Distributed representations

ML4SE

Denis Litvinov

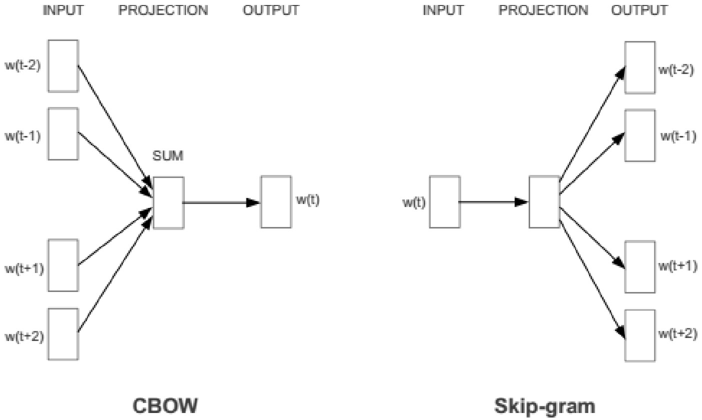
October 26, 2021

Table of Contents

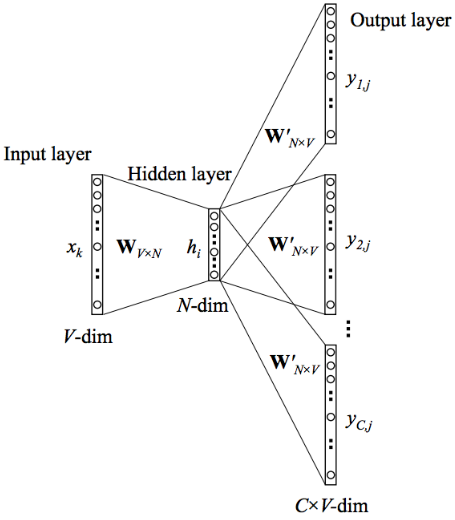
Problem Statement

Given a sequence of tokens (words), build a vector in R^N for each token, which are in some sense representative.

Word2Vec Model



Skip-gram Model



where N - desired embedding dimension

Skip-gram Model

For each word t predict surrounding words in a window of size m (context)

Objective is to maximize probability of context words given the current center word:

$$J(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m; j \neq 0} p(x_{t+j}|x_t;\theta) \to \max_{\theta}$$

, or

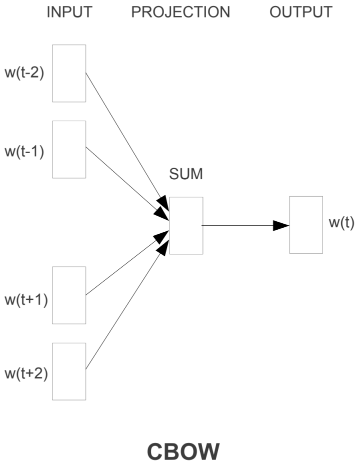
$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m; j \neq 0} \log p(x_{t+j}|x_t;\theta) \to \min_{\theta}$$

where x_t - center word,
x_{t+j} - word from context,
m - context size.

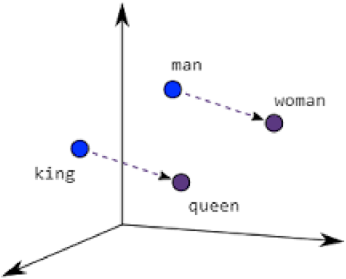
$$p(x_{t+j}|x_t) = p(out|center) = \frac{e^{u_{out}^T v_{center}}}{\sum_{i=1}^{V} e^{u_i^T v_{center}}}$$

Continuous Bag of Words Model

= Predict center word from surrounding context

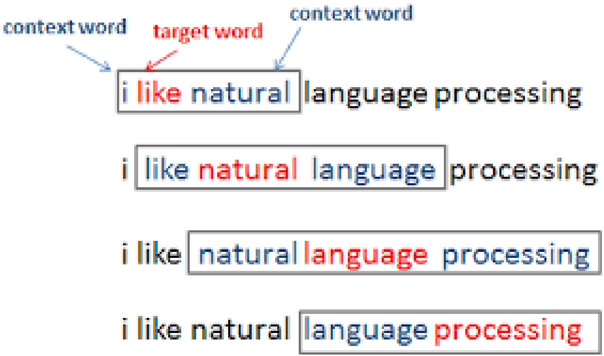


Why Embeddings?



- What are other ways to construct a vector in R^N for each word?
- Embeddings allow to apply simple algebra on words
- Embeddings can describe entities (words, documents) that are absent in the dataset.

