MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Shift-Invariant Probabilistic Latent Component Analysis

Paris Smaragdis, Bhiksha Raj

TR2007-009 December 2007

Abstract

In this paper we present a model which can decompose a probability densities or count data into a set of shift invariant components. We begin by introducing a regular latent variable model and subsequently extend it to deal with shift invariance in order to model more complex inputs. We develop an expectation maximization algorithm for estimating components and present various results on challenging real-world data. We show that this approach is a probabilistic generalization of well known algorithms such as Non-Negative Matrix Factorization and multi-way decompositions, and discuss its advantages over such approaches.

Shift-Invariant Probabilistic Latent Component Analysis

Paris Smaragdis PARIS@ADOBE.COM
Adobe Systems Inc.
Newton, MA 02466, USA

Bhiksha Raj BHIKSHA@MERL.COM
Mitsubishi Electric Research Laboratories
Cambridge, MA 02139, USA

Abstract

In this paper we present a model which can decompose a probability densities or count data into a set of shift invariant components. We begin by introducing a regular latent variable model and subsequently extend it to deal with shift invariance in order to model more complex inputs. We develop an expectation maximization algorithm for estimating components and present various results on challenging real-world data. We show that this approach is a probabilistic generalization of well known algorithms such as Non-Negative Matrix Factorization and multi-way decompositions, and discuss its advantages over such approaches.

Keywords: Convolutive bases, shift invariance, non-negative, factorization, latent decomposition, positive deconvolution

1. Introduction

Component-wise decompositions have long enjoyed a ubiquitous use in machine learning problems. It is customary to preprocess data using a component analysis in order to either perform dimensionality reduction, extract features, or discover something about the data's structure. Popular approaches such as Principal Component Analysis (PCA) and Independent Component Analysis (ICA) are frequently employed for various tasks such as feature discovery or object extraction and their statistical properties have made them indispensable tools for machine learning applications. More recently the introduction of Positive Matrix Factorization (12) (more prevalently known in the machine learning community as Non-Negative Matrix Factorization (5)) introduced a new desirable property in component decompositions, that of non-negativity. Non-negativity has proven to be a very valuable property for researchers working with positive only data and its use has since flourished especially for audio and image decompositions where non-negative representations are prevalent. For such representations a non-negative decomposition is advantageous as compared to alternatives like PCA or ICA, since it discovers non-negative elements which can be easily interpreted and are often perceptually meaningful decompositions. In contrast methods not using non-negativity are bound to discover bases that contain negative elements, and then employ cross-cancellation between them in order to approximate the input. Such components are hard to interpret in a positive only data framework and are often used for their statistical properties and not for the insight they provide. NMF and its subsequent mutations have been found to provide meaningful components on a variety of data types such as image (5) and audio magnitude spectra data (13). The downside of NMF however is that it is defined in a purely non-statistical framework which prohibits it to be applied in various probabilistic frameworks.

In this paper we will present a series of decompositions, very much alike the aforementioned techniques, that address the issues raised in the previous section. Our approach operates on probability densities, or count data, making it inherently non-negative and appropriate for a variety of applications where we observe either histograms, or energy signals. Due to the nature of the space we operate in, the results of the proposed decompositions can be used seamlessly in a learning environment due to the fact that all the derived variables can be linked to model likelihoods. This makes this approach ideal for decompositions which in later stages will feed into a classification, or clustering module where probabilistic interpretations of the inputs are strongly desired. In addition to providing this framework, we also show that this series of decompositions are numerically identical to NMF at their simplest level, but can easily surpass it by allowing decompositions of arbitrary dimensionality (tensor decompositions), as well as properties of shift-invariance.

The remainder of this paper is organized as follow. In section 2 we will cover the Probabilistic Latent Component Analysis (PLCA) which is the basic model which we will expand. In section 3 we will gradually introduce shift invariant extensions and in section 4 we will evaluate the analysis method in real-world audio and image data. We will conclude with a discussion section where we will describe relationships of our approach to past work and relative concepts.

2. Probabilistic Latent Component Analysis

In this section we present the model for Probabilistic Latent Component Analysis (PLCA). We will formulate it, present an Expectation-Maximization (EM) algorithm variant to compute the parameters of the model and present some simulations on toy problems.

The model that we will work with on this section is defined as:

$$P(\mathbf{x}) = \sum_z P(z) \prod_{j=1}^{N} P(x_j|z) \tag{1}$$

where $P(\mathbf{x})$ is an N-dimensional distribution of the random variable $\mathbf{x} = x_1, x_2, ..., x_N$. The z is a latent variable, and the $P(x_j|z)$ are one dimensional distributions. Effectively this model represents a mixture of marginal distribution products to approximate an N-dimensional distribution. The marginal distributions themselves are dependent on a latent variable z. The objective of this analysis is to find out the underlying structure of a probability distribution by using this latent model. This is done by estimating both $P(x_j|z)$ and $P(z)$ from an observed $P(\mathbf{x})$.

The estimation of the marginals $P(x_j|z)$ is performed using a variant of the EM algorithm which is described in more detail in appendix B. In short this algorithm contains an expectation and a maximization step which we alternate between in an iterative manner. In the expectation step we estimate the 'contribution' of the latent variable z:

$$R(\mathbf{x}, z) = \frac{P(z) \prod_{j=1}^{N} P(x_j|z)}{\sum_{z'} P(z') \prod_{j=1}^{N} P(x_j|z')} \tag{2}$$

and in a maximization step we re-estimate the marginals using the above weighting to obtain a new and more accurate estimate:

$$P(z) = \int P(\mathbf{x})R(\mathbf{x}, z)d\mathbf{x} \tag{3}$$

$$P(x_j|z) = \frac{\int \cdots \int P(\mathbf{x})R(\mathbf{x}, z)dx_k, \forall k \neq j}{P(z)} \tag{4}$$

$P(x_j|z)$ will contain a latent marginal distribution across the dimension of variable x_j, relating to the latent variable z, and $P(z)$ will contain the prior of that latent variable. Repeating the above steps in an alternating manner multiple times produces a converging solution for the marginals and the latent variable priors.

This above process can also be adapted to work for a discrete $\mathbf{x}$ and z (or all possible combinations), as described in appendix B. This process will also work if the provided input $P(\mathbf{x})$ is an un-normalized histogram as opposed to a density. The only added measure we need to take in this case is to normalize each $P(x_j|z)$ to integrate (or sum) to one in every iteration to ensure that it corresponds to a true marginal distribution.

We now illustrate the use of this decomposition using a toy problem. Assume that we observe a two dimensional random variable composed of three 2-d Gaussians with diagonal covariances:

$$\mathbf{x} \sim \frac{1}{2}\mathcal{N}\left(\begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}0.4 & 0\\0 & 0.4\end{bmatrix}\right) + \frac{1}{4}\mathcal{N}\left(\begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0.7 & 0\\0 & 0.1\end{bmatrix}\right) + \frac{1}{4}\mathcal{N}\left(\begin{bmatrix}-2\\1\end{bmatrix}, \begin{bmatrix}0.1 & 0\\0 & 0.4\end{bmatrix}\right) \tag{5}$$

The distribution $P(\mathbf{x})$ of this variable is shown in figure 1 in the lower left panel[1]. If we perform the analysis we've just described on this distribution we would hope to obtain three sets of marginals that describe each of the three Gaussians. To try this out we sample $P(\mathbf{x})$ and operate in the discrete domain using the discrete forms of the above equations as shown in appendix B (equations 36,37,38). The latent variable z is also discretized so as to assume only three values (one for each component we desire to extract).

Indeed after training we obtain the expected results. The results after 40 iterations are shown in the top panels in figure 1. The leftmost presents the priors $P(z_i)$, the middle one presents the marginals from the up-down dimension $P(x_2, |z_i)$, and the rightmost one the marginals from the left-right dimension $P(x_1, |z_i)$. The bottom right plot is the approximation to $P(\mathbf{x})$ using the PLCA model. By multiplying pairs of marginals that belong to the same latent variable we can see that we can describe all of the Gaussians that originally made up $\mathbf{x}$. The latent variable priors reflect the relative presence of each Gaussian, we can see that the prior values properly describe the ratio in the mixture (albeit normalized to sum to unity).

In the special case of a two-dimensional input $P(\mathbf{x})$, this technique is identical to PLSA (2), and can be shown to be numerically equivalent to NMF (see appendix 6 for proof). In fact one can discover a long trace of multiple equivalent mixture models originating early on from (4) (more information on related models is presented in the discussion section). Details on rank selection and convergence properties can be found in the literature relating

1. for readability and better printing, in most examples in this paper we use an inverse colormap in which higher values correspond to darker colors.

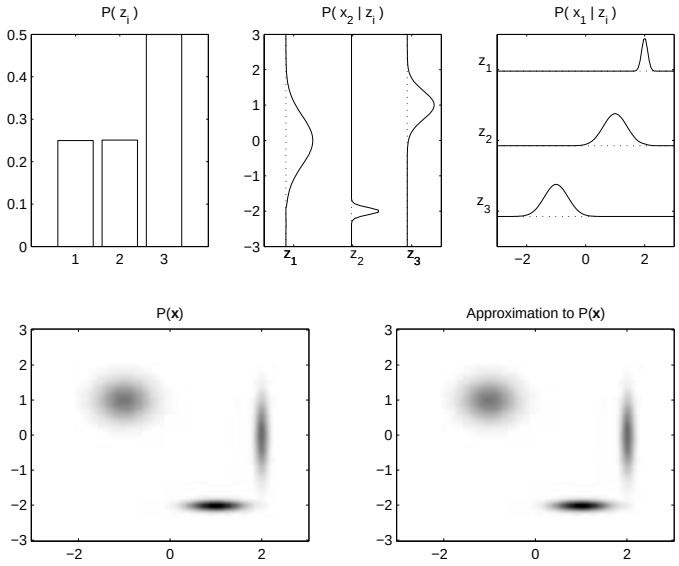

Figure 1: Toy example of PLCA. The 2-d distribution $P(\mathbf{x})$ in the lower left plot is composed out of three Gaussians. PLCA analysis discovers the relevant marginals and latent variable priors as shown in the top plots. We can see that the three Gaussians have been properly described by their marginals, and the latent priors reflect their proper mixing weights. The lower right plot shows the approximation of $P(\mathbf{x})$ using the weighted sum of the discovered marginal products.

to these techniques. We will simply mention that convergence is fairly rapid for inputs up to a few hundreds of dimensions and the results are not always identical, but they are for the vast majority qualitatively the same.

