Homework #4

Yongjin Shin
20090488
Industrial Management Engineering, POSTECH
dreimer@postech.ac.kr

November 26, 2018

Problem 1. Explaining what the strengths and weakness of PLSA compared to LSA are.

Solution

1. About LSA

The latent in Latent Semantic Analysis (LSA) means latent topics. Basically, LSA finds low-dimension representation of documents and words. Given documents $D = \{d_1, ..., d_m\}$ and vocabulary words $W = \{w_1, ..., w_n\}$, we construct a document-term matrix $X \in \mathbb{R}^{m \times n}$ where x_{ij} describes the occurrence of word w_j in document d_i. For example, x_{ij} can be the raw count, 0-1 count, or TF-IDF. The dot product of row vectors in the document similarity, while the dot product of column vectors is the word similarity. To reduce the dimensionality of X, apply truncated Singular Value Decomposition(SVD):

$$X \approx U_t \Sigma_t V_t^T \tag{1}$$

Each column of $U_t \in \mathbb{R}^{m \times t}$ and $V_t \in \mathbb{R}^{n \times t}$ corresponds to a document topic.

1.1. Weakness

Though LSA is easy to use, (1) it lacks of interpretable embeddings. We dont even know what the topics are, and the components may be arbitrarily either positive or negative. Also (2) since reconstruction may contain negative entries, L2 norm might be inappropriate as a distance function for count vectors. And (3) Choosing the number of dimensions in LSA is typically based on heuristics. Lastly, (4) it can fail to handle polysemy, meaning that the coexistence of many possible meanings for a word or phrase, since each occurence of a word being represented as a single point in space. On the other hand PLSA can resolve these issues in LSA

2. About PLSA

Instead of using matrices, Probabilistic Latent Semantic Analysis(PLSA) uses a probabilistic method. Its graphical model is as follows:

$$\begin{aligned} p(d,w) &= \sum_z p(d,w|z)p(z) \\ &= \sum_z p(d|z)p(w|z)p(z) \\ &= \sum_z \frac{p(z|d)p(d)}{p(z)} p(w|z)p(z) \\ &= p(d)\sum_z p(w|z)p(z|d) \end{aligned} \tag{2}$$

where d =document index, z =word's topic drawn from $p(z|d)$, w =word drawn from $p(w|z)$. Both $p(z|d)$ and $p(w|z)$ are modeled as multi-nomial distributions. The parameters can be trained with EM algorithms.

If we take a look at *equation 2* closely, we can see the relation to LSA. From the second equality of *equation 2*, if we make it with matrix form:

$$\begin{aligned} p(d,w) &= \sum_z p(d|z)p(z)p(w|z) \\ &= \hat{U} \cdot \hat{S} \cdot \hat{V}^T \end{aligned} \tag{3}$$

where $\hat{U} = p(d_i|z_k)$, $\hat{V} = p(w_i|z_k)$ and $\hat{S} = diag(p(z_k))$

2.1. Strengths

(1) The directions in the PLSA space are interpretable as multi-nomial word distributions. Since PLSA relies on the likelihood function of multi-nomial sampling and aims at an explicit maximization of the predictive power of the model, this corresponds to a minimization of the KL divergence between the empirical distribution and the model. This offers that the mixture approximation p of the co-occurrence table is a well-defined probability distribution and factors have a clear probabilistic meaning. Therefore probabilistic systems allow for the evaluation of propositions under conditions of uncertainty.

(2) This probabilistic approach can also take advantage of the well established statistical theory for model selection and complexity control. Not like LSA, we can find the probability of each mixture components so that we might have some clue to determine the optimal number of latent space dimensions, so called K.

(3) Probabilistic systems provide a uniform mechanism for integrating and reasoning over heterogeneous information. In PLSA ,semantic dimensions are represented by unigram language models, more transparent that eigenvectors. Also the latent variable structure allows for subtopics so that more complicated model can be achieved such as hierarchical PLSA.

(4) PLSA is able to deal with the polysemy of words. For a potentially ambiguous word, the probability that a particular term occurrence w_j in document d_i is associated with concept z_r, different contexts d_i correspond to different probabilities $p(z_r|d_i)$ and hence yield differenc posterior probabilities.

2.2. Weakness

(1) It offers no probabilistic structure at the level of documents. In pLSA, the document probability is a fixed point in the dataset so that if we havent seen a document, we dont have that data point. Since we have no parameters to model $p(d)$, we dont know how to assign probabilities to new documents. Still we can do some EM process with new document such that computing $p(z|w, d_{new})$ at E-step, and computing $p(z|d_{new})$ with keeping all other parameters unchanged. But this will be discriminative so that it is hard to say that it is a proper generative model.

(2) Another weak point is that PLSA is easy to get huge number of parameters as $p(D|Z)$ grows linearly with the number of documents we have. Therefore it is prone to overfitting easily.

(3) And PLSA leads to unstable estimation since it can fall into the local maxima. Even though the author mentioned that it does not affect that much, PLSA still have a chance to be in the wrong way. Even NMF is for optimizing the same objective function with PLSA, PLSA can jump out of the local minima where NMF converges to and vice versa[5]. Which means that PLSA cannot guarantee that it will have the best solution.

Latent Dirichlet Allocation generalize PLSA by changing the fixed d to a Dirichlet prior so that it alleviates the first weakness of PLSA and was the first fully probabilistic model for text clustering. In LDA, therefore, the dataset serves as training data for the dirichlet distribution of document-topic distributions. If we havent seen a document, we can easily sample from the dirichlet distribution and move forward from there.

