

# On the Relationship between Probabilistic Semantic Analysis and Nonnegative Matrix

## Factorization

A Project for Data Mining

Martin Blom

Zack McCoy

May 8, 2007

## 1 Introduction

Recently, a number of papers have come out proposing and showing a fundamental relationship between Nonnegative Matrix Factorization (NMF) and Probabilistic Latent Semantic Analysis (pLSA) ([3, 2]). This relationship might not seem to surprising, factorizing a discrete and finite joint probability distribution and is fundamentally nothing different than factorizing a nonnegative matrix for which the sum of all the elements is 1. However, the two standard algorithms used for the different methods do *a priori* look nothing alike. NMF uses a multiplicative update rule and pLSA a Expectation-Maximization (EM) algorithm. The exact nature of the relationship turned out to be rather subtle.

In this report we investigate the exact nature of the relationship between pLSA and NMF, closely following Ref. [2]. We also look at how well pLSA and NMF do on a number of different data sets, and try to explain why some data sets seem to be more suited to pLSA/NMF than others.

## 2 The Underlying Algorithms

The setups of both NMF and pLSA are essentially the same. We are given a nonnegative matrix  $V$  and we want to explain this structure using a much simpler structure. The matrix  $V$  can represent many different kinds of data. The motivating context for pLSA is words and documents, and NMF has

been applied to a wide range of data: words and documents, facial recognition, DNA microarrays, and so forth. We will later apply both algorithms to the institution/conference data.

For the rest of this paper we will assume that  $V$  is normalized,  $V \leftarrow V/\|V\|_{L_1}$ . This does not make any difference in the initial discussion but the similarities between the two methods only holds under this normalization. This has to be the case, since pLSA deals with factorizations of *probabilities*, and therefore implicitly factorize matrices with norm 1.

## 2.1 NMF

NMF, as described by Lee and Seung in Ref. [5], aims to minimize

$$J_{\text{NMF}} = D(V, WH) \quad (1)$$

where  $D$  is some measure of "distance" and  $W, H$  are nonnegative matrices, usually of much lower rank than  $V$ . The distance measures used in Ref. [6] for  $J_{\text{NMF}}$  are the Euclidean distance,

$$D(V, WH) = \|V - WH\|^2 = \sum_{ij} (V_{ij} - (WH)_{ij})^2 \quad (2)$$

and the Kullback-Leibler divergence

$$D(V||WH) = \sum_{ij} \left( V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij} \right). \quad (3)$$

For the rest of this report we will concentrate on the KL divergence, but the convergence arguments proposed by Lee and Seung are the same for the Euclidean case.<sup>1</sup>

The essential idea behind Lee and Seungs algorithm is that of gradient descent with variable stepsize. Deriving Eq. 3 with respect to  $W_{ij}$  we obtain

$$\frac{dD}{dW_{ij}} = \sum_a H_{aj} \frac{V_{ia}}{(WH)_{ia}} - H_{aj}. \quad (4)$$

Gradient descent now gives us the update rule

$$W_{ij} \leftarrow W_{ij} + \eta \left( \sum_a H_{aj} \frac{V_{ia}}{(WH)_{ia}} - H_{aj} \right), \quad (5)$$

---

<sup>1</sup>In Ref. [1] it is shown that any Bregman divergence can be used. However, the Euclidean and the KL divergence are to our knowledge the only two measures widely used.

for some  $\eta$ . If  $\eta$  is chosen wisely, this should cause the cost function to decrease. Lee and Seung’s choice for  $\eta$  is

$$\eta_{ij} = \frac{W_{ij}}{\sum_a H_{aj}}. \quad (6)$$

A similar argument for  $H$  leaves us the update rule

$$W_{ia} \leftarrow W_{ia} \frac{\sum_\mu H_{a\mu} V_{i\mu} / (WH)_{i\mu}}{\sum_v H_{av}} \quad H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu} / (WH)_{i\mu}}{\sum_k W_{ka}} \quad (7)$$

This is *not* a relatively small value, so it might seem surprising that it actually consistently decreases the cost function. However, one can prove that the cost function decreases under the above update. The cost function is nondecreasing under the update 7 is *nondecreasing*.

The proof of this theorem is rather lengthy but not very complicated, and uses *auxiliary functions*, a technique commonly used with EM algorithms.