3. Shift Invariant Probabilistic Latent Component Analysis

In the previous section we described a simple model for decomposing probability distributions based on sums of products of marginals. As one might expect this is not a particularly powerful model when it comes to dealing with complex distributions, therefore in this section we extend it to deal with shift-invariance.

Although why such a behavior might be desired is not that clear at the moment, once this feature is implemented we will demonstrate its utility on real-world data and uncover a fundamental relationship with a seemingly unrelated family of algorithms.

3.1 Shift invariance across one dimension

We first consider the problem where we have a somewhat more complex description for each component than just a product of marginal distributions. Let us consider the case where we approximate a two-dimensional distribution using a set of left-right 'shifting' two-dimensional kernel distributions:

$$P(x,y) = \sum_z P(z) \int P(x,\tau|z)P(y-\tau|z)d\tau \tag{6}$$

What is effectively done is that instead of multiplying marginals we now convolve a "kernel" distribution $P(x,\tau|z)$ with an "impulse" distribution $P(y-\tau|z)$ along the left-right dimension. Note that for this model to make sense the variables y and τ have to be cardinal numbers so that the imposed shifting by the convolution operation is a meaningful operation. The kernel distributions are defined so that they extend over both dimensions, whereas the impulse distributions exist only on the dimension on which the convolution takes place. An optimal but uninteresting estimate for the kernel and impulse distributions is $P(z) = \delta(z)$, $P(y|z=0) = \delta(y)$, $P(x,\tau|z=0) = P(x,y)$. This solution merely returns to us the original distribution $P(x,y)$, without providing any further insight into its structure. To avoid this uninteresting solution and obtain more useful decompositions, we constrain the kernel to be zero outside $[\tau_1,\tau_2]$, i.e. we impose the constraint that $P(x,\tau|z) = 0\ \forall\tau \notin [\tau_1,\tau_2], \forall z$. Note that if $P(\tau|z) = \delta(\tau)$, equation (6) reduces to the original PLCA form in equation (1).

In this new setting our goal is still to estimate all three distributions (priors, kernels and impulses) in equation 6 given $P(x,y)$. In order to perform the estimation under this new model we need to appropriately update the existing PLCA algorithm to deal with the convolution operation. The appropriate extensions are described in detail in appendix C. One again we will use an EM variant as described before. The 'contribution' of each latent variable to be used in the expectation step is now defined over the parameter τ as well as the latent variable z and will now be:

$$R(x,y,\tau,z) = \frac{P(z)P(x,\tau|z)P(y-\tau|z)}{\sum_{z'} P(z') \int P(x,\tau|z')P(y-\tau|z')d\tau} \tag{7}$$

and the new estimates for $P(z)$, $P(x,\tau|z)$ and $P(y|z)$ will be naturally defined by the proper integrations over the input $P(x,y)$ weighted by the contribution $R(x,y,\tau,z)$:

$$P(z) = \iiint P(x,y)R(x,y,\tau,z)\ dx\ dy\ d\tau \tag{8}$$

$$P(x,\tau|z) = \frac{\int P(x,y)R(x,y,\tau,z)\ dy}{P(z)} \tag{9}$$

$$P(y|z) = \frac{\int\int P(x,y+\tau)R(x,y+\tau,\tau,z)dx\ d\tau}{\iiint P(x,y+\tau)R(x,y+\tau,\tau,z)\ dx\ dy\ d\tau} \tag{10}$$

Just as before the above equations are iteratively applied until the estimates for $P(z)$, $P(x,\tau|z)$ and $P(y|z)$ converge to a solution. Although the above equations assume that x,y and τ are continuous, this algorithm is also trivially adapted to discrete x,y and τ, as described in appendix C.

This particular model can allow us to deal with more complex inputs as demonstrated by the following example. Consider the distribution shown in the bottom left plot in figure 2. In this case we have two repeating patterns that compose the input distribution $P(x,y)$. One of them is a Gaussian pointing from bottom left to top right, and the other is a set of two Gaussians that form a wedge pointing towards the top left. Both of these patterns are

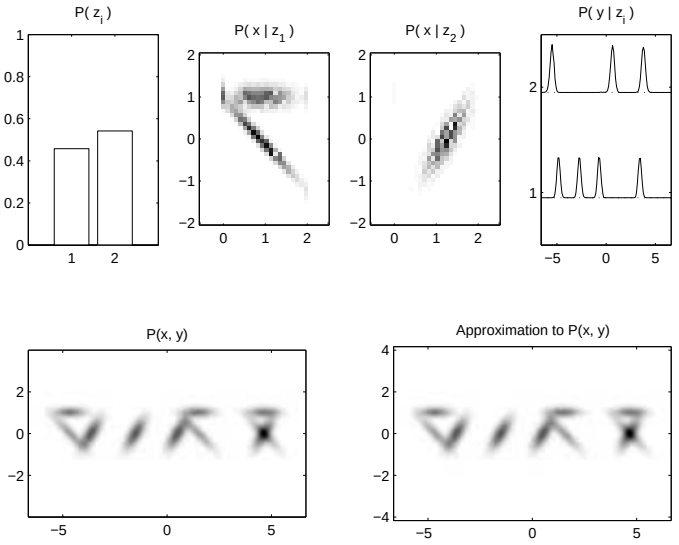

Figure 2: Example of left-right shift invariant decomposition using convolution. The top plots respectivelly display the latent variables priors, the kernel distributions and the impulse distributions that correspond to the input distribution in the lower left plot. The lower right plot displays the approximation of the model to the input. The extracted components have clearly captured the structure of the input.

not easily approximated by products of marginals as in the previous model. They can be modeled by the convolutive model since they exhibit repetition across the left-right axis.

We proceed to analyze this distribution using the discrete form of this algorithm (described in equations 58, 59, 60 and 60 in appendix C). In this particular example we limited the kernel distributions so that $P(x, \tau|z) = 0, \forall \tau < 0$ and $\tau > 2$. The latent variable z was also discretized and assumed only two values (z_1 and z_2). The results of the converged convolutive model are shown in figure 2. The top row of plots displays on the left the latent variable priors $P(z)$, the two top middle plots display the two kernel distributions $P(x, t|z)$ and $P(x, t|z)$, and the right top plot displays the impulse distributions $P(y|z)$. We can see that by convolving the pairs $P(x, t|z)$ with $P(y|z)$ we can model the input very well and also discover useful information about its structure.

Let us note at this point the importance of setting the kernel distribution $P(x, \tau|z)$ to be non-zero for only a limited interval of τ. If $P(x, \tau|z)$ were unconstrained, then a variety of other solutions, *e.g.* the previously mentioned $P(x, \tau|z) = P(x, y), \quad P(y|z) = \delta(y)$, may be obtained that may explain $P(x)$ better (in a KL sense) than the solutions obtained in the example. Other forms of partitioning $P(x, y)$ in $P(x, \tau|z)$ and setting $P(y|z)$ to be an appropriate assortment of delta functions would also provide an adequate, but uninformative solution. So like many dimensionality reduction schemes, the limiting of the extent of $P(x, \tau|z)$ serves as the bottleneck that forces the kernels to be informative.

3.2 Shift invariance across multiple dimensions

Having dealt with the case of shift invariance on one dimension, we now turn to shift invariance on multiple dimensions. Specifically, since in this paper we will be applying this model to two-dimensional real-world data such as images and spectrograms, we will present the case of shift invariance on both-dimensions of a two-dimensional distribution, whose two dimensions x and y we will designate the "left-right" and "up-down" dimensions respectively. For generalizations to an arbitrary number of dimensions, we refer the reader to appendix C.

We will now assume that the kernel distributions we derive can not only be shifted in the left-right dimension, but also in the up-down dimension. The model for this case is defined using a two-dimensional convolution as:

$$P(x,y) = \sum_z P(z) \iint P(\tau_x, \tau_y|z) P(x-\tau_x, y-\tau_y|z) d\tau_x d\tau_y \tag{11}$$

Note that now we will restrict the kernel distributions $P(\tau_x, \tau_y|z)$ such that $P(\tau_x, \tau_y|z) = 0 \; \forall (\tau_x, \tau_y) \notin \Re_{\tau_x,\tau_y}$, where $\Re_{\tau_x,\tau_y}$ defines a convex region. $\Re_{\tau_x,\tau_y}$ is chosen such that its extent is smaller than that of the input distribution $P(x,y)$, while the domain of the impulse distributions $P(x,y|z)$ is set to be as large as the input distribution, so that there is space to shift the kernel with respect to the impulse distribution in both dimensions.