Distributional hypothesis



Word embedding is defined by it's context.

Problem statement

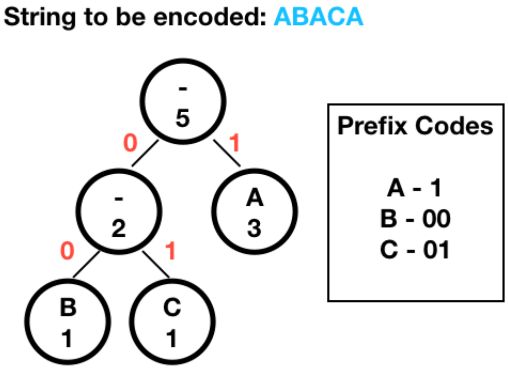
What computational problems do you see in the objective function?

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m; j\neq 0} \log p(x_{t+j}|x_t;\theta) \to \min_{\theta}$$

$$p(x_{t+j}|x_t) = p(out|center) = \frac{e^{u_{out}^T v_{center}}}{\sum_{i=1}^{V} e^{u_i^T v_{center}}}$$

Huffman tree

How to build a binary prefix tree?



Huffman tree

Complexity $O(V) \to O(\log_2 V)$
$x = v_{n(x,j)}^T v_x$,
where $n(x, j)$ is the j-th node on the path from the root to token x.
$p(n, left) = \sigma(v_n^T v_x)$ - probability to go left.
$p(n, right) = \sigma(-v_n^T v_x)$ - probability to go right.
Then,
$p(x_j|x) = \prod_{j=1}^{L(x)-1} \sigma(I[n(x, j+1) == child(n(x, j))] v_n^T v_x)$,
where $L(x)$ - depth of the tree,
$child(x)$ - child of node n.

Huffman tree

Using negative sampling with k samples:

$$\log p(x_{t+j}|x_t;\theta) = \log\sigma(u_{outer}^T v_{center}) + \sum_{i=1}^{k} E_{j\sim P(x)}[\log\sigma(-u_j^T v_{center})]$$

Co-occurrence matrix

= Word embeddings through decomposition of co-occurrence matrix

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

The resulting counts matrix will then be:

$$X = \begin{array}{c|cccccccc} & I & like & enjoy & deep & learning & NLP & flying & . \\ \hline I & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ like & 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ enjoy & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ deep & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ learning & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ NLP & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ flying & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ . & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}$$

Singular Value Decomposition

Every matrix $M \in R^{nxm}$, $n < m$ can be represented as a product

$$M = U \Sigma V^T$$

where $U \in R^{nxn}$, $V \in R^{nxm}$ are orthogonal matrices,
$\Sigma \in R^{nxn}$ - diagonal matrix

$$Mv = \sigma u$$

$$M^* u = \sigma v$$

SVD complexity $O(nm^2)$

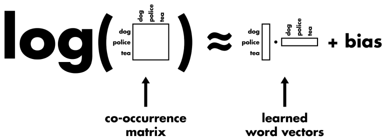
Glove

P_{ij} - occurrence of i-th word along with j-th in the window of size m
Cons:

1. Very high-dimensional, not used in practice
2. Hard to add new words and docs

Trivial solution: use some dimension-reduction method, usually SVD

Glove



$\text{J}(\theta) = \frac{1}{2}\sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})$

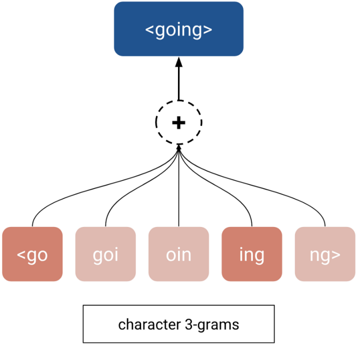
$f(x)$ - some weight functions that obeys following properties:

- $f(0) = 0$
- non-decreasing, so rare co-occurrences won't overweight
- relatively small for large x, to compensate frequent co-occurrences

The authors have chosen

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & x \leq x_{max} \\ 1 & else \end{cases}$$

FastText



FastText

Subword embeddings.
Introduce scoring function (instead of scalar product as in w2v):

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c$$

where G_w - set of 3-grams appearing in word w
z_g - embedding of 3-gram g
v_c - context vector