However, theorem 2.1 alone does *not* guarantee convergence of the algorithm.

## 2.2 Probabilistic Latent Semantic Analysis

Hofmann’s pLSA (Ref. [4]) is, as one might infer from its name, based on the idea that underlying the observed data (words and documents in the motivating example) are unobserved *class variables*  $z \in \mathcal{Z} = \{z_1, z_2, \dots, z_r\}$ . The assumption of pLSA is, given a certain class variable  $z$  there is a certain probability to generate the word  $w$ ,  $P(w|z)$ , and a certain probability to generate the document  $d$ ,  $P(d|z)$ .<sup>2</sup> The aim of pLSA is to find these probabilities, and the probability of choosing a class  $z$ ,  $P(z)$ , to begin with.

Two statistical assumptions have to be made in this framework. Firstly, observed pairs  $(w, d)$  are assumed to be generated *independently*. Secondly, we assume that given a class  $z$ , words are generated independently from the document. Given this framework, we seek to find probabilities  $P(z)$ ,  $P(w|z)$ , and  $P(d|z)$ , such that the log-likelihood function

$$J_{\text{pLSA}} = \sum_{i,j} V_{ij} \log P(w_i, d_j) = \sum_{i,j} V_{ij} \log \sum_{z \in \mathcal{Z}} P(z) P(w_i|z) P(d_j|z). \quad (8)$$

---

<sup>2</sup>Hofmann formulates the problem in a slightly different way, and considers the probabilities  $P(d)$ ,  $P(z|d)$  and  $P(w|z)$ . However, a simple application of Bayes’ rule translates his formulation into ours.

Hofmann does this by the Expectation Maximization (EM) algorithm. The EM algorithm alternates between a Expectation (E) step during which posterior probabilities are computed for the latent variables, and a Maximization (M) step in which the parameters are updated using the new posterior probability obtained in the update step.

Bayes' rule now gives the E-step

$$P(z|d, w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z'} P(z')P(d|z')P(w|z')}, \quad (9)$$

and maximizing  $J_{\text{pLSA}}$  given the above  $P(z|d, w)$  gives us the M-step

$$P(w_i|z) = \frac{\sum_j V_{ij}P(z|d_j, w_i)}{\sum_{j,k} V_{ij}P(z|d_j, w_k)}, \quad (10)$$

$$P(d_j|z) = \frac{\sum_i V_{ij}P(z|d_j, w_i)}{\sum_{k,i} V_{ij}P(z|d_k, w_i)}, \quad (11)$$

$$P(z) = \sum_{j,i} V_{ij}P(z|d_j, w_i). \quad (12)$$

Alternating the E and M step approaches a local maximum of the log-likelihood in the same way as alternately updating  $H$  and  $W$  approaches a local maximum of  $J_{\text{NMF}}$ .

In order to avoid overfitting, Hofmann also proposes a slightly different method, called Tempered Expectation Maximization (TEM). This is similar to ideas of *deterministic annealing*, and introduces an *inverse computational temperature*  $\beta$ . Intuitively, what Hofmann does is increasing the temperature to "shake" the solution out of local optima that are more coincidences of the training set than true trends in the data. The only implementational difference between TEM and EM is that the E-step is changed to

$$P(z|d, w) = \frac{P(z)[P(d|z)P(w|z)]^\beta}{\sum_{z'} P(z')[P(d|z')P(w|z')]^\beta}. \quad (13)$$

For our implementation we follow Hofmann and initialize  $\beta$  to one, run until convergence, then rescale  $\beta$  by a factor  $\eta < 1$ , run again until convergence, and iterate this until changing  $\beta$  no longer improves the result.

### 3 Equivalence of NMF and pLSA, and a Hybrid Algorithm

There are many reasons to suspect some kind of relationship between KL NMF and pLSA. Firstly, pLSA clearly gives a nonnegative factorization of the joint probability matrix  $V = \hat{P}(d, w)$ , so in some sense it concerns non-negative matrix factorizations. Secondly, already in Hofmann’s original paper it is observed that the log-likelihood can be seen as the Kullback-Leibler divergence from the empirical distribution of words in a document,  $\hat{P}(w|d)$ , and the reconstructed model distribution,  $P(w|d)$ . As Hofmann observes, for a fixed  $P(w|z)$  maximizing the log-likelihood thus amounts to projecting  $\hat{P}(w|d)$  on the subspace spanned by the factors, *along the geodesics given by the KL divergence*. In the same way, the update step in the NMF algorithm follows the gradient (geodesic) given by the KL divergence. However, it wasn’t until 2005 that Gaussier and Goutte made the first connection between NMF and pLSA. In Ref. [3] they claim that pLSA is NMF with KL divergence. This might be correct. Their proof is not though.

