Annotated version

Machine Learning Course - CS-433

Text Representation Learning

Nov 24, 2016

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

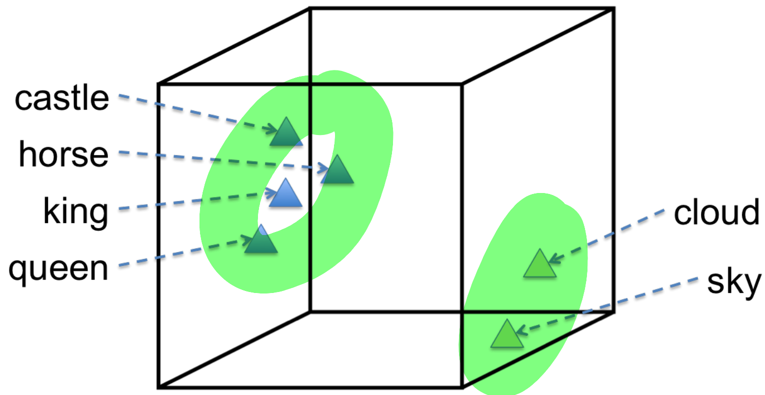
Motivation

Finding numerical representations for words is fundamental for all machine learning methods dealing with text data.

Goal: For each word, find mapping (embedding)

$$w_i \mapsto \underline{\mathbf{w}_i \in \mathbb{R}^K}$$

Representation should capture semantics of the word.



Constructing good feature representations (= representation learning) benefits all ML applications.

Bag of words

$$w_i \rightarrow \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow i \quad \mathbb{R}^{1M}$$

The Co-Occurence Matrix

A big corpus of un-labeled text can be represented as the co-occurrence counts

n_{ij} := #contexts where word w_i occurs together with word w_j.

w_n

w_d

	1	1		
		3		
	1			
	2		1	
1				1
		1		
	1		1	1

0 = „unobserved" (not part of Ω)

n_{dn} += 1

…… | w_d | · | · | w_n | · | …

Needs definition of

- Context e.g. document, paragraph, sentence, window

window size 5

- Vocabulary
  $\mathcal{V} := \{w_1, \ldots, w_D\}$

$|V| = 1M$

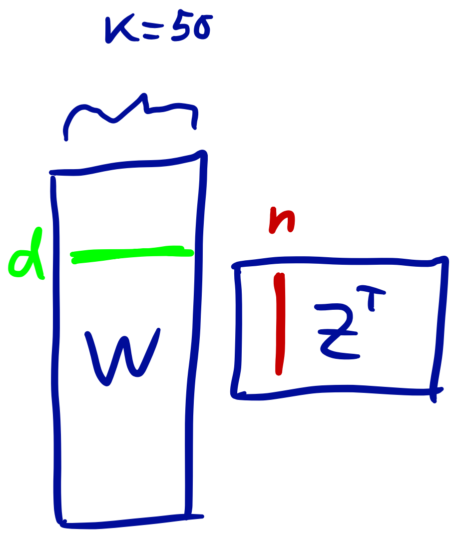
For words $w_d = 1, 2, \ldots, D$ and context words $w_n = 1, 2, \ldots, N$, the co-occurence counts n_{ij} form a very sparse $D \times N$ matrix.

Learning Word-Representations (Using Matrix Factorization)

Find a factorization of the co-occurence matrix!
Typically uses log of the actual counts, i.e. $x_{dn} := \log(n_{dn})$.

We will aim to find $\mathbf{W}, \mathbf{Z}$ s.t.

