Distributional semantics (II)

CS 690N, Spring 2018

Advanced Natural Language Processing

http://people.cs.umass.edu/~brenocon/anlp2018/

Brendan O'Connor

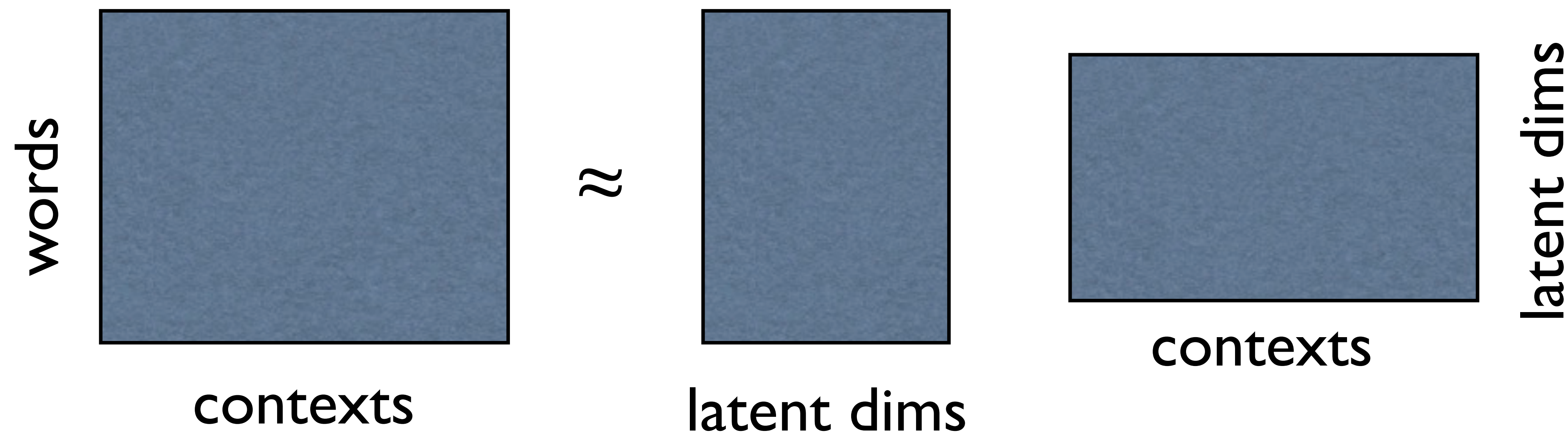
College of Information and Computer Sciences

University of Massachusetts Amherst

Matrix factorization

V ≈ E B

(counts) (embeddings)



Reconstruct the co-occurrence matrix

$$V_{i,c} \approx \sum_k E_{i,k} B_{k,c}$$

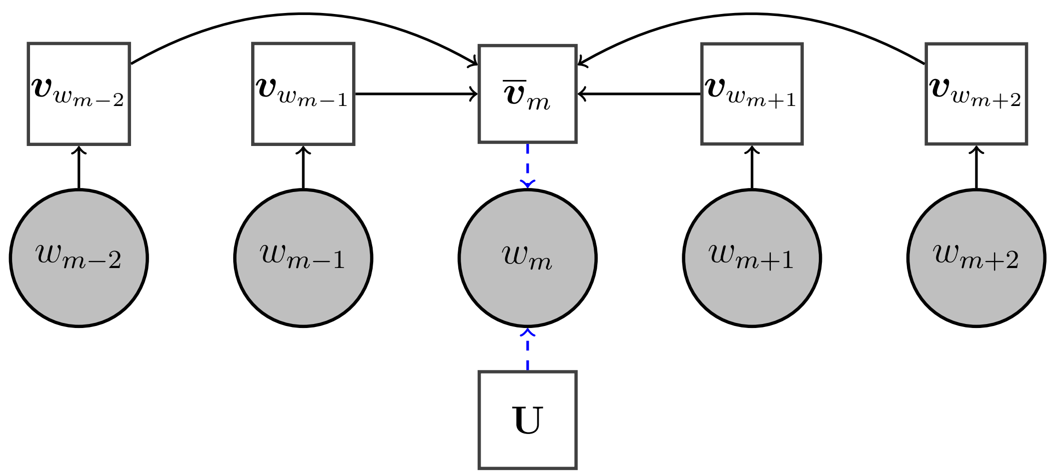
Singular Value Decomposition learns E,B
(or other matrix factorization techniques)