#### 3.1 Goutte and Gaussier’s argument

Let  $W'$  and  $H'$  be matrices such that

$$W'_{iz} = P(w_i|z)P(z) \quad H'_{zj} = P(d_j|z). \quad (14)$$

Further, for the NMF problem, let  $A_{kk} = \sum_i W_{ik}$  and  $B_{kk} = \sum_j H_{kj}$ . We then have

$$WH = (WA^{-1}A)(BB^{-1}H) = (WA^{-1})(AB)(B^{-1}H), \quad (15)$$

where  $(WA^{-1})$  and  $(B^{-1}H)$  can both be seen as conditional probabilities, since their rows and columns sum to one, and  $AB$  is a diagonal matrix, which can be interpreted as  $P(z)$ . Using this notation, we can combine the M-steps for  $P(z)$  and  $P(d|z)$  in pLSA to obtain (in the notation given by Eq. 14)

$$W'_{ia}{}^{(t+1)} \leftarrow W'_{ia}{}^{(t)} \frac{\sum_{\mu} H'_{a\mu}{}^{(t)} V_{i\mu} / (W'{}^{(t)} H'{}^{(t)})_{i\mu}}{\sum_{\nu} H'_{a\nu}{}^{(t)}} \quad (16)$$

$$H'_{a\mu}{}^{(t+1)} \leftarrow H'_{a\mu}{}^{(t)} \frac{\sum_i W'_{ia}{}^{(t)} V_{i\mu} / (W'{}^{(t)} H'{}^{(t)})_{i\mu}}{\sum_k W'_{ka}{}^{(t+1)}}. \quad (17)$$

This is clearly very similar to Eq. 7 and led Gaussier and Goutte to conclude that the algorithms are the same. However, this conclusion is far to strong

to draw just from this. What it *does* show is that a stationary point of pLSA is also a stationary point of NMF, since for a stationary point  $W^{(t+1)} = W^{(t)}$ , and so forth. That the stationary points of two cost functions are the same does clearly not imply that the cost functions themselves are the same.

### 3.2 Ding, Li and Peng’s argument

Even though Gaussier and Goutte’s argument is not strong as they claimed, it does point to a deep connection between pLSA and NMF. A stronger argument, namely that the cost functions for NMF and pLSA are the same, has been made by Ding, Li and Peng in Ref. [2], in which they propose the following theorem:

**Theorem (Ding, Li, Peng):** The objective function of pLSA is equivalent to the objective function of NMF with KL divergence,

$$\max J_{\text{pLSA}} \Leftrightarrow \min J_{\text{NMF}}. \quad (18)$$

*Ding, Li and Peng’s proof:* By definition, we have

$$J_{\text{pLSA}} = \sum_{i,j} V_{ij} \log P(w_i, d_j). \quad (19)$$

Subtracting a constant  $\sum_{i,j} V_{ij} \log V_{i,j}$  does clearly not change the optima of the objective function. Since pLSA is solved by maxima of  $J_{\text{pLSA}}$ , it is also equivalently solved by minima for

$$\sum_{i,j} V_{ij} \log \frac{V_{i,j}}{P(w_i, d_j)}. \quad (20)$$

Furthermore, they show that under the update in Eq. 7, the row sums and column sums of  $WH$  are the same as the row and column sums of  $V$ , *when  $W$  and  $H$  has converged to a stationary point*. This cancels those spurious terms in  $J_{\text{NMF}}$ , which shows that a minimum for  $J_{\text{NMF}}$  is a maximum for  $J_{\text{pLSA}}$  and vice versa. Note that this does *not* show that the objective functions of NMF and pLSA are the same. What it does show is that an optimal solution of NMF is also an optimal solution of pLSA, and vice versa. However, it is yet another strong indication that the two techniques are the same, even if the algorithms are not.

