Computational Intelligence Laboratory

Tutorial session 6

LDA and word embeddings

Antonio Orvieto

ETH Zurich – cil.inf.ethz.ch

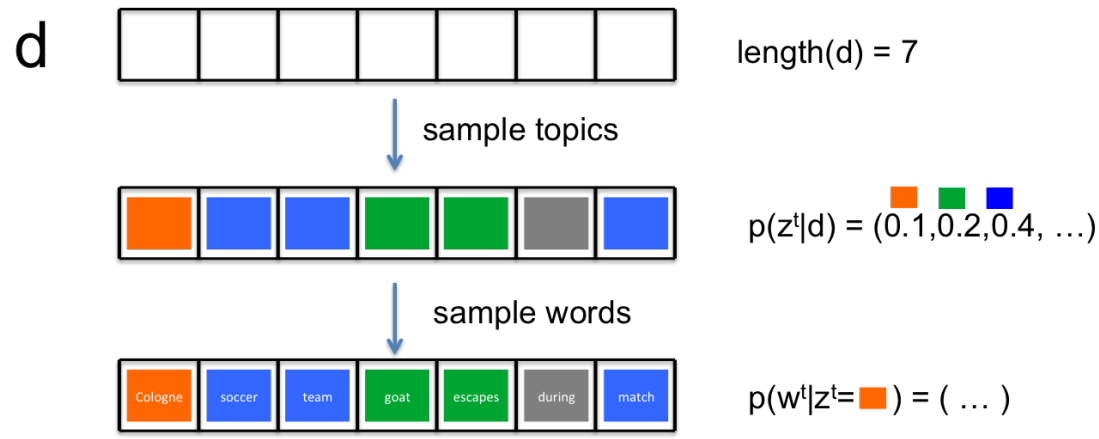
28 March 2020

Section 1

Latent Dirichlet Allocation

pLSA (model)

- each document = specific mix of topics (colors): $p(z|d)$
- each topic (color) = specific distribution of words: $p(w|z)$



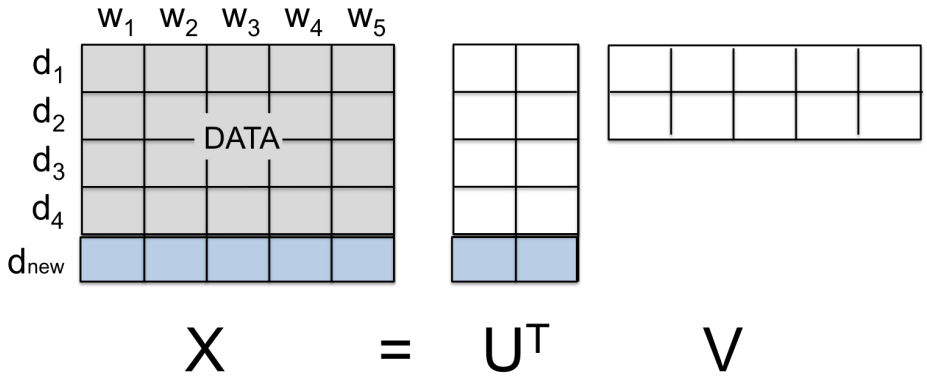
$$p(w|d) = \sum_z p(w, z|d) = \sum_z p(w|d, z)p(z|d) \stackrel{*}{=} \sum_z p(w|z)p(z|d)$$

Parameters: matrices U and V, containing probability distributions (probabilities of topics given a specific document and of words given a specific topic).

Optimized with EM (see lecture..)

Exercise 1: Limitations of pLSA

- $p(z|d)$ learned *only for documents on which model is trained*. **Not a well-defined generative model**. How to sample new document?



