DISTRIBUTED WORD REPRESENTATIONS

Natural Language Processing

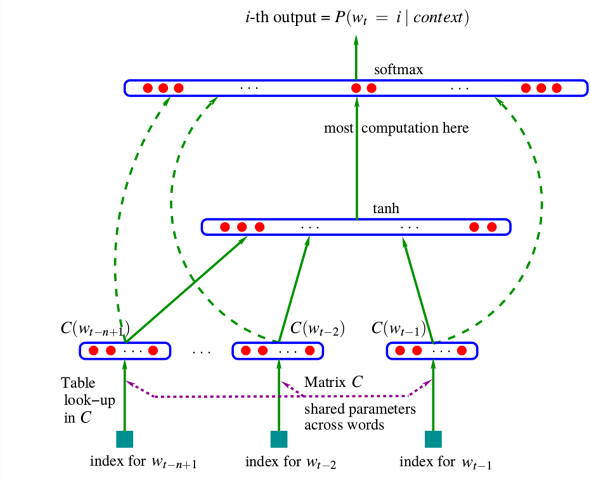
Sergey Nikolenko

Harbour Space University, Barcelona, Spain
January 12, 2018

WORD EMBEDDINGS

- Distributional hypothesis in linguistics: words with similar meaning will occur in similar contexts.
- *Distributed word representations* map words to a Euclidean space (usually of dimension several hundred):
    - started in earnest in (Bengio et al. 2003; 2006), although there were earlier ideas;
    - *word2vec* (Mikolov et al. 2013): train weights that serve best for simple prediction tasks between a word and its context: continuous bag-of-words (CBOW) and skip-gram;
    - *Glove* (Pennington et al. 2014): train word weights to decompose the (log) cooccurrence matrix.

- Bengio et al.:
  - each word $i \in V$ corresponds to a feature vector $\mathbf{w}_i \in \mathbb{R}^d$ (word embedding);



- Bengio et al.:
  - probability of the event that i occurs in a local context $c_1, \dots, c_n$ is a function of these features:

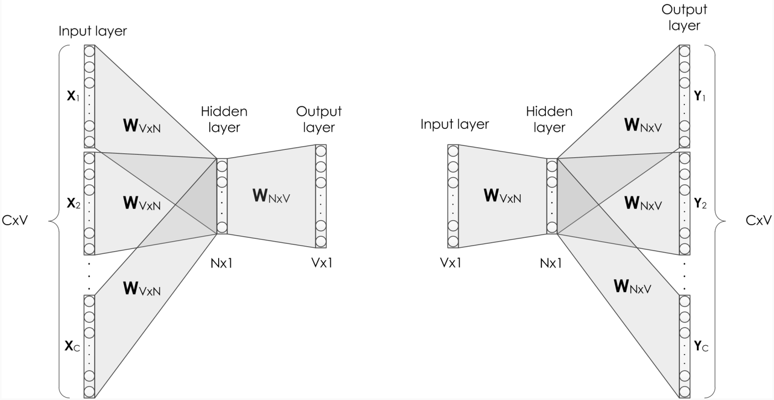
$$\hat{p}(i|c_1, \dots, c_n) = f(\mathbf{w}_i, \mathbf{w}_{c_1}, \dots, \mathbf{w}_{c_n}; \theta),$$

    where $\mathbf{w}_{c_1}, \dots, \mathbf{w}_{c_n}$ are vectors of context words and f is a function with parameters θ;
  - now we can simply train both θ and $\mathbf{w}$ at the same time, maximizing the joint likelihood

$$L(W, \theta) = \frac{1}{T} \sum_t \log f(\mathbf{w}_t, \mathbf{w}_{t-1}, \dots, \mathbf{w}_{t-n+1}; \theta) + R(W, \theta),$$

    where t spans context windows and $R(W, \theta)$ is a regularizer.
- This was quite slow and did not work too well… but actually word2vec is pretty much based on the same idea.

- So what does word2vec do? Two main architectures.
- Difference between skip-gram and CBOW architectures:
  - CBOW model predicts a word from its local context;
  - skip-gram model predicts context words from the current word.