A detailed derivation of this algorithm is presented in appendix C. The 'contribution' of each latent variable will now need to be over not only the latent variables, but also both the convolution parameters τ_x and τ_y:

$$R(x,y,\tau_x,\tau_y,z) = \frac{P(z)P(\tau_x,\tau_y|z)P(x-\tau_x,y-\tau_y|z)}{\sum_z P(z) \iint P(\tau_x,\tau_y|z)P(x-\tau_x,y-\tau_y|z)d\tau_x d\tau_y} \tag{12}$$

Like before, the estimation of the updated priors, kernel and impulse distributions can be done by the proper integrations:

$$P(z) = \iiiint P(x,y)R(x,y,\tau_x,\tau_y,z) \; dx \; dy \; d\tau_x \; d\tau_y \tag{13}$$

$$P(\tau_x,\tau_y|z) = \frac{\iint P(x,y)R(x,y,\tau_x,\tau_y,z) \; dx \; dy}{P(z)} \tag{14}$$

$$P_I(x,y|z) = \frac{\iint P(x+\tau_x,y+\tau_y)R(x+\tau_x,y+\tau_y,\tau_x,\tau_y,z) \; dx \; dy}{\iiiint P(x+\tau_x,y+\tau_y)R(x+\tau_x,y+\tau_y,\tau_x,\tau_y,z) \; dx \; dy \; d\tau_x \; d\tau_y} \tag{15}$$

The iterative algorithm above is guaranteed to obtain a locally optimal estimate for the kernel and impulse distributions. However, we have found it advantageous to utilize a modification, whereby at each iteration we "anneal" the kernel by raising it to an exponent:

$$P(\tau_x,\tau_y) \leftarrow c \cdot P(\tau_x,\tau_y)^\alpha \tag{16}$$

where $0 < \alpha \leq 1$ and c is a normalizing constant. α is initially set to a value less than 1, and as the iterations proceed, it is gradually raised to 1. While this procedure will not affect the algorithm's ability to obtain a locally optimal estimate of the kernel and

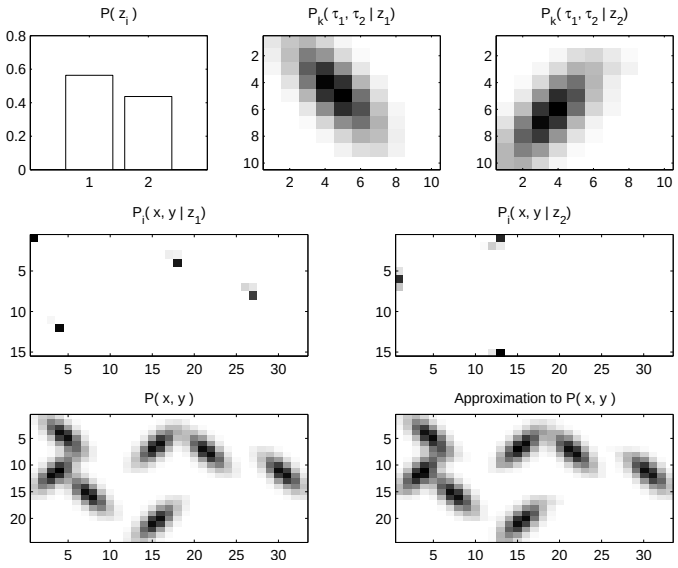

Figure 3: Example of a fully shift invariant decomposition using convolution. The top plots display the latent variable priors at the left most, whereas the remaining two display the two kernel distributions we extracted. The second row of plots display the impulse distributions, whereas the bottom row displays the original input distribution at the left and the model approximation at the right.

impulse distributions, we have empirically found that it is much more likely to result in "meaningful" decompositions, wherein the kernel captures most of the repetitive structure in the distribution while the impulse distribution chiefly consists of a mixture of impulse like peaks identifying the location of these repeating structures (thereby producing a sparse code). A more detailed look on this exponentiation step takes place later in the discussion section. As before convergence, for the vast majority of cases, results into qualitatively similar results over multiple runs and for most cases presented in this paper can be achieved in about a hundred iterations. The use of the annealing factor mentioned above, usually speeds up convergence and guarantees more uniform results. This technique can be shown to be almost the same as the convolutive NMF model (15), when we observe shifting of components into only one dimension. Details in convergence and rank selection can be found in the relevant literature.

We now present the results from a simple simulation that necessitates such a convolutive model. The input distribution $P(x, y)$ is shown as the bottom left plot in figure 3. The learned latent variable priors, kernel and impulse distributions are also shown in the same figure. Note that the kernel distributions have correctly converged to the two repeating forms, whereas the impulse distributions show us where these kernels are to be placed to perform the proper decomposition. Convolving each pair of kernel and impulse and summing them results in a good approximation of the input distribution.

4. Experiments on Real-World Data

In this section we present some results from real world data that highlight the ability of this algorithm to extract interesting features from complex inputs real-world. So far we considered the application of this work on probability distributions. However these algorithms are applicable as it in any form of count data, such as histograms, and energy or power measurements (all of which can be interpreted as a scaled distribution).

We present results from two domains, auditory and visual.

One might note the unorthodox approach of trying to perform EM-like training not based on the training data, but on their distribution or observed histogram. The reason this was done was to provide a way to analyze certain classes of highly complex distributions and obtain easily interpretable results. The class of distributions that we are most interested in dealing with are time-frequency distributions from audio data and intensity distributions from image data. In this section we present various experiments on such distributions and yield a variety of interesting results.

4.1 Audio data

We start by presenting some results using audio signals. To do so we will operate on a time-frequency distribution representation. This representation represents an audio signal by the distribution of its energy in the time and frequency axes. It is effectively a scaled histogram of sound atoms that fall in each time and frequency grid point.

We start with an example where only one kernel distribution is sought, albeit in multiple shifted positions. The example we picked to demonstrate this is a recording of a piano passage which included multiple notes sounding simultaneously. The nature of a music note is such that we usually have most signal energy at what is called the fundamental frequency (which also defines the pitch of the note played) and then decreasing amounts of energy on higher frequencies which are integer multiples of the fundamental (the so called harmonics of the sound). On an instrument like the piano it is safe to assume that neighboring notes have essentially the same energy distribution across frequencies albeit shifted along the frequency axis (assuming a logarithmic frequency spacing representation). In a time-frequency representation by playing different notes at different times we effectively have shifting in both the time axis denoting when a note was played, and the frequency axis denoting which note it was. In the bottom left plot of figure 4 we show a constant-Q (1) time-frequency distribution of the aforementioned piano passage. One can see the harmonic series repeating in various positions shifted in time (horizontal axis) and frequency (vertical axis) so as to represent each note. This was a discrete measurement sized 234 frequency bands by 317 time points. We analyzed this time-frequency distribution seeking a single latent variable. We defined the frequency axis dimension of the kernel distribution to be 180 frequencies long, and the time axis dimension to have a width of a single 'slice' of the input. The impulse distribution was responsible of placing that kernel distribution in the right places to reconstruct the input distribution. After training for about 100 iterations we obtained the results shown in figure 4 in the two top plots. The kernel distribution looks very much so like a harmonic series, whereas the impulse distribution has energy only at the place of the fundamental frequency, thereby noting where the kernels need to be placed to reconstruct the input. Thus we have, in an unsupervised manner, found out

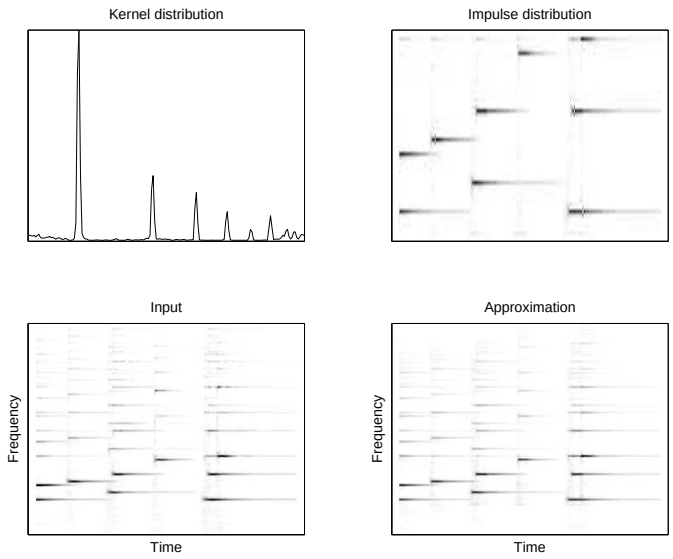

Figure 4: An example of discovering the shift-invariant structure in music. The bottom left plot displays a constant-Q time-frequency distribution of a piano notes passage. Analysis of this distribution results into a kernel function that is a harmonic series (top left), and an impulse function that places that harmonic series in the time-frequency plane (top right). The impulse function correctly identifies the time and frequency placement of all the notes played. The approximation to the input is shown in the bottom right plot.

that the piano recording was constructed by single harmonic template shifted appropriately in time and frequency. From this analysis we can define the timbre of a piano note (the kernel distribution) and also perform a transcription of the performance by noting the maxima of the impulse distribution. An interesting point to make is that although the harmonic structure is mostly obscured, due to constant note overlap in the passage, it is easily discovered by our analysis.

In the next example we will extract multiple kernel distributions from speech. We will analyze a magnitude spectrogram representation of male speech and see what type of kernel distributions compose it. For this example we used about 30 seconds of male speech from the TIMIT database and extracted 513 frequencies which resulted in a discretized input distribution of 513 frequencies by 938 time points. We looked for a latent variable with 20 states and defined the kernel distribution size to extend throughout all frequencies but only for 8 time points. This kernel size resulted into bases that only shifted in time but not in frequency (since both the kernel and the input had the same frequency width there was no space to shift along that dimension). The resulting kernel distributions are shown in figure 5. The speech researcher will be quick to recognize that the time-frequency form of these kernels resembles the structure of various phonemes. One can see a harmonic structure in each kernel distribution as well as a formant structure characteristic of a phoneme. Interestingly enough, and due to the additivity in this model, qualitatively similar results

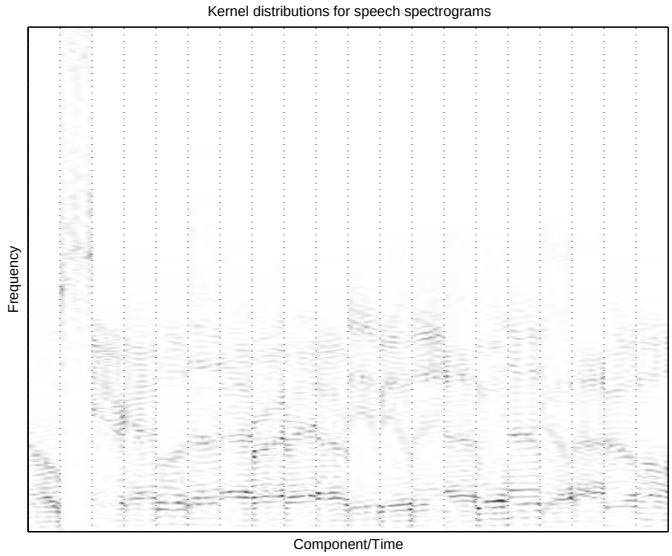

Figure 5: The resulting kernel distributions from analysis of the magnitude spectrogram of speech. The distributions are stacked from left to right and separated by a dotted line. One can see that their shape corresponds to magnitude spectrograms of various speech phonemes.

have been obtained when using mixtures of speakers as an input. In effect we find that the building blocks of speech are indeed phonemes placed in various parts in time. Analyzing different speaker types results in a different set of kernel distributions (phonemes) that reflects the unique nature of each speaker.