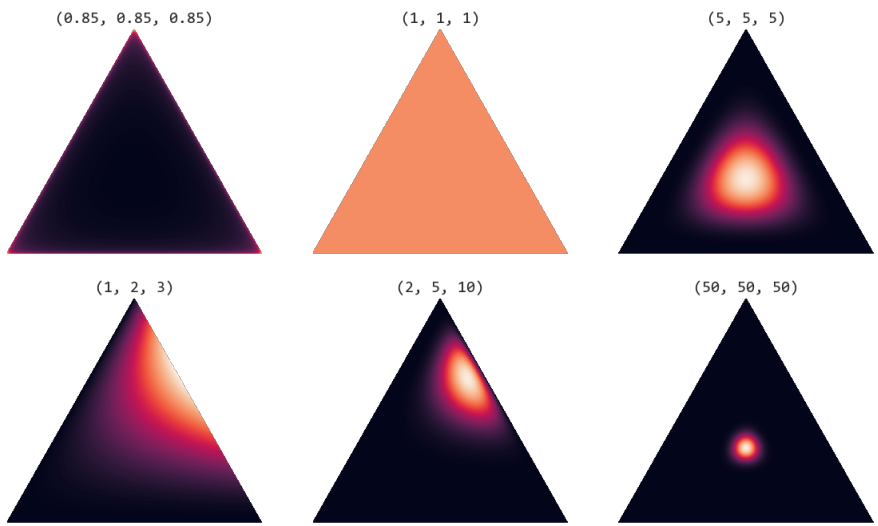
  - Need to be able to sample topic weights $u_i = (u_{1i}, \ldots, u_{Ki})^\top$ for a new document. Combine with existing V to predict d_i.
  - pLSA has no good way to provide a new u_i Thus, pLSA has to use a heuristic: new document is "folded in" and EM is re-run (holding the old parameters fixed) to estimate the topic proportion parameter for this new document.

- **number of parameters grows linearly with corpus**.

The topics simplex and the Dirichlet distribution

u_i contains probabily of each topic for document i. Instead of learning it (for each i) as a parameter, can we instead generate it from a *meaningful* distribution?

$$p(u_i|\alpha) = \frac{1}{B(\alpha)} \prod_{z=1}^{K} u_{zi}^{\alpha_z - 1}. \qquad B(\alpha) = \frac{\prod_{z=1}^{k} \Gamma(\alpha_z)}{\Gamma(\sum_{z=1}^{k} \alpha_z)}$$



distribution of 3 topic, for a chosen hyperparameter α,
https://towardsdatascience.com/dirichlet-distribution-a82ab942a879

From original paper: *The Dirichlet is a convenient distribution on the simplex — it is in the exponential family, has finite dimensional sufficient statistics, and is conjugate to the multinomial distribution.*