- The CBOW *word2vec* model operates as follows:
  - inputs are one-hot word representations of dimension V;
  - the hidden layer is the matrix of vector embeddings W;
  - the hidden layer's output is the average of input vectors;
  - as output we get an estimate u_j for each word, and the posterior is a simple $\mathrm{softmax}$:

$$\hat{p}(i|c_1, \dots, c_n) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})};$$

  - thus, the loss function on a local window is to make the posterior distribution as close as possible to the data distribution:

$$L = -\log p(i|c_1, \dots, c_n) = -u_j + \log \sum_{j'=1}^{|V|} \exp(u_{j'}).$$

- In skip-gram, it's the opposite:
  - we predict each context word from the central word;
  - so now there are several multinomial distributions, one $\mathrm{softmax}$ for each context word:

$$\hat{p}(c_k|i) = \frac{\exp(u_{kc_k})}{\sum_{j'=1}^{V} \exp(u_{j'})}.$$

- How do we train a model like that?
- E.g., in skip-gram we choose θ to maximize

$$L(\theta) = \prod_{i \in D} \left(\prod_{c \in C(i)} p(c \mid i; \theta) \right) = \prod_{(i,c) \in D} p(c \mid i; \theta),$$

and we parameterize

$$p(c \mid i; \theta) = \frac{\exp(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i)}{\sum_{c'} \exp(\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i)},$$

where $\tilde{\mathbf{w}}_c$ is the context vector for word c; different from $\mathbf{w}_i$!

- This leads to the total likelihood

$$\arg\max_{\theta} \prod_{(i,c)\in D} p(c \mid i;\theta) = \arg\max_{\theta} \sum_{(i,c)\in D} p(c \mid i;\theta) =$$

$$= \arg\max_{\theta} \sum_{(i,c)\in D} \left(\exp(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i) - \log \sum_{c'} \exp(\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i) \right).$$

- How do we maximize this? It's a huge sum...

- *Negative sampling*: instead of $\sum_{c'} \exp(\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i)$ we sample a few elements and compute $\sum_{c' \in D'} \exp(\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i)$. Why does this work?
- Consider a pair (i, c) of word i and context c; we want to maximize $p((i,c) \in D; \theta)$, parameterized by θ.
- There are lots of pairs like this:

$$\arg\max_\theta \prod_{(i,c)\in D} p((i,c) \in D; \theta) = \arg\max_\theta \sum_{(i,c)\in D} \log p((i,c) \in D; \theta).$$

- Let's parameterize $p((i, c) \in D; \theta)$ via softmax, i.e., via logistic sigmoid $\sigma(x) = \frac{1}{1+\exp(-x)}$:

$$p((i, c) \in D; \theta) = \frac{1}{1 + \exp\left(-\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right)}.$$

- We maximize the overall log-likelihood:

$$\arg\max_\theta \sum_{(i,c)\in D} \log p((i, c) \in D; \theta) = \arg\max_\theta \sum_{(i,c)\in D} \log \frac{1}{1 + \exp\left(-\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right)}.$$

- How do we solve this?..

- Let's parameterize $p((i,c) \in D; \theta)$ via $\mathrm{softmax}$, i.e., via logistic sigmoid $\sigma(x) = \frac{1}{1+\exp(-x)}$:

$$p((i,c) \in D; \theta) = \frac{1}{1 + \exp\left(-\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right)}.$$

- We maximize the overall log-likelihood:

$$\arg\max_\theta \sum_{(i,c)\in D} \log p((i,c) \in D; \theta) = \arg\max_\theta \sum_{(i,c)\in D} \log \frac{1}{1 + \exp\left(-\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right)}.$$

- How do we solve this?..
- ...easy: just set all $\tilde{\mathbf{w}}_c$ and $\mathbf{w}_i$ equal to each other and very large, then $\tilde{\mathbf{w}}_c^\top \mathbf{w}_i$ will be large. :)
- What's wrong with it?

- We have basically binary classification with only positive examples and no negative ones!
- This is what negative sampling does: provide a set D' of negative examples. The maximal likelihood problem becomes

$$\arg\max_{\theta} \prod_{(i,c)\in D} p((i,c) \in D; \theta) \prod_{(i',c')\in D'} p((i',c') \notin D; \theta).$$

- Transforming:

$$\arg\max_{\theta} \prod_{(i,c)\in D} p((i,c)\in D;\theta) \prod_{(i',c')\in D'} (1 - p((i',c')\in D;\theta)) =$$

$$= \arg\max_{\theta} \left[\sum_{(i,c)\in D} \log p((i,c)\in D;\theta) + \sum_{(i',c')\in D'} \log\left(1 - p((i',c')\in D;\theta)\right) \right] =$$

$$= \arg\max_{\theta} \sum_{(i,c)\in D} \left[\log \frac{1}{1+\exp\left(-\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right)} + \sum_{(i,c')\in D'} \log \frac{1}{1+\exp\left(\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i\right)} \right] =$$

$$= \arg\max_{\theta} \sum_{(i,c)\in D} \left[\log\sigma\left(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right) + \sum_{(i,c')\in D'} \log\sigma\left(-\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i\right) \right].$$

- This is the main formula, it remains to take a gradient descent step for every local window:

$$-\log\sigma\left(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right) - \sum_{(i,c')\in D'} \log\sigma\left(-\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i\right).$$

- CBOW works in a very similar way.