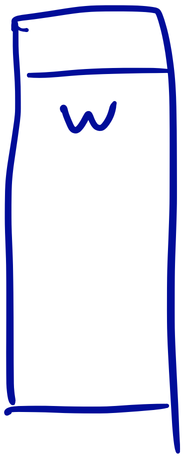
$$\boxed{\mathbf{X} \approx \mathbf{W}\mathbf{Z}^\top \, .}$$



So for each pair of words (w_d, w_n), we try to 'explain' their co-occurence count by a numerical representation of the two words
- in fact by the inner product of the two feature vectors $\mathbf{W}_{d:}, \mathbf{Z}_{n:}$.

$$\min_{\mathbf{W},\mathbf{Z}} \; \mathcal{L}(\mathbf{W},\mathbf{Z}) := \frac{1}{2} \sum_{(d,n)\in\Omega} f_{dn} \left[x_{dn} - (\mathbf{W}\mathbf{Z}^\top)_{dn} \right]^2$$

where $\mathbf{W} \in \mathbb{R}^{D\times K}$ and $\mathbf{Z} \in \mathbb{R}^{N\times K}$ are tall matrices, having only $K \ll D, N$ columns.
The set $\Omega \subseteq [D] \times [N]$ collects the indices of non-zeros of the count matrix $\mathbf{X}$.
Each row of those matrices forms a representation of a word ($\mathbf{W}$) or a context word ($\mathbf{Z}$) respectively.



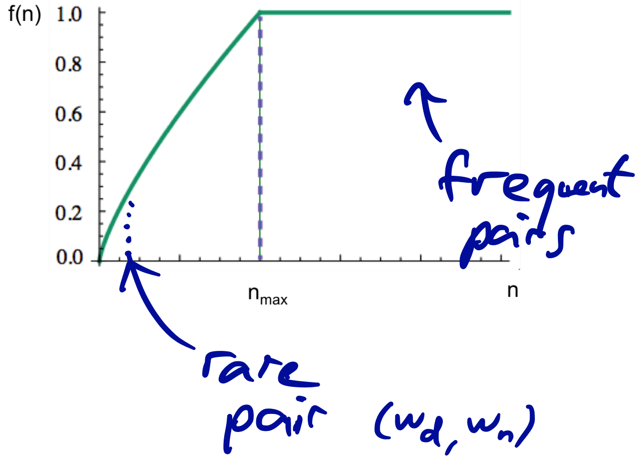
GloVe

This model is called GloVe, and is a variant of word2vec.

Weights f_{dn}: Give "importance" of each entry. Choosing $f_{dn} := 1$ is ok. GloVe weight function:

$$f_{dn} := \min\left\{1, (n_{dn}/n_{\max})^{\alpha}\right\}, \quad \alpha \in [0;1] \quad \text{e.g. } \alpha = \tfrac{3}{4}$$