4.2 Image data

Image data can be thought of a distributions as well (the probability or count of photons landing on a particular point on an optical sensing grid), and can also be decomposed by our approach so that they yield interesting results. Like before we start with an example where we wish to extract a single kernel distribution. The input is a 136 by 511 pixel RGB color photograph of a choir shown in at the top of figure 6, which constitutes a $136 \times 511 \times 3$ input. The structure of the input is easy to see, we have a multitude of heads repeating at various positions along the picture. We analyzed the input looking for a single latent variable and kernel sized $35 \times 25 \times 3$ (35 by 25 pixels by 3 color channels). After analysis of the input we obtained the kernel distribution shown at the bottom left of the same figure, and the impulse distribution shown at the bottom right. One can see that the kernel distribution is shaped like the average head, maintaining the proper colors, and the impulse distribution is placing it at the appropriate points for each chorus singer.

A more complex example is shown with the data in figure 7. The bottom left plot shows a greyscale 81 by 113 pixel example of handwriting. Three characters are composing the picture and are positioned arbitrarily around the x,y grid. Analyzing the data we

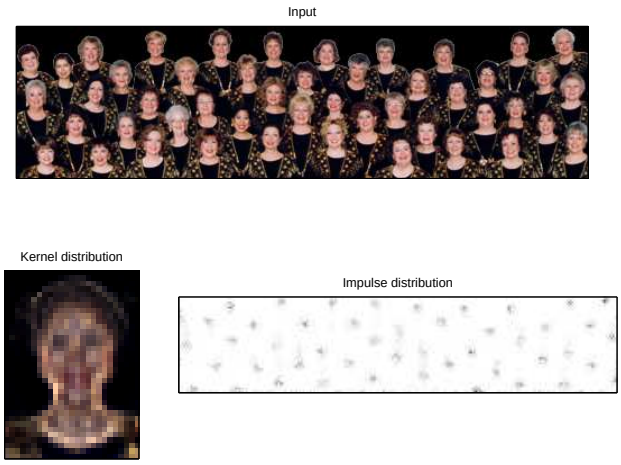

Figure 6: A picture of a chorus which exhibits many heads at different positions is shown at the top plot. The extracted kernel and impulse distributions are shown in the bottom left and right plots respectively.

extracted three 15 by 15 kernel distributions that are shown in the top left plot of the same figure. Note how the three kernel distributions are actually the three letters that make up the input. The impulse distribution contains spikes at the positions that correspond to each letter in the input. The approximation that results using this decomposition has the amusing property of streamlining the handwriting and making all instances of the same letters look more alike than in the original. The latent variable priors, shown under the kernel distributions, essentially tell us how much energy each letter consumes. The 'alpha', due to a more elaborate stroke, contains more energy from the 'gamma' which in turn contains somewhat more energy than the less elaborate 'one'.

Just as in the case of the audio data, we can obtain qualitatively the same results even when the characters overlap as long as they did not do so consistently.

5. Discussion

Of significant interest is the relationship of this type of analysis (at least in the basic form) to various existing methods. The non-convolutive form of PLCA is essentially a multi-variate generalization of Hofmann's bi-variate Probabilistic Latent Semantic Analysis (PLSA) (2). Through association with PLSA we can see how our method relates and enriches well known data mining approaches employing Latent Semantic Analysis (LSA), Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) as described by (2). More generally, the PLCA model may be related to latent class models (3) and latent structure analysis (4) which similarly decompose the probability distribution of unconditionally dependent variables into mixtures latent-variable-conditioned distributions of sets of variables. None of these earlier models, however, have been extended to incorporate shift invariance. Another family of algorithms that our approach is related to is the PARAFAC family (9).

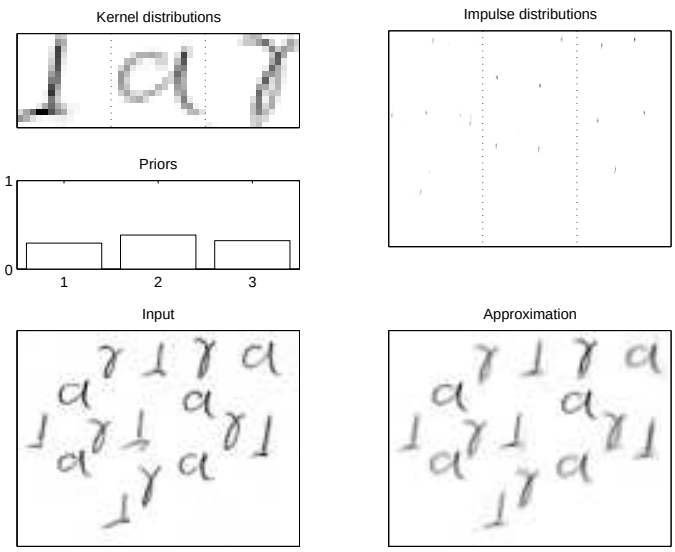

Figure 7: A handwriting example using three different characters is shown in the bottom left figure. The three extracted kernel distributions are shown in the top left plot, their priors right under them, and the three impulse distributions are shown in the top right plot. The resulting approximation is shown at the bottom right.

PARAFAC decompositions in general attempt to factor multilinear structures into vector components. This corresponds to PLCA for arbitrary dimensional distributions. The key difference here is that PARAFAC algorithms are predominantly least squares approximations to an arbitrary input, whereas our approach is explicitly approximating probability densities by marginals and has a probabilistic foundation. Certain versions of PARAFAC algorithms have been developed for non-negative inputs which brings this a step closer to to PLCA, but to our knowledge no probabilistic or convolutive framework has been proposed.

Another relative to this family of algorithms is the Non-Negative Matrix Factorization (NMF) algorithm (5). In NMF the objective is again to factor a non-negative matrix using two lower rank non-negative matrices. Effectively this performs a similar operation as PLCA for two dimensional distributions. An interesting connection of our work with NMF comes through with the cost function that is minimized when performing NMF. This function is most often an adaptation of the Kullback-Leibler divergence for arbitrary non-negative functions. This divergence is minimized between the input and the product of the estimated factors. Interestingly enough the training we propose for PLCA is essentially minimizing the KL divergence between the input probability density and the density specified product of marginals. One can see how the left and right factors in NMF correspond to $P(x_1|z)$ and $P(x_2|z)$ with $P(z)$ already factored in them. Even though the estimation algorithms for NMF and PLCA appear to be radically different, they can be shown to be numerically identical when NMF updates are derived using the KL-like objective function (see appendix 6). Subsequent convolutive extensions to NMF directly correspond to convolutive PLCA, in fact all results in (13) (14) (15) which use various forms of NMF can

be replicated using the algorithms described in this paper. Recent extensions of NMF have focused on tensor decompositions (18). In the context of PLCA this corresponds to an input distribution of more than two variables which is naturally handled in the training. An example of such a case is shown in figure 6. Although there are strong similarities to non-negative factorization approaches, we note that the particular approach we have chosen allows a better integration of this work into other machine learning frameworks and meta-learning models since it is described in a probabilistic setting as opposed to a setting of numerical approximation. Such extensions can incorporate latent Dirichlet allocation models, correlated topics, hierarchical structures and other ideas which are currently available in latent variable models.

It may be useful, at this point, to consider the generative model underlying both the simple and convolutive PLCA models. According to the simple PLCA model given by the right hand side of Equation 1, data are generated by a two-step drawing procedure. At each draw, first a latent variable z is drawn from the distribution $P(z)$, and in the second step each of the variables x_i is drawn from the conditional distribution $P(x_i|z)$. The set of x_i together form the observation $\mathbf{x}$. For the convolutive models the drawing procedure is more complex. At each draw a latent variable z is drawn from $P(z)$. $\hat{\mathbf{x}}$ is drawn from a set of conditional distributions $P(x_i|z)$. The components τ_i of a second vector $\boldsymbol{\tau}$ are drawn from the set of conditional distributions $P(\tau_i|z)$. There is a one-to-one correspondence between all components of $\boldsymbol{\tau}$ and some subset of components of $\hat{\mathbf{x}}$. The components of $\boldsymbol{\tau}$ are added to the corresponding components of $\hat{\mathbf{x}}$ to produce the final observation $\mathbf{x}$. Consequently, the final distribution of the observations is given by the weighted sum of the *convolution* of the conditional distributions of $\mathbf{x}$ and $\boldsymbol{\tau}$. The key point here is that for any given non-negative data, the generative process represented by the PLCA model actually draws instances of the *support* that the data lie on. The various component conditional distributions are also defined over the support. The data themselves, after due normalization, represent the overall distribution for the process. Thus, for an image the pixel positions (coordinates) represent the random variables that are drawn while pixel values represent the probabilities of the corresponding pixel positions.