### 3.3 A Hybrid Algorithm

Whether pLSA and NMF are equivalent or not still seems inconclusive to us. What *is* clear, though, is that the two *algorithms* are different; indeed, they often reach different local optima even when starting from the same initial conditions. In Ref. [2], this fact is used to propose a clever hybrid algorithm, running one of the two till some sort of convergence has been reached, and then switching to the other and running that till convergence, and then switching back. It might seem like this would not produce new results since a fix point of one should also be a fix point of the other. Even so, the difference between the algorithms seem to be enough to often bump us out of local minima, which often leads to slight improvements in results. This can be thought of as similar to the  $k$ -means/local swapping procedure for clustering, where two different techniques are alternated to achieve better results than could be achieved with  $k$ -means alone. We have in the results below compared the hybrid algorithm with both pure NMF and pure pLSA.

## 4 Results

Running the function `main()` in the file `nmfplsa.py` will run NMF, the hybrid algorithm, the tempered hybrid algorithm, pLSA, and tempered pLSA, and return the resulting KL divergences and pictures of  $W$  and  $H$  for each of the 5 algorithms.

For example, we have the results of 4 tests with the CS Journal data, for 2, 3, 6, and 8 classes. The results our program shows are as follows:

**2 classes:** NMF; Time taken: 1.29700016975s. KL Divergence:0.431162396085  
Hybrid; Time taken: 2.67199993134s. KL Divergence:0.428620981765  
Tempered Hybrid; Time taken: 6.87500023842s. KL Divergence:0.429173279657  
EM; Time taken: 2.92199993134s. KL Divergence:0.502868258035  
Tempered EM; Time taken: 12.2970001698s. KL Divergence:0.476884009442

**3 classes:** NMF; Time taken: 2.0150001049s. KL Divergence:0.368554760791  
Hybrid; Time taken: 5.7349998951s. KL Divergence:0.362956874326  
Tempered Hybrid; Time taken: 42.2180001736s. KL Divergence:0.354976406885  
EM; Time taken: 6.09399986267s. KL Divergence:0.506120584199

Tempered EM; Time taken: 47.2810001373s. KL Divergence:0.414783089384

**6 classes:** NMF; Time taken: 2.51600003242s. KL Divergence:0.281134876751  
Hybrid; Time taken: 13.1559998989s. KL Divergence:0.238569399044  
Tempered Hybrid; Time taken: 30.6880002022s. KL Divergence:0.238431965941  
EM; Time taken: 13.8119997978s. KL Divergence:0.487582264925  
Tempered EM; Time taken: 33.3130002022s. KL Divergence:0.460050885156

**8 classes:** NMF; Time taken: 10.8440001011s. KL Divergence:0.191017053694  
Hybrid; Time taken: 26.7649998665s. KL Divergence:0.18690869239  
Tempered Hybrid; Time taken: 54.3600001335s. KL Divergence:0.188733736331  
EM; Time taken: 27.6239998341s. KL Divergence:0.476013077017  
Tempered EM; Time taken: 62.0320003033s. KL Divergence:0.406478500468

A quick glance at this data shows that NMF is the fastest algorithm as we've implemented it. Tempered EM is by far the slowest and takes more time and a stricter threshold to arrive at results similar to what we get with NMF. The Tempered Hybrid usually returns the best results overall. The interesting thing is that even when we have reached (or almost reached) a fixed point in NMF or pLSA, running the other doesn't immediately return a fixed point as Gaussier and Goutte expected.

The graphs and all the code are available on our project homepage at <http://www.ma.utexas.edu/users/zmccoy/plsinmf.html>.

## References

- [1] I.S. Dhillon and S. Sra. Generalized Nonnegative Matrix Approximations with Bregman Divergences. *Proceeding of the Neural Information Processing Systems (NIPS) Conference, Vancouver, BC, 2005*.
- [2] C. Ding, T. Li, and W. Peng. Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence, chi-square statistic, and a hybrid method. *Proc. of National Conf. on Artificial Intelligence (AAAI-06), 2006*.

- [3] E. Gaussier and C. Goutte. Relation between PLSA and NMF and implications. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602, 2005.
- [4] T. Hofmann. Probabilistic latent semantic indexing. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.
- [5] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- [6] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.