FastText

Objective function for skip-gram case:

$$\sum_{t=1}^{T}[\sum_{c\in C_t} log(1+e^{-s(w_t,w_c)}) + \sum_{n\in N_{t,c}} \log(1+e^{s(w_t,n)})] \to \min$$

where c - chosen context position
C_t set of context position dependent on current word t
T - total number of words
$N_{t,c}$ - set of negative samples dependent on chosen word and context
Inference: Embedding of word w from 3-grams G_w:

$$v_w = \sum_{g\in G_w} z_g$$

FastText

Tweaks in Negative sampling: sampling with probability

$$p(w) = \frac{\sqrt{U(w)}}{Z}$$

where $Z = \sum_w \sqrt{U(w)}$
and $U(w)$ - the count of a particular word w
Probability of token w to be discarded during training:

$$P(w) = \sqrt{\frac{t}{f(w)}} + \frac{t}{f(w)}$$

where $f(w) = \frac{U(w)}{Z}$ - frequency of token w

Problem Statement

Topic modeling is equivalent to soft clustering:
Given document d_j, assign each document a vector of probabilities for each topic (cluster) $v_j = [p(t_0), ..., p(t_K)]$, $\sum_k p(t_k) = 1$
Usually,

- number of topics K is fixed and is a subject for cross-validation.
- document is described by term-document matrix (bag of words model).

How to overcome BoW assumption?

Latent Semantic Analysis

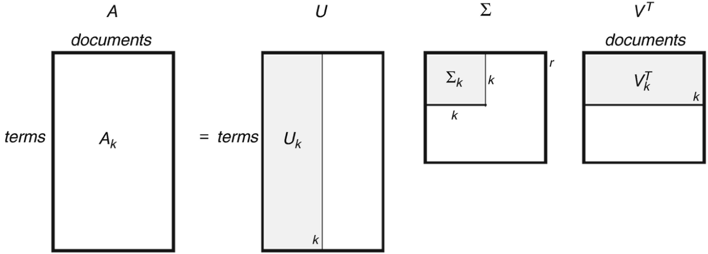
Let $X \in R^{VxD}$ be word-document matrix
where D - number of documents
V - number of words
then applying SVD we get

$$X = U\Sigma V^T$$

Select k largest singular values and corresponding singular vectors$\rightarrow$ make truncated SVD

$$X \sim U_k \Sigma_k V_k^T$$

Latent Semantic Analysis



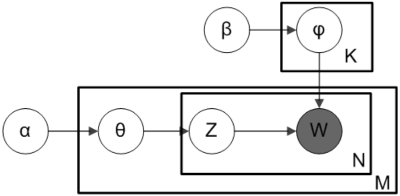
then $\Sigma_k^{\frac{1}{2}} V_k$ - document embeddings

$U_k \Sigma_k^{\frac{1}{2}}$ - learned word embeddings

- What are pros and cons of this method?

Latent Dirichlet Allocation

K - number of topics
N - number of words
M - number of documents



Assumed generative process:

1. sample $\theta_d \sim Dir(\theta|\alpha)$, $d \in 1..M$
2. sample $\phi_k \sim Dir(\phi|\beta)$, $k \in 1..K$
3. for each word w_{ij} in document i:
   1. sample a topic $z_{ij} \sim Multinomial(\theta_i)$
   2. sample a word $w_{ij} \sim Multinomial(\phi_{z_{ij}})$

Latent Dirichlet Allocation

$$p(z_{d,n} == k|\theta_d) = (\theta_d)_k$$

$$p(w_{d,n} = \nu|z_{d,n}, \phi) = (\phi_{z_{d,n}})_\nu$$

Joint distribution on word and documents

$$p(w, z, \theta, \phi|\alpha, \beta) = \prod_k Dir(\phi_k|\beta) \prod_d [Dir(\theta_d|\alpha) \prod_n p(z_{d,n}|\theta_d) p(w_{d,n}|z_{d,n}, \phi)]$$

- Essentially, Dirichlet prior works as a regularizer
- Obviously, you can introduce your own regularizers

Latent Dirichlet Allocation

Multinomial distribution

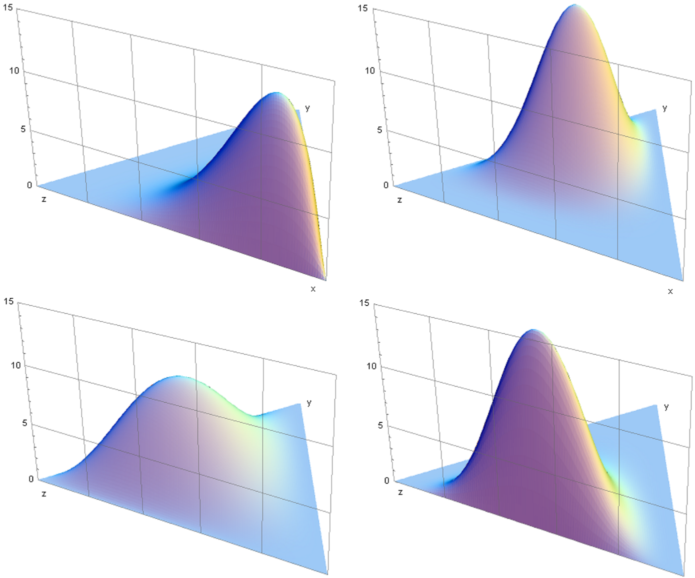
$$p(x_1, ..., x_k) = \frac{n!}{x_1!...x_k!} p_1^{x_1}...p_k^{x_k}$$