Parallels may also be observed between our convolutive PLCA model and the "Transformed Component Analysis" (TCA) presented by Jojic et. al. (10; 11). TCA is a decomposition technique that extracts structural thematic blocks of data such as images in a manner that is invariant to transforms such as shifts. Although their approach also results in shift-invariant decompositions, there is a fundamental difference in the nature of the random variable considered by the TCA and PLCA models. For instance, when analysing an image, the RV considered by TCA is the vector of pixel *values*, that is transformed to derive observation. On the other hand, for convolutive PLCA the RVs are the pixel *positions* as explained above, and the pixel value at any position simply represents the probability that the corresponding pixel will be drawn. Another key difference between TCA and PLCA is that TCA analyzes *collections* of data to derive bases such that any specific data instance may be explained as a sum of tranformations of a number of bases. The bases obtained by the PLCA algorithm, on the other hand, identify repeating patterns *within a single* multidimensional data instance by analysing it. Thus the analyses obtained from the two algorithms are fundamentally different.

We should mention that our work also closely relates to positive deconvolution (8). This is a particularly desirable operation in the fields of astronomical imaging and bioinformatics. The objective is given a convolved sequence and a filter already applied on it obtain a non-negative deconvolution of the two. Various algorithms have been proposed mostly relying on least-squares formulation. Using our framework it is easy to adapt to this problem. Upon defining the filter to be a kernel distribution we can proceed by performing shift invariant PLCA only this time we will only update the impulse distribution and keep the kernel distribution fixed to the filter we wish to deconvolve with. Due to the lower number of variables to be estimated convergence is much more rapid than when performing a complete shift invariant PLCA.

An point worthy of some discussion in our approach is the exponentiation operation we described in section 3.2 which we use as a mechanism to ensure sparsity on the impulse distributions. Although we stopped short of a probabilistic explanation we note that this operation corresponds to information theoretic manipulations. The 'flattening' that the exponentiation produces causes the entropy of the kernel distributions to increase. Since the data we try to model has a fixed entropy, the increased kernel entropy is 'borrowed' from the impulse distributions. This forces the entropy of the impulse distributions to decrease which causes this form of sparse learning. We could alternatively raise the impulse distributions to a power greater than one to achieve similar results, however since the kernel distributions are in general smaller it is more efficient to manipulate them instead. This way of forcing sparsity in such a decomposition related to (7), although in our approach it is done in a probability space as opposed to a numerical approximation. It is somewhat straightforward to impose this flattening as an entropy constraint in the derivation of the presented algorithms. This can be done with the use of entropic priors (21). However the resulting process becomes much more complicated due to the necessary use of Lambert's W functions and additional iterative estimations. Treatment of these issues deems additional analysis that is out of the scope of this paper, details on such a constraint in this particular context are shown in (20) and (19). In practice we found the ad-hoc exponentiation to be an efficient compromise, at least in the context of the problems we analyzed.

Another issue that arises is the number and size of components that one might seek for. In most of the examples we presented in this paper we knew a priori how many components the scene was made of and roughly how big they should be. Estimating more components usually has the effect of distributing the desired answer into more components or allocating to them otherwise irrelevant information, thereby providing a more detailed description. Estimating fewer components results in either the non-detection of some desired components or a consolidation of multiple desired components into one. Asking for large-sized components results in some overfitting since there is little space to shift, whereas small-sized components end up being insufficient to model the input desirably. In general, as in many dimensionality reduction processes it is hard to reliably estimate how many and how large the components must be as an optimal choice, although the probabilistic setting allows the use of popular approaches such as the Schwarz-Bayesian information criterion and other similar measures. It is also possible to employ sparsity constraints on the priors of the components and effectively impose an automatic muting of redundant components. However this approach also requires an arbitrary selection of a sparsity parameter and merely transposes the problem of rank selection to that of sparsity bias selection.

6. Conclusion

In this paper we presented an analysis method that allows us to decompose probability distributions into shift invariant components. We presented our approach in gradually complicating cases starting from a simple static model to an arbitrary dimensionality and shift invariance model. We demonstrated how to derive an EM algorithm to perform the decomposition and presented various examples where such a decomposition can be useful. Finally we discussed how our approach relates to various other decomposition methods in order to provide some more insight, and show how this work can be applied to already known problems as a substitute.

Appendix A: The basic update rule

The update rules for PLCA are obtained through a variant of the expectation maximization algorithm. We attempt to estimate the parameters Λ of a model $P(\mathbf{x};\Lambda)$ for the distribution (or density) of a random variable $\mathbf{x}$, such that the KL divergence between $P(\mathbf{x};\Lambda)$ and the *true* distribution of $\mathbf{x}$, $P(\mathbf{x})$ is minimized. The KL divergence between the two is defined by:

$$D(P(\mathbf{x})||P(\mathbf{x};\Lambda)) = -E_{\mathbf{x}} \log P(\mathbf{x};\Lambda) - H(\mathbf{x}) \tag{17}$$

Here $E_{\mathbf{x}}$ refers to the expectation operator with respect to $P(\mathbf{x})$, the *true* distribution of $\mathbf{x}$. $H(\mathbf{x})$ is the entropy of $\mathbf{x}$.

Introducing a second random variable z, and by Bayes' rule

$$\log P(\mathbf{x};\Lambda) = \log P(\mathbf{x},z;\Lambda) - \log P(z|\mathbf{x};\Lambda)$$

Taking expectations on both side with respect to $P(z|\mathbf{x};\Lambda')$, i.e. the conditional probability of z obtained with any parameter Λ', and noting that $\log P(\mathbf{x};\Lambda)$ does not depend on z,

$$\log P(\mathbf{x};\Lambda) = E_{z|\mathbf{x};\Lambda'}\big\{\log P(\mathbf{x},z;\Lambda)\big\} - E_{z|\mathbf{x};\Lambda'}\big\{\log P(z|\mathbf{x};\Lambda)\big\} \tag{18}$$

Combining the above with Equation 17,

$$\begin{aligned} D(P(\mathbf{x})||P(\mathbf{x};\Lambda)) = &-E_{\mathbf{x}}\Big\{E_{z|\mathbf{x};\Lambda'}\big\{\log P(\mathbf{x},z;\Lambda)\big\}\Big\} + \\ &E_{\mathbf{x}}\Big\{E_{z|\mathbf{x};\Lambda'}\big\{\log P(z|\mathbf{x};\Lambda)\big\}\Big\} - H(\mathbf{x}) \end{aligned} \tag{19}$$

$$\begin{aligned} D(P(\mathbf{x})||P(\mathbf{x};\Lambda)) - D(P(\mathbf{x})||P(\mathbf{x};\Lambda')) = \\ E_{\mathbf{x}}\Big\{E_{z|\mathbf{x};\Lambda'}\big\{\log P(\mathbf{x},z;\Lambda')\big\} - E_{z|\mathbf{x};\Lambda'}\big\{\log P(\mathbf{x},z;\Lambda)\big\}\Big\} \\ - D(P(z|\mathbf{x};\Lambda')||P(z|\mathbf{x};\Lambda)) \end{aligned} \tag{20}$$

The KL divergence between two distributions is always non-negative (Theorem 2.6.3 in (17)), *i.e.* $D(P(z|\mathbf{x};\Lambda')||P(z|\mathbf{x};\Lambda)) \geq 0\ \forall\Lambda$. Hence, assuredly,

$$\begin{aligned} E_{\mathbf{x}}\Big\{E_{z|\mathbf{x};\Lambda'}\big\{\log P(\mathbf{x},z;\Lambda)\big\}\Big\} \geq E_{\mathbf{x}}\Big\{E_{z|\mathbf{x};\Lambda'}\big\{\log P(\mathbf{x},z;\Lambda')\big\}\Big\} \Rightarrow \\ D(P(\mathbf{x})||P(\mathbf{x};\Lambda)) \leq D(P(\mathbf{x})||P(\mathbf{x};\Lambda')) \end{aligned} \tag{21}$$

I.e., the distance between $P(\mathbf{x}|\Lambda)$ and $P(\mathbf{x})$ is assuredly lesser than or equal to the distance between $P(\mathbf{x})$ and $P(\mathbf{x}|\Lambda')$ if Λ minimizes $E_{\mathbf{x}}\big\{E_{z|\mathbf{x};\Lambda'}\{\log P(\mathbf{x},z;\Lambda)\}\big\}$. This leads to the following iterative update rule for the parameters of $P(\mathbf{x};\Lambda)$:

$$\begin{aligned} \Lambda^{(n+1)} &= \mathrm{argmax}_{\Lambda} \mathrm{Q}(\Lambda,\Lambda^{(\mathrm{n})}) \\ Q(\Lambda,\Lambda^{(n)}) &= E_{\mathbf{x}}\Big\{E_{z|\mathbf{x};\Lambda^{(n)}}\big\{\log P(\mathbf{x},z;\Lambda)\big\}\Big\} \end{aligned} \tag{22}$$

where $\Lambda^{(n)}$ is the estimate obtained for Λ in the n^{th} update. Iterations of Equation 22 will result estimates of Λ that will monotonically decrease $D(P(\mathbf{x})||P(\mathbf{x};\Lambda))$.

Appendix B: Update rules for non-convolutive mixture models

We define an "integral" operator $I_x\{f(x)\}$ such that for a continuous variable x $I_x\{f(x)\} = \int_{-\infty}^{\infty} f(x)dx$, while for a discrete random variable $I_x\{f(x)\} = \sum_x f(x)$. By the definition of the operator, $I_x\{I_y\{f(x,y)\}\} = I_y\{I_x\{f(x,y)\}\}$, $I_{\mathbf{x}}\{P(\mathbf{x})\} = 1$ and $I_{\mathbf{x}}\{P(\mathbf{x})g(\mathbf{x})\} = E_{\mathbf{x}}g(\mathbf{x})$.

A non-convolutive mixture model for the distribution of the data is

$$P(\mathbf{x};\Lambda) = I_z\Big\{P(z)\prod_j P(x_j|z)\Big\} \tag{23}$$

where $\mathbf{x} = \{x_j\}$. Note that the above formulation places no restriction on z, which might be either continuous or discrete. Similarly each x_j can be continuous or discrete. The parameters of this distribution are $P(z)$ and $P(x_j|z)$, i.e $\Lambda = \{P(z), P(x_j|z) : \forall(z,j)\}$. We will denote the estimates obtained in the n^{th} update by the superscript (n).

