Tensor PLSA

October 26, 2023

Model Specification

Tensor Probability Latent Semantic Analysis is a generalization of Probability Latent Semantic Analysis (PLSA). The latter is a tool for topic modelling. It models the joint distribution of two categorical variables D and W (document and word) as

$$P(D = d, W = w) = \sum_c P(C = c)P(D = d|C = c)P(W = w|C = c)$$

where C is the latent class (or aspect, topic). The number of different latent classes controls the complexity of the model. It has great interpretation. Each document is written under certain topic, and different topics have different word frequencies.

PLSA only has two observed variables (document and word). It is natural to extend this to contain more observed variables. For example, in shopping review, we could have the customer, the product as well as the words in review be mutually independent given the latent class of the review. For concreteness, consider the case when we have 3 observed variables

$$P(X = \mathbf{x}) = \sum_c P(C = c)P(X_1 = x_1)P(X_2 = x_2)P(X_3 = x_3)$$

This is equivalent to nonnegative tensor factorization just as PLSA is equivalent to nonnegative matrix factorization. Thus, it is fitting to call it tensor Probability Latent Semantic Analysis (tPLSA).

Estimation

The parameters of tPLSA can be estimated through EM algorithm, with E-step

$$P(C = c|X = \mathbf{x}) \propto P(C = c)\prod_{i=1}^{3} P(X_i = x_i|C = c)$$

And M-step

$$P(X_1 = x_1|C = c) \propto \sum_{u_2,u_3} M(X = (x_1, u_2, u_3))P(C = c|X = (x_1, u_2, u_3))$$

$$P(X_2 = x_2 | C = c) \propto \sum_{u_1, u_3} M(X = (u_1, x_2, u_3)) P(C = c | X = (u_1, x_2, u_3))$$

$$P(X_3 = x_3 | C = c) \propto \sum_{u_1, u_2} M(X = (u_1, u_2, x_3)) P(C = c | X = (u_1, u_2, x_3))$$

where $M(X = (x_1, x_2, x_3))$ means the count of co-occurrence corresponding to $X_1 = x_1$, $X_2 = x_2$, $X_3 = x_3$. In our shopping review example, this is the count of a specific word in the review from a customer to a certain product.

Comparison with Nonnegative Tensor Factorization

The main advantage over the usual nonnegative tensor factorization with multiplicative update[1] is that we do not have to calculate the whole probability tensor $P(C = c|X = \mathbf{x})$ explicitly. If $M(X = \mathbf{x}) = 0$, we do not have to do anything for the particular tensor element. In addition, at each time point, we only need to calculate $P(C|X)$ for one single value of X. This can lead to great speedup when the tensor is sparse, and saves memory in general. This can be seen in my wikipedia request-for-adminship example, where multiplicative update method from TensorLy leads to memory error while EM algorithm for tPLSA works as expected.

[1] Multiplicative update is usually for L_2 loss, while EM algorithm minimizes Kullbak Leibler divergence, so they are not exactly comparable. However, in practice, the difference may not be important.