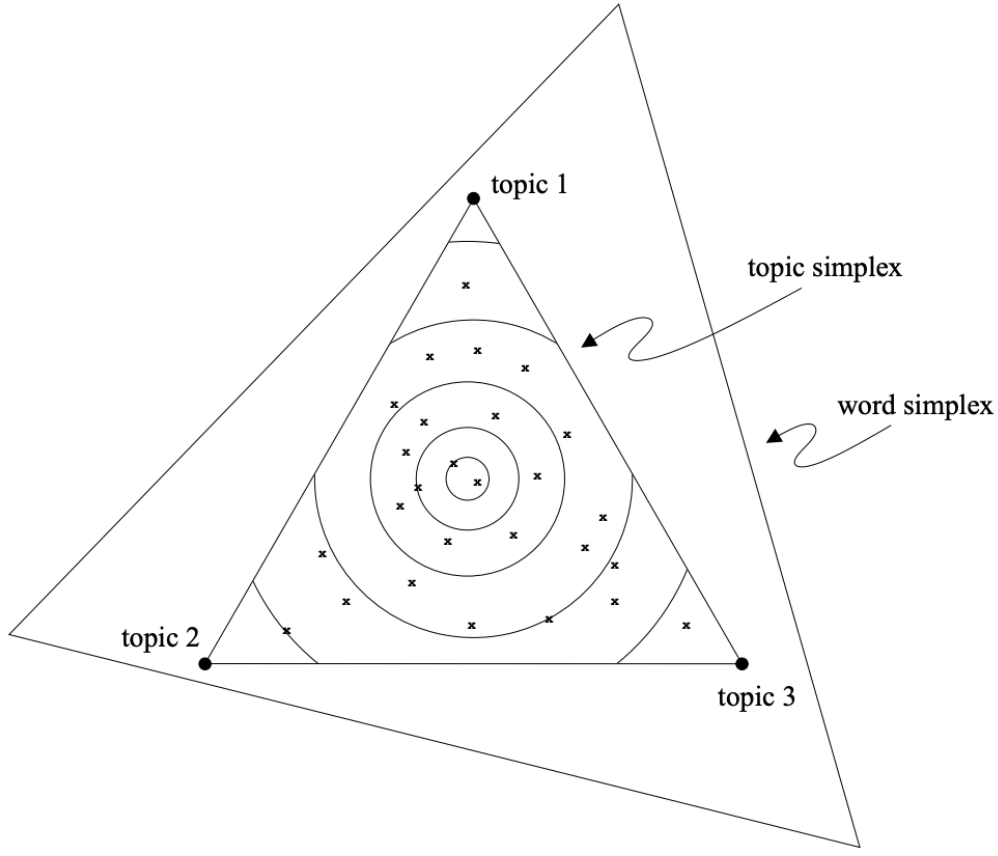


Figure 4: The topic simplex for three topics embedded in the word simplex for three words. The corners of the word simplex correspond to the three distributions where each word (respectively) has probability one. The three points of the topic simplex correspond to three different distributions over words. The mixture of unigrams places each document at one of the corners of the topic simplex. The pLSI model induces an empirical distribution on the topic simplex denoted by x. LDA places a smooth distribution on the topic simplex denoted by the contour lines.

http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

Exercise 1 (ii): Conjugate prior of multinomial distribution

- **Multinomial distribution**. Consider a simplified setting:
  - total of N words in our dictionary, and we pick (with replacement) L_i words to form document d_i;
  - order of the words does not matter: document summarized in vector x_i, where x_{ij} counts occurrence of word j in document i;
  - probability of word j in document i is π_{ij}.

  Simple combinatorics *give* (i.e. not an assumption)

$$p(x_i|\pi_i) = \frac{L_i!}{\prod_j x_{ij}!} \prod_{j=1}^{N} \pi_{ij}^{x_{ij}}, \quad x_i \sim \text{Multi}(\pi_i).$$

- If $\pi_i \sim \text{Dir}(\alpha_i)$, then something magic happens to the posterior..

$$p(\pi_i|x_i, \alpha) \propto p(x_i|\pi_i)p(\pi_i|\alpha_i)$$

$$= \left(\frac{L_i!}{\prod_j x_{ij}!} \prod_{j=1}^{N} \pi_{ij}^{x_{ij}} \right) \left(\frac{1}{B(\alpha_i)} \prod_{j=1}^{N} \pi_{ij}^{\alpha_{ij}-1} \right) \propto \prod_{j=1}^{N} \pi_{ij}^{x_{ij}+\alpha_{ij}-1}.$$

  $\Rightarrow$ **posterior also going to be Dirichlet**, with parameter $x_i + \alpha_i$.