Let us define

$$R(\mathbf{x},z) \equiv P^{(n)}(z|\mathbf{x}) = \frac{P^{(n)}(z)\prod_j P^{(n)}(x_j|\mathbf{x})}{I_{z'}\Big\{P^{(n)}(z')\prod_j P^{(n)}(x_j|z')\Big\}} \tag{24}$$

We can now write

$$\begin{aligned} E_{z|\mathbf{x};\Lambda^{(n)}}\big\{\log P(\mathbf{x},z;\Lambda)\big\} &= I_z\Big\{R(\mathbf{x},z)\log\left(P(z)\prod_j P(z_j|\mathbf{x})\right)\Big\} \\ &= I_z\Big\{R(\mathbf{x},z)\left(\log P(z) + \sum_j \log P(x_j|z))\right)\Big\} \end{aligned} \tag{25}$$

The update equations are easily derived from Equations 22 and 25, with the additional incorporation of Lagrangian terms to enforce the constraints that the total probability masses under $P(z)$ and $P(x_j|z)$ are unity.

We can express the constrained form of the equation to be optimized as:

$$\Lambda^{(n+1)} = \text{argmax}_\Lambda \text{Q}(\Lambda,\Lambda^{(\text{n})}) \tag{26}$$

$$\begin{aligned} Q(\Lambda,\Lambda^{(n)}) &= E_{\mathbf{x}}\Big\{I_z\big\{R(\mathbf{x},z)\log P(z)\big\}\Big\} + E_{\mathbf{x}}\Big\{I_z\Big\{\sum_j R(\mathbf{x},z)\log P(x_j|z)\Big\}\Big\} \\ &\quad - \lambda I_z\Big\{P(z)\Big\} - I_z\Big\{\sum_j \lambda_{z,j} I_{x_j}\big\{P(x_j|z)\big\}\Big\} \\ &= I_z\Big\{E_{\mathbf{x}}\big\{R(\mathbf{x},z)\big\}\log P(z) - \lambda P(z)\Big\} \\ &\quad + I_z\Big\{\sum_j\Big\{E_{\mathbf{x}}\big\{R(\mathbf{x},z)\log P(x_j|z)\big\} - \lambda_{z,j} I_{x_j}\big\{P(x_j|z)\big\}\Big\}\Big\} \end{aligned} \tag{27}$$

We note that in general the optimization of $I_{\mathbf{x}}\{h(g(\mathbf{x}))\}$ with respect to $g(\mathbf{x})$ leads to $\frac{dh(g(\mathbf{x}))}{dg(\mathbf{x})} = 0$ both for discrete and continuous $\mathbf{x}$, by direct differentiation in the former case and by the calculus of variations in the latter.

The $(n+1)^{\text{th}}$ estimate of $P(z)$ is obtained by optimizing $Q(\Lambda, \Lambda^{(n)})$with respect to $P(z)$, which gives us

$$\begin{aligned} \frac{E_{\mathbf{x}}\{R(\mathbf{x},z)\}}{P^{(n+1)}(z)} - \lambda &= 0 \\ \lambda P^{(n+1)}(z) &= E_{\mathbf{x}}\{R(\mathbf{x},z)\} \end{aligned} \tag{28}$$

Since $I_z\{P^{(n+1)}(z)\} = 1$ and $I_z\{R(\mathbf{x},z)\} = 1$, applying the $I_z\{.\}$ operator to both sides of Equation 28 we get $\lambda = 1$, leading to the update equation

$$P^{(n+1)}(z) = E_{\mathbf{x}}\{R(\mathbf{x},z)\} \tag{29}$$

To derive the $(n+1)^{\text{th}}$ estimate of $P(x_j|z)$ we first note from reference (16) that

$$E_{\mathbf{x}}\big\{R(\mathbf{x},z)\log P(x_j|z)\big\} = E_{x_j}\big\{\log P(x_j|z) E_{\mathbf{x}|x_j}\{R(\mathbf{x},z)\}\big\} \tag{30}$$

We can therefore rewrite $Q(\Lambda, \Lambda^{(n)})$ as

$$\begin{aligned} Q(\Lambda,\Lambda^{(n)}) &= I_z\Big\{\sum_j \Big\{E_{x_j}\big\{\log P(x_j|z) E_{\mathbf{x}|x_j}\{R(\mathbf{x},z)\}\big\} - \lambda_{z,j} I_{x_j}\big\{P(x_j|z)\big\}\Big\}\Big\} + C \\ &= I_z\Big\{\sum_j \Big\{I_{x_j}\big\{P(x_j)\log P(x_j|z) E_{\mathbf{x}|x_j}\{R(\mathbf{x},z)\} - \lambda_{z,j} P(x_j|z)\big\}\Big\}\Big\} + C \end{aligned} \tag{31}$$

where C represents all terms that are not a function of $P(x_j|z)$. $P(x_j)$ is the *true* marginal density of x_j. Optimizing $Q(\Lambda, \Lambda^{(n)})$ with respect to $P^{(n+1)}(x_j|z)$, we obtain

$$\begin{aligned} \frac{P(x_j)E_{\mathbf{x}|x_j}\{R(\mathbf{x},z)\}}{P^{(n+1)}(x_j|z)} - \lambda_{z,j} &= 0 \\ P(x_j)E_{\mathbf{x}|x_j}\{R(\mathbf{x},z)\} &= \lambda_{z,j} P^{(n+1)}(x_j|z). \end{aligned} \tag{32}$$

Since $I_{x_j}\{P^{(n+1)}(x_j|z)\} = 1$, we can apply the $I_{x_j}\{.\}$ operator to both sides of the above equation to obtain

$$\begin{aligned} \lambda_{i,j} =& I_{x_j}\big\{P(x_j)E_{\mathbf{x}|x_j}\{R(\mathbf{x},z)\}\big\} = \\ & E_{x_j}\big\{E_{\mathbf{x}|x_j}\{R(\mathbf{x},z)\}\big\} = E_{\mathbf{x}}\{R(\mathbf{x},z)\} = P^{(n+1)}(z) \end{aligned} \tag{33}$$

and

$$P^{(n+1)}(x_j|z) = \frac{P(x_j)E_{\mathbf{x}|x_j}\{R(\mathbf{x},z)\}}{P^{(n+1)}(z)} = \frac{I_{\mathbf{x}/x_j}\big\{P(\mathbf{x})R(\mathbf{x},z)\big\}}{P^{(n+1)}(x)} \tag{34}$$

where $\mathbf{x}/x_j = \{x_i : i \neq j\}$ represents the set of all components of $\mathbf{x}$ *excluding* x_j. Equations 29 and 34 form the final update equations.

If z is a discrete random variable, the non-convolutive mixture model is given by

$$P(\mathbf{x};\Lambda) = \sum_z P(z)\prod_j P(x_j|z) \tag{35}$$

The update equations are given by

$$R(\mathbf{x}, z) \equiv \frac{P(z)\prod_j P(x_j|z)}{\sum_{z'} P(z')\prod_j P(x_j|z')} \tag{36}$$

$$P^{(n+1)}(z) = I_{\mathbf{x}}\big\{P(\mathbf{x})R(\mathbf{x}, z)\big\} \tag{37}$$

$$P^{(n+1)}(x_j|z) = \frac{I_{\mathbf{x}/x_j}\big\{P(\mathbf{x})R(\mathbf{x}, z)\big\}}{P^{(n+1)}(z)} \tag{38}$$

If $\mathbf{x}$ is a discrete random variable (i.e. every x_j is discrete), the specific form of the update rules (Equations 29 and 34) are:

$$P^{(n+1)}(z) = \sum_j \sum_{x_j} P(\mathbf{x})R(\mathbf{x}, z) \tag{39}$$

$$P^{(n+1)}(x_j|z) = \frac{\sum_{i:i\neq j}\sum_{x_i} P(\mathbf{x})R(\mathbf{x}, z)}{P^{(n+1)}(z)} \tag{40}$$

If $\mathbf{x}$ is a continuous random variable (i.e. every x_j is continuous), the update equations become:

$$P^{(n+1)}(z) = \int_{-\infty}^{\infty} P(\mathbf{x})R(\mathbf{x}, z)d\mathbf{x} \tag{41}$$

$$P^{(n+1)}(x_j|z) = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} P(\mathbf{x})R(\mathbf{x}, z)dx_1dx_2..dx_i \;\forall i \neq j}{P^{(n+1)}(z)} \tag{42}$$

Appendix C: Update rules for shift-invariant mixture models

The shift-invariant mixture model models the distribution of some dimensions of a multi-variate random variable as a convolution of a density kernel and a shift-invariant "impulse" density. As before, let $\mathbf{x}$ be the multi-variate random variable. Let $\mathbf{y}$ represent the set of components of $\mathbf{x}$ that are modelled in a shift-invariant manner, and $\mathbf{w}$ the rest of the components, i.e. $\mathbf{x} = \mathbf{w} \cup \mathbf{y}$ and $\mathbf{w} \cap \mathbf{y} = \phi$ (where ϕ represents the null set).

The shift-invariant model for the distribution of $\mathbf{x}$ models it as follows:

$$P(\mathbf{x}; \Lambda) = I_z\Big\{P(z)I_{\boldsymbol{\tau}}\big\{P(\mathbf{w}, \boldsymbol{\tau}|z)P(\mathbf{y} - \boldsymbol{\tau}|z)\big\}\Big\} \tag{43}$$

where $\boldsymbol{\tau}$ is a random variable that is defined over the same domain as $\mathbf{y}$. The terms to be estimated are $P(z), P(\mathbf{w}, \boldsymbol{\tau}|z)$ and $P(\mathbf{y}|z)$, i.e. $\Lambda = \{P(z), P(\mathbf{w}, \boldsymbol{\tau}|z), P(\mathbf{y}|z)\}$. Note that Equation 43 assumes that the random variable $\mathbf{y}$ is cardinal, irrespective of whether it is discrete or continuous. Also, as before, z may be either continuous or discrete.