, where $\sum_{i=1}^{k} x_i = n$
Dirichlet distribution on simplex

$$Dir(z|\alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{k} z_i^{\alpha_i - 1}$$

, where $\alpha_i > 0$
Simplex is a set of points $\{z | \sum_{i=1}^{k} z_i = 1 \wedge z_i \geq 0\}$

Latent Dirichlet Allocation



Topic Coherence

Topic coherence is a measure of topic quality.
Algorithm sketch:

1. Each topic is described by top-n most probable words
2. Introduce similarity measure between words: for example, based on co-occurrence matrix or cosine distance of word embeedings
3. Compute average of pairwise similarities of top-n words for each topic
4. Average scores over topics

Problem Statement

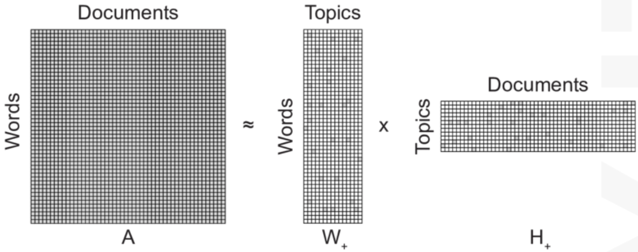
Given feature matrix $X \in R^{NxD_1}$ create a mapping $\phi : R^{NxD_1} \rightarrow R^{NxD_2}$, $D_1 > D_2$ such that most of important information about features is preserved.

- Make your own

Non-negative Matrix Factorization

$$A \sim WH$$

$$\begin{cases} ||A - WH||_F^2 \to \min_{W,H} \\ W, H > 0 \end{cases}$$



Principal Component Analysis

sample mean

$$m = \frac{1}{N}\sum_{i=1}^{N} x_i$$

sample covariance

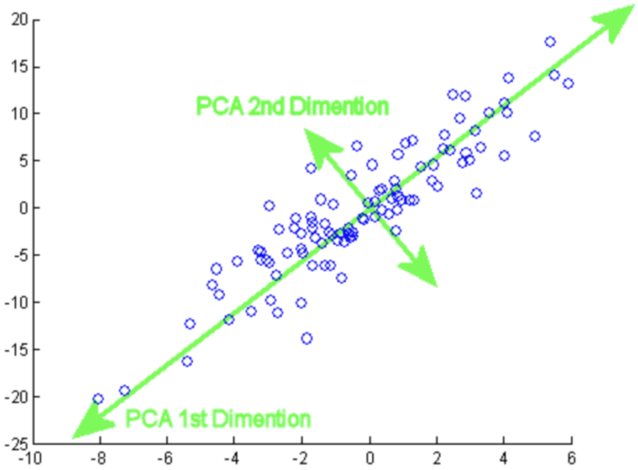
$$S = \frac{1}{N-1}(X-m)(X-m)^T$$

Eigenvalue decomposition = SVD for square matrices

$$S = U\Sigma U^T$$

Select eigenvectors corresponding to biggest k eigenvalues (principal components)
Then $X = U_k\Sigma_k$ is new feature matrix

Principal Component Analysis



Problem statement

Given feature matrix $X \in R^{NxD_1}$ create a mapping $\phi : R^{NxD_1} \to R^{NxD_2}$, $D_1 > D_2$ that preserves local structure (distances?).

t-distributed Stochastic Neighborhood Embeddings

$$p_{j|i} = \frac{exp(-\frac{1}{2\sigma_i^2}||x_i - x_j||^2)}{\sum_{k \neq l} exp(-\frac{1}{2\sigma_i^2}||x_l - x_k||^2)}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

$$Loss = KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-distributed Stochastic Neighborhood Embeddings

Student t-distribution

$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma\frac{\nu}{2}}(1 + \frac{t^2}{\nu})^{-\frac{\nu+1}{2}}$$

for $\nu = 1$

$$f_{\nu=1}(t) = \frac{1}{\pi}(1 + t^2)^{-1}$$

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k\neq i}(1 + ||y_k - y_l||^2)^{-1}}$$