Preserve pairwise distances
between words i, j

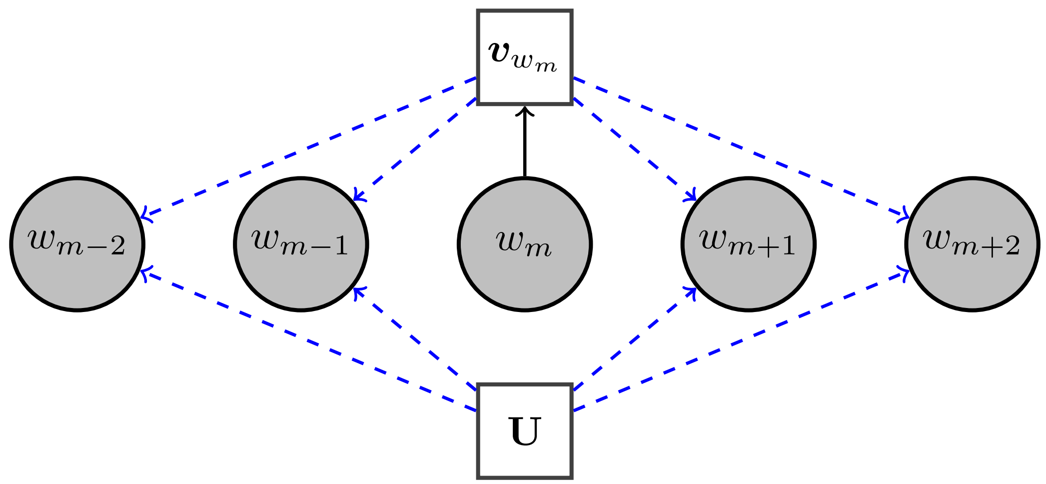
$$V_i^\mathsf{T} V_j \approx E_i^\mathsf{T} E_j$$

Eigen Decomposition learns E

Local models



(a) Continuous bag-of-words (CBOW)



(b) Skipgram

- CBOW: faster
- Skip-grams: work better
  (because more like context matrix factorization?)

Skip-gram model

$$u_\theta(w,c) = \exp\left(\boldsymbol{a}_w^\top \boldsymbol{b}_c\right)$$

$$J = \frac{1}{M}\sum_m \sum_{-c\le j\le c, j\ne 0} \log \mathrm{p}(w_{m+j} \mid w_m)$$

$$\mathrm{p}(w_{m+j} \mid w_m) = \frac{u_\theta(w_{m+j}, w_m)}{\sum_{w'} u_\theta(w', w_m)}$$

$$= \frac{u_\theta(w_{m+j}, w_m)}{Z(w_m)}$$

- *[Mikolov et al. 2013]*
- In word2vec. Learning: SGD under a contrastive sampling approximation of the objective
- Levy and Goldberg: mathematically similar to factorizing a PMI(w,c) matrix; advantage is streaming, etc. (though see Arora et al.'s followups...)
- Practically: very fast open-source implementation
- Variations: enrich contexts

Skip-gram model

$$\log \mathrm{p}(\boldsymbol{w}) \approx \sum_{m=1}^{M} \sum_{n=1}^{h_m} \log \mathrm{p}(w_{m-n} \mid w_m) + \log \mathrm{p}(w_{m+n} \mid w_m)$$
$$= \sum_{m=1}^{M} \sum_{n=1}^{h_m} \log \frac{\exp(\boldsymbol{u}_{w_{m-n}} \cdot \boldsymbol{v}_{w_m})}{\sum_{j=1}^{V} \exp(\boldsymbol{u}_i \cdot \boldsymbol{v}_{w_m})} + \log \frac{\exp(\boldsymbol{u}_{w_{m+n}} \cdot \boldsymbol{v}_{w_m})}{\sum_{j=1}^{V} \exp(\boldsymbol{u}_i \cdot \boldsymbol{v}_{w_m})}$$
$$= \sum_{m=1}^{M} \sum_{n=1}^{h_m} \boldsymbol{u}_{w_{m-n}} \cdot \boldsymbol{v}_{w_m} + \boldsymbol{u}_{w_{m+n}} \cdot \boldsymbol{v}_{w_m} - 2 \log \sum_{j=1}^{V} \exp\left(\boldsymbol{u}_i \cdot \boldsymbol{v}_{w_m}\right)$$