Let us define

$$R(\mathbf{x}, \boldsymbol{\tau}, z) = R(\mathbf{w}, \mathbf{y}, \boldsymbol{\tau}, z) \equiv P^n(z, \boldsymbol{\tau}|\mathbf{x}) = \frac{P^{(n)}(z)P^{(n)}(\mathbf{w}, \boldsymbol{\tau}|z)P^{(n)}(\mathbf{y} - \boldsymbol{\tau}|z)}{I_{z'}\big\{P(z')I_{\boldsymbol{\tau}'}\{P(\mathbf{w}, \boldsymbol{\tau}'|z')P(\mathbf{y} - \boldsymbol{\tau}'|z')\}\big\}} \tag{44}$$

$$R(\mathbf{x}, z) \equiv I_{\boldsymbol{\tau}}\{R(\mathbf{w}, \mathbf{y}, \boldsymbol{\tau}, z)\} = P^{(n)}(z|\mathbf{x}) \tag{45}$$

The $(n+1)^{\text{th}}$ estimate of $P(z)$ is derived identically as in Appendix B and is given by

$$P^{(n+1)}(z) = E_{\mathbf{x}}\{R(\mathbf{x}, z)\} \tag{46}$$

To determine the update equations for $P(\mathbf{w}, \boldsymbol{\tau}|z)$ and $P(\mathbf{y}|z)$ we define the Q function to be optimized as

$$Q(\Lambda, \Lambda^{(n)}) = E_{z,\boldsymbol{\tau}|\mathbf{x};\Lambda^{(n)}}\Big\{ E_{\mathbf{x}}\big\{ \log P(\mathbf{x}, z, \boldsymbol{\tau}; \Lambda)\big\}\Big\} + C(\Lambda), \tag{47}$$

where $C(\Lambda)$ includes all constraint terms required to ensure that the estimated functions integrate to 1. To obtain the $(n+1)^{\text{th}}$ estimate of $P(\mathbf{y}|z)$, we use the following Q function (where all terms not related to $P(\mathbf{y}|z)$ are represented by the term D):

$$\begin{aligned}
Q(\Lambda, \Lambda^{(n)}) =& I_{z,\boldsymbol{\tau}}\Big\{ E_{\mathbf{y}}\big\{ \log P(\mathbf{y} - \boldsymbol{\tau}|z) E_{\mathbf{w}|\mathbf{y}}\{R(\mathbf{w}, \mathbf{y}, \boldsymbol{\tau}, z)\}\big\}\Big\} - I_z\Big\{\lambda_{\mathbf{y},z} I_{\mathbf{y}}\big\{P(\mathbf{y}|z)\big\}\Big\} + D \\
=& I_{z,\boldsymbol{\tau},\mathbf{y}}\big\{P(\mathbf{y}) \log P(\mathbf{y} - \boldsymbol{\tau}|z) E_{\mathbf{w}|\mathbf{y}}\{R(\mathbf{w}, \mathbf{y}, \boldsymbol{\tau}, z)\}\big\} - I_{z,\mathbf{y}}\big\{\lambda_{\mathbf{y},z} P(\mathbf{y}|z)\big\} + D \\
=& I_{z,\boldsymbol{\tau},\mathbf{y}}\big\{P(\mathbf{y} + \boldsymbol{\tau}) \log P(\mathbf{y}|z) E_{\mathbf{w}|\mathbf{y}}\{R(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\}\big\} - I_{z,\mathbf{y}}\big\{\lambda_{\mathbf{y},z} P(\mathbf{y}|z)\big\} + D \\
=& I_{z,\mathbf{y}}\Big\{ \log P(\mathbf{y}|z) I_{\boldsymbol{\tau}}\big\{P(\mathbf{y} + \boldsymbol{\tau}) E_{\mathbf{w}|\mathbf{y}}\{R(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\}\big\} - \lambda_{\mathbf{y},z} P(\mathbf{y}|z)\Big\} + D
\end{aligned} \tag{48}$$

Optimizing Equation 48 with respect to $P(\mathbf{y}|z)$ to obtain $P^{(n+1)}(\mathbf{y}|z)$, gives us

$$\begin{aligned}
\frac{I_{\boldsymbol{\tau}}\big\{P(\mathbf{y} + \boldsymbol{\tau}) E_{\mathbf{w}|\mathbf{y}}\{R(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\}\big\}}{P^{(n+1)}(\mathbf{y}|z)} - \lambda &= 0; \\
I_{\boldsymbol{\tau}}\big\{P(\mathbf{y} + \boldsymbol{\tau}) E_{\mathbf{w}|\mathbf{y}}\{R(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\}\big\} &= \lambda P^{(n+1)}(\mathbf{y}|z)
\end{aligned} \tag{49}$$

Since $I_{\mathbf{y}}\{P^{(n+1)}(\mathbf{y}|z)\} = 1$, applying the $I_{\mathbf{y}}(.)$ operator to both sides of the above equation, we get

$$\begin{aligned}
\lambda &= I_{\mathbf{y}}\Big\{ I_{\boldsymbol{\tau}}\big\{P(\mathbf{y} + \boldsymbol{\tau}) E_{\mathbf{w}|\mathbf{y}}\{R(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\}\big\}\Big\} \\
&= I_{\mathbf{y},\mathbf{w},\boldsymbol{\tau}}\Big\{P(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}) R(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\Big\}
\end{aligned} \tag{50}$$

and

$$\begin{aligned}
P^{(n+1)}(\mathbf{y}|z) &= \frac{I_{\boldsymbol{\tau}}\big\{P(\mathbf{y} + \boldsymbol{\tau}) E_{\mathbf{w}|\mathbf{y}}\{R(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\}\big\}}{I_{\mathbf{y}',\mathbf{w},\boldsymbol{\tau}}\Big\{P(\mathbf{w}, \mathbf{y}' + \boldsymbol{\tau}) R(\mathbf{w}, \mathbf{y}' + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\Big\}} \\
&= \frac{I_{\mathbf{w},\boldsymbol{\tau}}\big\{P(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}) R(\mathbf{w}, \mathbf{y} + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\big\}}{I_{\mathbf{y}',\mathbf{w},\boldsymbol{\tau}}\Big\{P(\mathbf{w}, \mathbf{y}' + \boldsymbol{\tau}) R(\mathbf{w}, \mathbf{y}' + \boldsymbol{\tau}, \boldsymbol{\tau}, z)\Big\}}
\end{aligned} \tag{51}$$

To obtain the $(n+1)^{\text{th}}$ estimate of $P(\mathbf{w}, \boldsymbol{\tau}|z)$, we use the following Q function:

$$\begin{aligned}
Q(\Lambda, \Lambda^{(n)}) =& I_{z,\boldsymbol{\tau}}\Big\{ E_{\mathbf{w}}\big\{ \log P(\mathbf{w}, \boldsymbol{\tau}|z) E_{\mathbf{y}|\mathbf{w}}\{R(\mathbf{w}, \mathbf{y}, \boldsymbol{\tau}, z)\}\big\}\Big\} - I_z\Big\{\lambda_{\boldsymbol{\tau},z} I_{\mathbf{w},\boldsymbol{\tau}}\big\{P(\mathbf{w}, \boldsymbol{\tau}|z)\big\}\Big\} + B \\
=& I_{z,\boldsymbol{\tau},\mathbf{w}}\Big\{P(\mathbf{w}) \log P(\mathbf{w}, \boldsymbol{\tau}|z) E_{\mathbf{y}|\mathbf{w}}\{R(\mathbf{w}, \mathbf{y}, \boldsymbol{\tau}, z)\} - \lambda_{\boldsymbol{\tau},z} P(\mathbf{w}, \boldsymbol{\tau}|z)\Big\} + B
\end{aligned} \tag{52}$$

where B represents all irrelevant terms. Optimizing $Q(\Lambda, \Lambda^{(n)})$ with respect to $P(\mathbf{w}, \boldsymbol{\tau}|z)$ to obtain $P^{(n+1)}(\mathbf{w}, \boldsymbol{\tau}|z)$ gives us:

$$P(\mathbf{w}) E_{\mathbf{y}|\mathbf{w}}\{R(\mathbf{w}, \mathbf{y}, \boldsymbol{\tau}, z)\} = \lambda_{\boldsymbol{\tau},z} P^{(n+1)}(\mathbf{w}, \boldsymbol{\tau}|z) \tag{53}$$

Since $I_{\mathbf{w},\boldsymbol{\tau}}\big\{P^{(n+1)}(\mathbf{w},\boldsymbol{\tau}|z)\big\} = 1$, we get

$$\begin{aligned}\lambda_{\boldsymbol{\tau},z} =& I_{\mathbf{w},\boldsymbol{\tau}}\big\{P(\mathbf{w})E_{\mathbf{y}|\mathbf{w}}\{R(\mathbf{w},\mathbf{y},\boldsymbol{\tau},z)\}\big\} = I_{\boldsymbol{\tau}}\Big\{E_{\mathbf{w}}\big\{E_{\mathbf{y}|\mathbf{w}}\{R(\mathbf{w},\mathbf{y},\boldsymbol{\tau},z)\}\big\}\Big\} \\ =& I_{\boldsymbol{\tau}}\big\{E_{\mathbf{x}}\{R(\mathbf{x},\boldsymbol{\tau},z)\}\big\} = E_{\mathbf{x}}\big\{I_{\boldsymbol{\tau}}\{R(\mathbf{x},\boldsymbol{\tau},z)\}\big\} = E_{\mathbf{x}}\{R(\mathbf{x},z)\} = P^{(n+1)}(z) \qquad (54)\end{aligned}$$

and

$$P^{(n+1)}(\mathbf{w},\boldsymbol{\tau}|z) = \frac{P(\mathbf{w})E_{\mathbf{y}|\mathbf{w}}\{R(\mathbf{w},\mathbf{y},\boldsymbol{\tau},z)\}}{P^{(n+1)}(z)} = \frac{I_{\mathbf{y}}\big\{P(\mathbf{x})R(\mathbf{x},\boldsymbol{\tau},z)\big\}}{P^{(n+1)}(z)} \qquad (55)$$

Equations 46, 51 and 55 form the final update equations.