Problem 2. (Optional) How to generate samples from Dirichlet distribution?

Solution

1. Polya's Urn

Suppose we want to generate a realization of $Q \sim Dir(\alpha)$. To start, put α_i balls of color i for $i = 1, 2, ..., k$ in an urn. (Note that $\alpha_i > 0$ is not necessarily an integer, so we may have a fractional or irrational number of balls of color i in the urn.) At each iteration, draw one ball uniformly at random from the urn, and then place it back into the urn along with an additional ball of the same color. As we iterate this procedure more and more times, the proportions of balls of each color will converge to a probability mass function that is a sample from the distribution $Dir(\alpha)$.

Step 1: Set a counter n=1. Draw $X_1 \sim \alpha/\alpha_0$. α/α_0 is a non-negative vector whose entries sum to 1, so it it a pmf.

Step 2: Update the counter to $n+1$. Draw $X_{n+1}|X_1, X_2, ..., X_n \sim \alpha_n/\alpha_{n0}$, where $\alpha_n = \alpha + \sum_{i=1}^{N} \delta x_i$ and α_{n0} is the sum of the entries of α_n. Repeat this step an infinite number of times.

After finishing Step2, calculate the proportions of the different colors: let $Q_n = (Q_{n1}, Q_{n2}, ..., Q_{nk})$, where Q_{ni} is the proportion of balls of color i after n balls are in the urn. Then Q_n converges into $Q \sim Dir(\alpha)$ as $n \to \infty$. The probability of drawing balls of each color is given by a pmf that is a realization of the distribution $Dir(\alpha)$. Thus asymptotically we have a sample from $Dir(\alpha)$.

2. The Stick-breaking Approach

The stick-breaking approach to generating a random vector with a $Dir(\alpha)$ distribution involves iteratively breaking a stick of length 1 into k pieces in such a way that the lengths of the k pieces follow a $Dir(\alpha)$ distribution. Assume that we know how to generate random variables from the Beta distribution. In the case where α has length 2, simulating from the Dirichlet is equivalent to simulating from the Beta distribution.

Assume that $k = 3$, and then generalize the procedure to $k > 3$. Over the course of the stick-breaking process, we will keeping track of a set of intermediate values $\{u_i\}$, which we use to ultimately calculate the realization q. To begin, we generate Q_1 from $Beta(\alpha_1, \alpha_2 + \alpha_3)$ and set u_1 equal to its values: $u_1 = q_1$. Then generate $(\frac{Q_2}{1-Q_1}|Q_1)$ from $Beta(\alpha_2, \alpha_3)$. The result by u_2, and set $q_2 = (1-u_1)u_2$. The resulting vector $u = (u1, (1-u_1)u_2, 1-u_1-(1-u_1)u_2)$ comes from a Dirichlet distribution with parameter vector α.

Step 1: Simulate $u_1 \sim Beta(\alpha_1, \sum_{i=2}^{k} \alpha_i)$, and set $q_1 = u_1$. This is the first piece of the stick. The remaining piece has length $1 - u_i$.

Step 2: For $2 \le j \le k-1$, if $j-1$ pieces, with lengths $u_1, u_2, ..., u_{j-1}$, have been broken off, the length of the remaining stick is $\prod_{i=1}^{j-1}(1-u_i)$. We simulate $u_j \sim Beta(\alpha_i, \sum_{i=j+1}^{k} \alpha_i)$ and set $q_i = u_j \prod_{i=1}^{j-1}(1-u_i)$. The length of the remaining part of the stick is $\prod_{i=1}^{j-1}(1-u_i) - u_j \prod_{i=1}^{j-1}(1-u_i) = \prod_{i=1}^{j}(1-u_i)$.

Step 3: The length of the remaining piece is q_k.

Note that at each step, if $j-1$ pieces have been broken off, the remainder of the stick, with length $\prod_{i=1}^{j-1}(1-u_i)$, will eventually be broken up into $k-j+1$ pieces with proportions distributed according to a $Dir(\alpha_j, \alpha_{j+1}, ..., \alpha_k)$ distribution.

3. From Gamma RVs

The generating samples from the Dirichlet distribution using Gamma random variables is more computationally efficient that both the urn-drawing method and the stick-breaking method. This method has two steps:

Step 1: Generate gamma realization: for $i = 1, ..., k$, draw a number z_i from $\Gamma(\alpha_i, 1)$.

Step 2: Normalize them to form a pmf: for $i = 1, ..., k$, set $q_i = \frac{z_i}{\sum_{j=1}^{k} z_j}$. Then q is a realization of $Dir(\alpha)$. The Gamma distribution $\Gamma(k, \theta)$ is defined by the following probability density:

$$f(x; k, \theta) = x^{k-1} \frac{\exp \frac{-x}{\theta}}{\theta^{k]}\Gamma(k)} \tag{4}$$

where $k > 0$ is called the shape parameter, and $\theta > 0$ isc called the scale parameter.

References

[1] Thomas Hofmann. Probabilistic Latent Semantic Analysis

[2] S. Deerwester et al. Indexing by latent semantic analysis

[3] Seungjin Choi. *Lecture Note.*

[4] NYC Predictive Analytics. Introduction to probabilistic latent semantic analysis

[5] Chris Ding et al. On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing

[6] Bela A. Frigyik et al. Introduction to the Dirichlet Distribution and Related Processes, Department of Electrical Engineering University of Washington