- Another view of word2vec (Levy, Goldberg, 2014) – let's consider the loss function again:

$$\ell = \sum_{(i,c)\in D} \left(\log\sigma\left(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right) + k\mathbb{E}_{c'}\left[\log\sigma\left(-\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i\right)\right]\right).$$

- Another view of word2vec (Levy, Goldberg, 2014) – let's consider the loss function again:

$$\ell = \sum_{(i,c)\in D} \left(\log \sigma\left(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right) + k\mathbb{E}_{c'}\left[\log \sigma\left(-\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i\right)\right]\right).$$

- We rewrite it, collecting the sums w.r.t. each pair (i, c); let $n_{i,c}$ be the number of times it occurs, n_i — occurrences of i, n_c, of c:

$$\ell = \sum_i \sum_c \left(n_{i,c} \log \sigma\left(\tilde{\mathbf{w}}_c^\top \mathbf{w}_i\right) + kn_{i,c}\mathbb{E}_{c'}\left[\log \sigma\left(-\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i\right)\right]\right),$$

  and the second part is

$$\sum_i \sum_c kn_{i,c}\mathbb{E}_{c'}\left[\log \sigma\left(-\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i\right)\right] = \sum_i kn_i\mathbb{E}_{c'}\left[\log \sigma\left(-\tilde{\mathbf{w}}_{c'}^\top \mathbf{w}_i\right)\right],$$

- Before we estimated the expectation via samples, now let's write it down in full:

$$\mathbb{E}_{c'}\left[\log\sigma\left(-\tilde{\mathbf{w}}_{c'}^{\top}\mathbf{w}_i\right)\right] = \sum_{c'} \frac{n_{c'}}{|D|}\log\sigma\left(-\tilde{\mathbf{w}}_{c'}^{\top}\mathbf{w}_i\right) =$$

$$= \frac{n_c}{|D|}\log\sigma\left(-\tilde{\mathbf{w}}_{c}^{\top}\mathbf{w}_i\right) + \sum_{c'\neq c} \frac{n_{c'}}{|D|}\log\sigma\left(-\tilde{\mathbf{w}}_{c'}^{\top}\mathbf{w}_i\right).$$

- So w.r.t. each pair we have

$$\ell_{i,c} = n_{i,c}\log\sigma\left(\tilde{\mathbf{w}}_{c}^{\top}\mathbf{w}_i\right) + kn_i\frac{n_c}{|D|}\log\sigma\left(-\tilde{\mathbf{w}}_{c}^{\top}\mathbf{w}_i\right).$$

- To optimize this, we denote $x = \tilde{\mathbf{w}}_c^\top \mathbf{w}_i$ and differentiate w.r.t. x on both sides:

$$\frac{\partial \ell_{i,c}}{\partial x} = n_{i,c}\sigma(-x) - kn_i \frac{n_c}{|D|}\sigma(x).$$

- Equating to zero, we get a quadratic equation w.r.t. e^x:

$$e^{2x} - \left(\frac{n_{i,c}}{kn_i \frac{n_c}{|D|}} - 1 \right) e^x - \frac{n_{i,c}}{kn_i \frac{n_c}{|D|}} = 0.$$

- Two roots, -1 doesn't fit:

$$e^x = \frac{n_{i,c}}{kn_i \frac{n_c}{|D|}} = \frac{1}{k} \frac{n_{i,c}|D|}{n_i n_c}.$$

- Thus, to optimize the original likelihood we need to find $\tilde{\mathbf{w}}_c$ and $\mathbf{w}_i$ such that

$$\tilde{\mathbf{w}}_c^\top \mathbf{w}_i = \log\left(\frac{n_{i,c}|D|}{n_i n_c}\right) - \log k.$$

- $\log\left(\frac{n_{i,c}|D|}{n_i n_c}\right)$ is the pointwise mutual information (PMI).
- So finding $\tilde{\mathbf{w}}_c$ and $\mathbf{w}_i$ is basically singular decomposition of the PMI matrix!
- Question: why do we need separate $\tilde{\mathbf{w}}$ and $\mathbf{w}$ vectors?
- Live demo: nearest neighbors, simple geometric relations.

THANK YOU!

Thank you for your attention!