Data likelihood under the Latent Dirichlet allocation

- we sample $u_i = (u_{1i}, \ldots, u_{Ki}) \sim \mathrm{Dir}(\alpha)$ and a document lenght L_i, described as vector x_i (bag-of-words), where x_{ij} is count of word j.
- The observation model picked to be a standard *multinomial*:

$$p\left(x_i | V, u_i\right) = \frac{L_i!}{\prod_j x_{ij}!} \prod_j \pi_{ij}^{x_{ij}}, \quad \pi_{ij} := \sum_{z=1}^{K} v_{zj} u_{zi}.$$

  π_{ij} is the probability that w_j sampled, given mixture of topics u_i.
- Finally, we integrate out (i.e. *get rid of*) u_i and **get the likelihood**.

$$\begin{aligned} p\left(x_i | V, \alpha\right) &= \int p\left(x | V, u_i\right) p(u_i | \alpha) du_i \\ &= \frac{1}{B(\alpha)} \int \left(\prod_z u_{zi}^{\alpha_z - 1} \right) \left(\frac{L_i!}{\prod_j x_{ij}!} \prod_j \pi_{ij}^{x_{ij}} \right) du_i. \end{aligned}$$

  Evaluation *intractable* due to coupling between u_i and V.
- Using **approximate inference** (Gibbs sampling etc) and properties such as *conjugacy*, one can get a way to infer V from the corpus.

Exercise 1 (iii): So, is LDA superior?

More pros of LDA:

- pLSA is shown to be more susceptible to overfitting than LDA for small datasets;
- the Dirichlet prior acts as a regularizer on the topics distribution;
- model is theoretically sound and motivated;

But..

- If your corpus is fixed, not clear one would need a generative model (even though it might be elengant);
- pLSA is far more easy to implement, used more often in industry;
- pLSA and LDA often achieve comparable performance;
- the hyperparameter α, if chosen in a wrong way, can make LDA way worse that pLSA.

Investigating task performance of probabilistic topic models: an empirical study of pLSA and LDA
by Yue Lu, Qiaozhu Mei, ChengXiang Zhai
`https://link.springer.com/content/pdf/10.1007/s10791-010-9141-9.pdf`

Section 2

Word Embeddings

What is an embedding? how is this useful?

- Lately, we discussed how to represent and discover *topics* in a corpus of documents. Key problem in information retrieval;
- Now, we want to learn a semantic representation (i.e. embedding) of words, so we can perform tasks in NLP.

Fundamentals:

- meaning of a word only depends on its use in language (Wittgenstein, 1953);
- our embedding of a word should summarize use we make of such word.. which also depends on all relations among all other words;
- can get a bit mind-bending.. especially since vocabularies are finite! Similar concepts led in philosophy to (post-)structuralism.

> *... the central signified, the original or transcendental signified, is* **never absolutely present outside a system of differences**. *The absence of the transcendental signified extends the domain and the interplay of signification ad infinitum.*
>
> – Derrida

Signifier is the sound associated with or image of something (e.g., a tree), the signified is the idea or concept of the thing (e.g., the idea of a tree), and the sign is the object that combines the signifier and the signified into a meaningful unit. `http://oregonstate.edu/instruct/theory/signs.html`

Context Model Likelihood

Suppose we want to learn an embedding from a text $\mathbf{w}$ of length T.
For each word $v \in \mathcal{V}$ we want to learn a vector representation $\mathbf{x}_v$, able to predict how v is used.

step 1) Given a window $\mathcal{I} = \{-R, \ldots, -1, 1, \ldots, R\}$, the likelihood of $\mathbf{w}$ is

$$\mathcal{L}(\theta; \mathbf{w}) = \sum_{t=1}^{T} \sum_{\triangle \in \mathcal{I}} \log p_\theta(w^{(t+\triangle)} \mid w^{(t)}).$$

step 2) Introduce a similarity measure (log-bilinear model)

$$\log p_\theta(w \mid w') = \langle \mathbf{x}_w, \mathbf{x}_{w'} \rangle + b_w + const.$$

step 3) Compute the normalizing constant and give a final formula for the log-likelihood

$$\mathcal{L}(\theta; \mathbf{w}) = \sum_{t=1}^{T} \sum_{\triangle \in \mathcal{I}} \left[b_{w^{(t+\triangle)}} + \langle \mathbf{x}_{w^{(t+\triangle)}}, \mathbf{x}_{w^{(t)}} \rangle - \log \sum_{v \in \mathcal{V}} \exp\left[\langle \mathbf{x}_v, \mathbf{x}_{w^{(t)}} \rangle + b_v \right] \right].$$