$\#(w_d, w_n)$

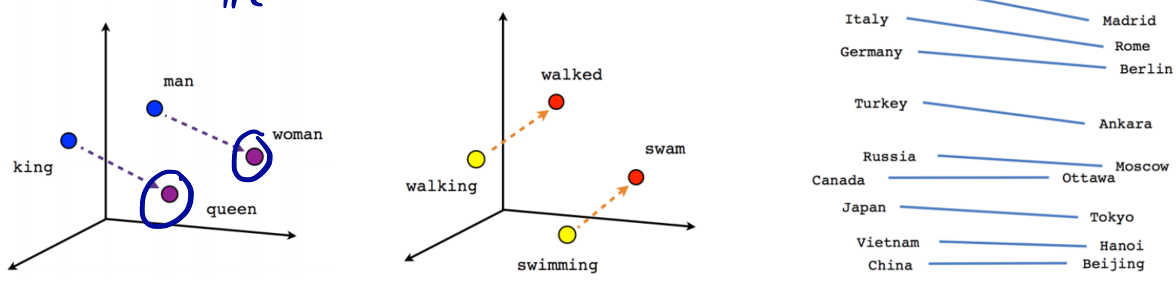


Choosing K

K e.g. 50, 100, 200

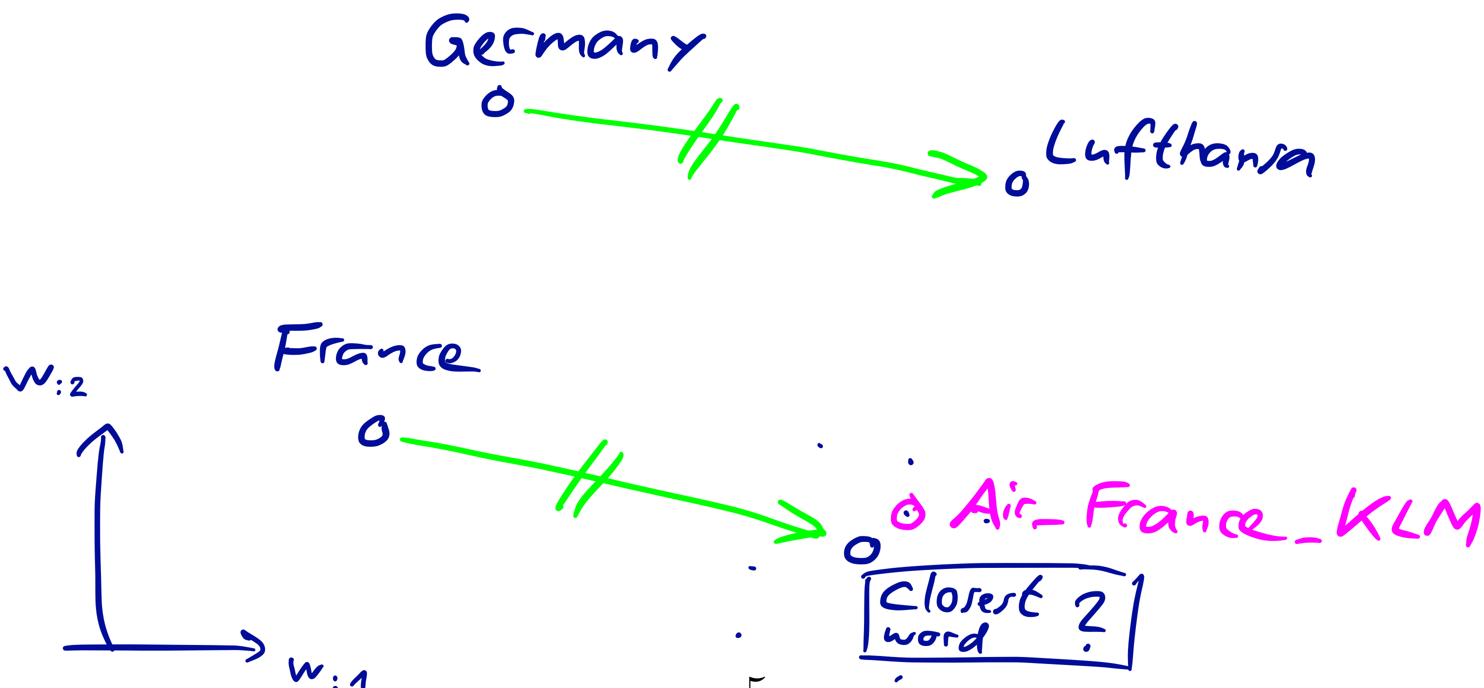
Word Analogies

$\mathbb{R}^{50}$



Male-Female Verb tense Country-Capital

Newspapers			
New York San Jose	New York Times San Jose Mercury News	Baltimore Cincinnati	Baltimore Sun Cincinnati Enquirer
NHL Teams			
Boston Phoenix	Boston Bruins Phoenix Coyotes	Montreal Nashville	Montreal Canadiens Nashville Predators
NBA Teams			
Detroit Oakland	Detroit Pistons Golden State Warriors	Toronto Memphis	Toronto Raptors Memphis Grizzlies
Airlines			
Austria Belgium	Austrian Airlines Brussels Airlines	Spain Greece	Spainair Aegean Airlines
Company executives			
Steve Ballmer Samuel J. Palmisano	Microsoft IBM	Larry Page Werner Vogels	Google Amazon



find w, z

Training

- Stochastic Gradient Descent (SGD)
- Alternating Least-Squares (ALS)

Open questions:

- Parallel and distributed training
- Does regularization help?

Alternative: Skip-Gram Model

(Original word2vec)

Uses binary classification (logistic regression objective), to separate good word pairs (w_d, w_n) from bad word pairs. Same inner product score = matrix factorization.

