij} = p(j|i) = \frac{X_{ij}}{X_i}$ be the probability that word *j* appear in the context of word *i*.

$$f(\mathbf{u}_j, \mathbf{u}_k, \mathbf{v}_i) \approx \boxed{\frac{p_{ij}}{p_{ik}}}$$

Extracted from the corpus

$w_k =$	solid	gas	water	fashion
$p_1 = P(w_k \mid \text{ice})$	0.00019	0.000066	0.003	0.000017
$p_2 = P(w_k \mid \text{steam})$	0.000022	0.00078	0.0022	0.000018
p_1/p_2	8.9	0.085	1.36	0.96

[Pennington et al. 2014] https://nlp.stanford.edu/pubs/glove.pdf

Function *f*()

$$f(\mathbf{u}_j, \mathbf{u}_k, \mathbf{v}_i) \approx \frac{p_{ij}}{p_{ik}}$$

Since vector spaces are inherently linear structures, the most natural way to do this is with vector differences.

$$f\left((\boldsymbol{u}_j - \boldsymbol{u}_k)^T \boldsymbol{v}_i\right) \approx \frac{p_{ij}}{p_{ik}}$$

If we exchange index j with k, the function f should satisfy:

$$f\left((\boldsymbol{u}_k - \boldsymbol{u}_j)^T \boldsymbol{v}_i\right) \approx \frac{p_{ik}}{p_{ij}}$$

$$f(x)f(-x) = 1$$

One choice of *f()* is *exp()*.

Restoring Symmetry

$$f\left(\left(\boldsymbol{u}_j - \boldsymbol{u}_k\right)^T \boldsymbol{v}_i\right) \approx \frac{p_{ij}}{p_{ik}}$$

$$exp\left(\left(\boldsymbol{u}_j - \boldsymbol{u}_k\right)^T \boldsymbol{v}_i\right) \approx \frac{p_{ij}}{p_{ik}}$$

$$\frac{exp(\boldsymbol{u}_j^T \boldsymbol{v}_i)}{exp(\boldsymbol{u}_k^T \boldsymbol{v}_i)} \approx \frac{p_{ij}}{p_{ik}}$$

$$\alpha\, exp\left(\boldsymbol{u}_j^T \boldsymbol{v}_i\right) \approx p_{ij} = \frac{X_{ij}}{X_i}$$

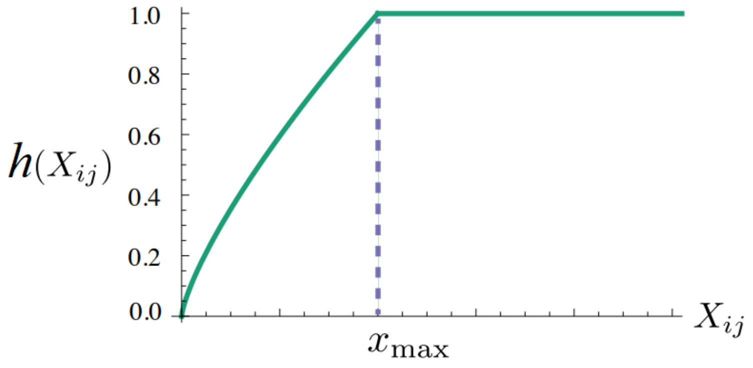
$$\log(\alpha) + \boldsymbol{u}_j^T \boldsymbol{v}_i \approx \log\left(X_{ij}\right) - \log(X_i)$$

$$\begin{tikzcd} \boldsymbol{u}_j^T \boldsymbol{v}_i + \log(\alpha) + \log(X_i) \approx \log\left(X_{ij}\right) \arrow[d, "c_j = \log(\alpha)"', "b_i = \log(X_i)"] \\ \boldsymbol{u}_j^T \boldsymbol{v}_i + c_j + b_i \approx \log\left(X_{ij}\right) \end{tikzcd}$$

Weighted Least Squares Regression Problem

The goal of GloVe is to minimize the following loss function:

$$\sum_{i\in V}\sum_{j\in V} h(X_{ij})\Big(\underbrace{\boldsymbol{u}_j^T \boldsymbol{v}_i + c_j}_{\log(p_{ij})} + \underbrace{b_i - \log(X_{ij})}_{-\log\left(\frac{X_{ij}}{X_i}\right)}\Big)^2$$



$$h(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

Summary

Use vectors (feature vectors) to represent words, referred to as word embedding. The semantics (similarity between words) are learned by neighboring words through self-supervised learning.

The Skip-Gram model assumes that a word can be used to generate its surrounding words in a text sequence.

The CBOW model assumes that a center word is generated based on its surrounding context words.

GloVe considers the global corpus statistics and can be interpreted from the ratio of word-word co-occurrence probabilities. The center word vector and the context word vector are mathematically equivalent for any word in GloVe.

GloVe uses squared loss to fit precomputed global corpus statistics.

The techniques are not limited to language modeling and have broader applications for embeddings in general.