step 4) A useful first modification is to introduce input-output embeddings (and to modify the cost accordingly)

$$\log p_\theta(w \mid w') = \langle \mathbf{x}_w, \mathbf{y}_{w'} \rangle + b_w.$$

step 5) (Exercise 3(i)) Realize that this likelihood is actually a scary monster... how can we optimize it?! Computing gradient of the normalization constant intractable (involves sum over dictionary).

$$\frac{\partial}{\partial \mathbf{x}_v} \log \left(\sum_{w' \in \mathcal{V}} \exp[\langle \mathbf{x}_{w'}, \mathbf{y}_w \rangle] \right) = \frac{\exp(\langle \mathbf{y}_w, \mathbf{x}_v \rangle)}{\sum_{w' \in \mathcal{V}} \exp(\langle \mathbf{y}_{w'}, \mathbf{x}_v \rangle)} \mathbf{y}_w.$$

Therefore, we need to modify this likelihood so that it is tractable to optimize with stochastic gradient descent.

(Exercise 3) Negative sampling

Main idea: a good embedding yields a classifier between semantically similar/non-similar words..

- observed (positive) pairs $\Longrightarrow$ positive training examples $\triangle^+$.
- random (negative) pairs$\Longrightarrow$ negative training examples $\triangle^-$.

Therefore we may just perform logistic regression, $\sigma(z) := \frac{1}{1+\exp(-z)}$, i.e. maximize (ignoring bias)

$$\mathcal{L}(\theta) = \sum_{(i,j)\in\triangle^+} \log \sigma(\langle \mathbf{x}_i, \mathbf{y}_j \rangle) + \sum_{(i,j)\in\triangle^-} \log \sigma(-\langle \mathbf{x}_i, \mathbf{y}_j \rangle).$$

- computation does not depend on $|\mathcal{V}|$ and should be fast enough to allow for cheap gradient updates.
- the overall gradient is completely decoupled into a sum, thus one can update the objective in sampled batches of the data.

(Exercise 4) GloVe

Main idea: why don't we just try to *match* the co-occurrence count of two words with the log-bilinear model?

- Data is organized in a matrix

$$\mathbf{N} = (n_{ij}) \in \mathbb{N}^{|\mathcal{V}|\cdot|\mathcal{C}|},$$

$$n_{ij} = \#\text{ occurrences of } w_i \in \mathcal{V} \text{ in context of } w_j \in \mathcal{C}$$

- The GloVe cost is

$$\mathcal{H}(\theta; \mathbf{N}) = \sum_{i,j} f(n_{ij}) \left(\underbrace{\log n_{ij}}_{\text{target}} - \underbrace{\log \tilde{p}_\theta(w_i|w_j)}_{\text{model}} \right)^2,$$

$$\tilde{p}_\theta(w_i|w_j) = \exp\left[\langle \mathbf{x}_i, \mathbf{y}_j \rangle + b_i + c_j\right],$$

where $f(n_{ij}) \to 0$ as $n_{ij} \to 0$, and saturates, to decrease effect of rare/too frequent words.

- Cost can be optimized very efficiently with SGD!

$$\mathbf{x}_i^{\text{new}} \leftarrow \mathbf{x}_i + 2\eta f(n_{ij}) \left(\log n_{ij} - \langle \mathbf{x}_i, \mathbf{y}_j \rangle\right) \mathbf{y}_j$$

$$\mathbf{y}_j^{\text{new}} \leftarrow \mathbf{y}_j + 2\eta f(n_{ij}) \left(\log n_{ij} - \langle \mathbf{x}_i, \mathbf{y}_j \rangle\right) \mathbf{x}_i$$