$w_{d:} \cdot z_{n:}^\top$

Given w_d, a context word w_n is

- Good = appearing together in a context window of 5
- Bad = any word $w_{n'}$ sampled randomly: Negative sampling

Learning Representations of Documents

(Sentences)

Supervised:

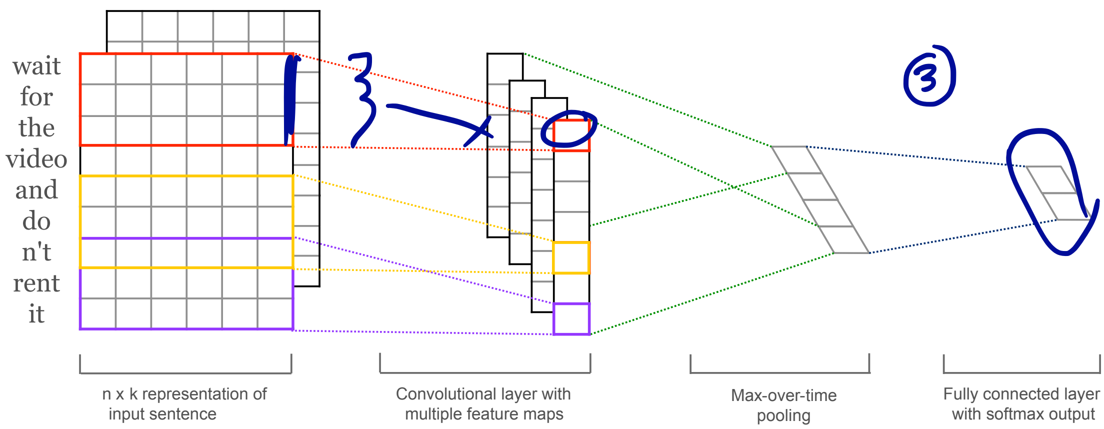
For a supervised task (e.g. predicting the emotion of a tweet), we can use matrix-factorization (below) or convolutional neural networks (see next weeks).

① add up word vectors

②

③

W



→ SemEval competition for tweet classification.

Unsupervised:

Open research.

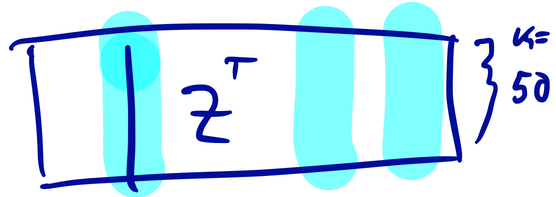
FastText

Matrix factorization to learn document/sentence representations (supervised).

$$x_n = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 1 \\ 0 \end{pmatrix} \in \mathbb{R}^m$$

of length m

Given a sentence $s_n = (w_1, w_2, \dots, w_m)$, let $\mathbf{x}_n \in \mathbb{R}^{|\mathcal{V}|}$ be the bag-of-words representation of the sentence.



$$\min_{\mathbf{W},\mathbf{Z}} \mathcal{L}(\mathbf{W}, \mathbf{Z}) := \sum_{s_n \text{ a sentence}} f(y_n \mathbf{W}\mathbf{Z}^\top \mathbf{x}_n)$$

where $\mathbf{W} \in \mathbb{R}^{1 \times K}$, $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times K}$ are the variables, and the vector $\mathbf{x}_n \in \mathbb{R}^{|\mathcal{V}|}$ represents our n-th training sentence.

Here f is a linear classifier loss function, and y_n is the classification label for sentence $\mathbf{x}_n$.

label $y_n \in \{-1, 1\}$

Supervised, need data (y_n, x_n)

model

Further Pointers

1. word2vec:
   code: code.google.com/p/word2vec/
   paper:
   "Distributed representations of words and phrases and their compositionality" - T Mikolov, I Sutskever, K Chen, GS Corrado, J Dean. NIPS 2013

2. GloVe:
   code and vectors: nlp.stanford.edu/projects/glove/
   paper:
   "GloVe: Global Vectors for Word Representation" - Pennington, J., Socher, R., Manning, C. D.. EMNLP 2014

3. FastText
   code: github.com/facebookresearch/fastText
   papers:
   "Bag of Tricks for Efficient Text Classification" - Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. - arXiv, 2016.
   "Enriching Word Vectors with Subword Information" - Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. - arXiv, 2016.