- *[Mikolov et al. 2013]*
- In word2vec. Learning: SGD under a contrastive sampling approximation of the objective
- Levy and Goldberg: mathematically similar to factorizing a PMI(w,c) matrix; advantage is streaming, etc. (though see Arora et al.'s followups...)
- Practically: very fast open-source implementation
- Variations: enrich contexts

Dealing with large output vocab

- Hierarchical softmax
  - Helps to have a good hierarchy: for example, Brown clusters work well. Or less expensive alternatives

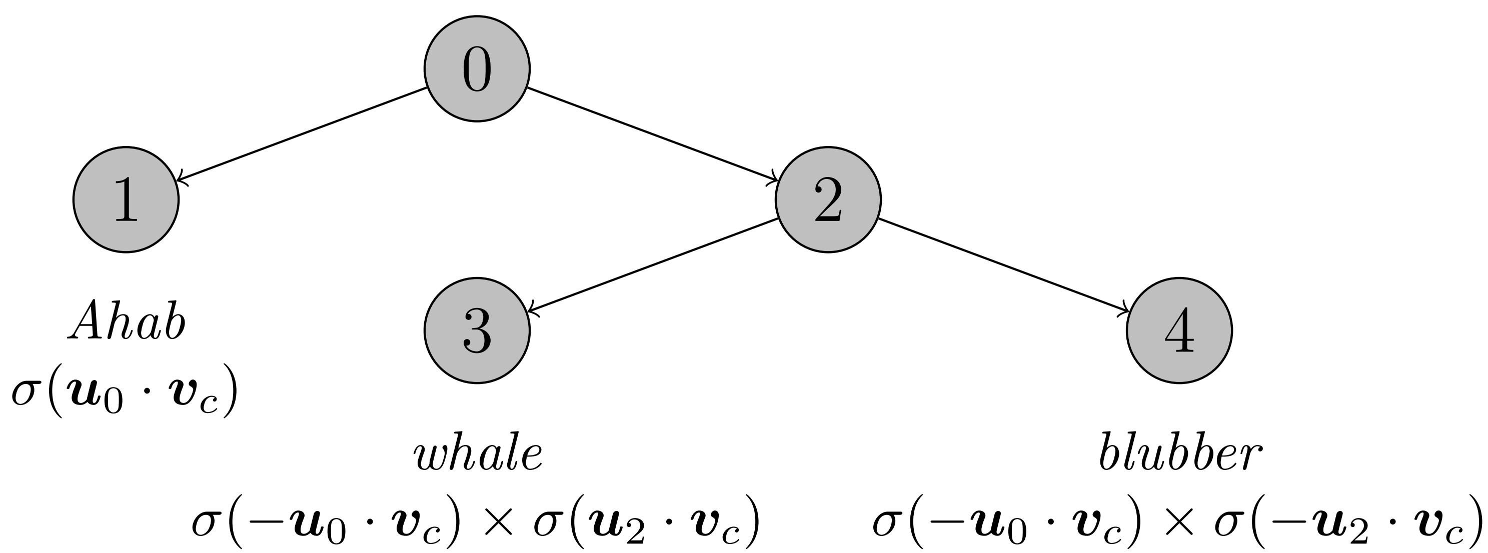


Figure 13.4: A fragment of a hierarchical softmax tree. The probability of each word is computed as a product of probabilities of local branching decisions in the tree.

$$\psi(i,j) = \log \sigma(\boldsymbol{u}_i \cdot \boldsymbol{v}_j) + \sum_{i' \in \mathcal{W}_{\text{neg}}} \log(1 - \sigma(\boldsymbol{u}_{i'} \cdot \boldsymbol{v}_j))$$

- Negative sampling
  - Choose negative set somehow -- e.g. a (rescaled) unigram language model (2-5? 5-20? samples)
- Levy and Goldberg show equivalence to word-context matrix factorization, where matrix cells are:

$$M_{ij} = \max(0, \text{PMI}(i,j) - \log k)$$

- “Distributional / Word Embedding” models
  - Typically, they learn embeddings to be good at word-context factorization,
    which seems to often give useful embeddings
- Pre-trained embeddings resources
  - *GLOVE*, *word2vec*, etc.
  - Make sure it’s trained on a corpus sufficiently similar to what you care about!
- How to use?
  - Fixed (or initializations) for word embedding model parameters
  - Similarity lookups

Extensions

- Alternative: Task-specific embeddings (always better...)
- Multilingual embeddings
- Better contexts: direction, syntax, morphology / characters...
- Phrases and meaning composition
  - vector(hardly awesome) = g(vector(hardly), vector(awesome))