Specific Case: Shift-invariance in one dimension

We now present the update rules for a shift-invariant mixture model of the joint distribution of two random variables x and y, $P(x,y)$, where only y is modelled in a shift invariant manner. z is assumed to be discrete. The mixture model is

$$P(x,y;\Lambda) = \sum_z P(z) I_t\big\{P(x,\tau|z)P(y-\tau|z)\big\} \qquad (56)$$

The update rules become:

$$\begin{aligned}R(x,y,\tau,z) &\equiv \frac{P^{(n)}(z)P^{(n)}(x,\tau|z)P^{(n)}(y-\tau|z)}{\sum_{z'} P^{(n)}(z') I_{\tau'}\big\{P^{(n)}(x,\tau'|z')P^{(n)}(y-\tau'|z')\big\}} && (57)\\ P^{(n+1)}(z) &= I_{x,y,\tau}\big\{P(x,y)R(x,y,\tau,z)\big\} && (58)\\ P^{(n+1)}(x,\tau|z) &= \frac{I_y\big\{P(x,y)R(x,y,\tau,z)\big\}}{P^{(n+1)}(z)} && (59)\\ P^{(n+1)}(y|z) &= \frac{I_{x,\tau}\big\{P(x,y+\tau)R(x,y+\tau,\tau,z)\big\}}{I_{y',x,\tau}\big\{P(x,y'+\tau)R(x,y'+\tau,\tau,z)\big\}} && (60)\end{aligned}$$

Specific Case: Shift-invariance in two dimensions

For the above case, both x and y can be modelled shift-invariantly as

$$P(x,y;\Lambda) = \sum_z P(z) I_{\tau_1,\tau_2}\big\{P(\tau_1,\tau_2|z)P(x-\tau_1,y-\tau_2|z)\big\} \qquad (61)$$

The update rules become:

$$\begin{aligned}R(x,y,\tau_1,\tau_2,z) &\equiv \frac{P^{(n)}(z)P^{(n)}(\tau_1,\tau_2|z)P^{(n)}(x-\tau_1,y-\tau_2|z)}{\sum_{z'} P(z') I_{\tau_1',\tau_2'}\big\{P(\tau_1',\tau_2'|z)P(x-\tau_1',y-\tau_2'|z)\big\}} && (62)\\ P^{(n+1)}(z) &= I_{x,y,\tau_1,\tau_2}\big\{P(x,y)R(x,y,\tau_1,\tau_2,z)\big\} && (63)\\ P^{(n+1)}(\tau_1,\tau_2|z) &= \frac{I_{x,y}\big\{P(x,y)R(x,y,\tau_1,\tau_2,z)\big\}}{P^{(n+1)}(z)} && (64)\\ P^{(n+1)}(x,y|z) &= \frac{I_{\tau_1,\tau_2}\big\{P(x+\tau_1,y+\tau_2)R(x+\tau_1,y+\tau_2,\tau_1,\tau_2,z)\big\}}{I_{x',y',\tau_1,\tau_2}\big\{P(x'+\tau_1,y'+\tau_2)R(x'+\tau_1,y'+\tau_2,\tau_1,\tau_2,z)\big\}} && (65)\end{aligned}$$

Appendix D: Equivalence between PLCA and NMF

In this appendix we show how the training algorithms for Probabilistic Latent Component Analysis (PLCA) and Non-Negative Matrix Factorization (NMF) using a KL-metric are numerically identical. Since NMF is specifically defined for non-negative matrices, we will consider the case of PLCA decomposition of a bivariate distribution $P(x, y)$ over two discrete random variables x and y, both of which take a finite number of values such that the set of all (x, y) pairs can be represented as a matrix. The PLCA model for the distribution is given by $P(x, y) = \sum_z P(z)P(x|z)P(y|z)$.

The update rules for the estimation of $P(z)$, $P(x|z)$ and $P(y|z)$ are obtained for Equations 36, 39 and 40 as:

$$
\begin{aligned}
R(x, y, z) &= \frac{P^{(n)}(z)P^{(n)}(x|z)P^{(n)}(y|z)}{\sum_z P^{(n)}(z)P^{(n)}(x|z)P^{(n)}(y|z)} && (66)\\
P^{(n+1)}(z) &= \sum_{x,y} P(x, y)R(x, y, z) && (67)\\
P^{(n+1)}(y|z) &= \frac{\sum_x P(x, y)R(x, y, z)}{\sum_{x,y} P(x, y)R(x, y, z)} && (68)\\
P^{(n+1)}(x|z) &= \frac{\sum_y P(x, y)R(x, y, z)}{\sum_{x,y} P(x, y)R(x, y, z)} && (69)
\end{aligned}
$$

where the superscript n refers to estimates obtained in the n^{th} iteration of the algorithm.

From Bayes' rule $P(y, z) = P(z)P(y|z)$. We also have $\sum_z P^{(n)}(z)P^{(n)}(x|z)P^{(n)}(y|z) = \sum_z P^{(n)}(x|z)P^{(n)}(y, z) = P^{(n)}(x, y)$, where $P^{(n)}(x, y)$ is the approximation obtained to $P(x, y)$ in the n^{th} iteration. So we can rearrange the update rules in terms of $P(x|z)$ and $P(y, z)$ and $P^{(n)}(x, y)$ as:

$$
\begin{aligned}
R(x, y, z) &= \frac{P^{(n)}(x|z)P^{(n)}(y, z)}{P^{(n)}(x, y)} && (70)\\
P^{(n+1)}(y, z) &= \sum_x P(x, y)R(x, y, z) && (71)\\
P^{(n+1)}(x|z) &= \frac{\sum_y P(x, y)R(x, y, z)}{\sum_y P^{(n+1)}(y, z)} && (72)\\
P^{(n+1)}(z) &= \sum_y P(y, z) && (73)\\
P^{(n+1)}(y|z) &= \frac{P^{(n)}(y, z)}{P^{(n)}(z)} && (74)
\end{aligned}
$$

Note that the above update rules are identical to the update rules in Equations 66-69, except that we now utilize $P(y, z)$ as an intermediate variable. The final estimates for $P(z), P(x|z)$ and $P(y|z)$ obtained from Equations 66-69 would be numerically identical to those obtained from Equation 70-74.

In Equations 72 and 71 the update for $P(x|z)$ is computed in terms of $R(x,y,z)$. Explicitly expanding $R(x,y,z)$ in the update rules for $P(x|z)$ and $P(y,z)$ we can write

$$\begin{aligned} P^{(n+1)}(y,z) &= \sum_x P(x,y)\frac{P^{(n)}(x|z)P^{(n)}(y,z)}{P^{(n)}(x,y)} \\ &= P^{(n)}(y,z)\sum_x \frac{P(x,y)}{P^{(n)}(x,y)}P^{(n)}(x|z) \qquad (75) \\ P^{(n+1)}(x|z) &= \frac{\sum_y P(x,y)\frac{P^{(n)}(x|z)P^{(n)}(y,z)}{P^{(n)}(x,y)}}{\sum_y P^{(n+1)}(y,z)} \\ &= \frac{P^{(n)}(x|z)\sum_y \frac{P(x,y)}{P^{(n)}(x,y)}P^{(n)}(y,z)}{\sum_y P^{(n+1)}(y,z)} \qquad (76) \end{aligned}$$

The above two equations represent the complete PLCA update rule, since $P^{(n+1)}(z)$ and $P^{(n+1)}(y|z)$ are derived from $P^{(n+1)}(y,z)$.

We can write the above in matrix notation. Let $\mathbf{V}$ represent the matrix of probability values $P(x,y)$. Let $\mathbf{W}_n$ represent the matrix of probability values $P^{(n)}(x|z)$, such that $P^{(n)}(x|z)$ is the x^{th} row of the z^{th} column of $\mathbf{W}_n$. Note that the columns of $\mathbf{W}_n$ sum to 1.0. Similarly, let $\mathbf{V}_n$ represent the matrix of $P^{(n)}(x,y)$ values and $\mathbf{H}_n^T$ represent the matrix of $P^{(n)}(y,z)$ values, such that $P^{(n)}(y,z)$ is the z^{th} row of the y^{th} column of $\mathbf{H}_n$. We thus have $\mathbf{V}_n = \mathbf{W}_n\mathbf{H}_n$. Equations 75 and 76 can now be written in matrix form as

$$\begin{aligned} \mathbf{H}_{n+1} &= \mathbf{H}_n \odot {\mathbf{W}_n}^T\frac{\mathbf{V}}{\mathbf{V}_n} \qquad (77) \\ \mathbf{W}_{n+1} &= \left(\mathbf{W}_n \odot \frac{\mathbf{V}}{\mathbf{V}_n}{\mathbf{H}_n}^T\right)\left(diag(\mathbf{H}_{n+1}\mathbf{1})\right)^{-1} \qquad (78) \end{aligned}$$

where $\odot$ represents Hadamard (component-wise) multiplication and the division is also component-wise. $\mathbf{1}$ represents a column vector of ones such that $\mathbf{H}_{n+1}\mathbf{1}$ represents the column vector derived by summing the rows of $\mathbf{H}_{n+1}$. The $diag(\cdot)$ operator constructs a diagonal matrix comprising the components of its vector argument.

We note that the columns of $\mathbf{W}_n$ sum to 1.0. Hence, $diag(\mathbf{1}^T\mathbf{W}_n)$ is an identity matrix and post multiplying $\mathbf{W}_n$ by its inverse will not change its value. Correspondingly, Equations 79 and 80 can be rewritten as:

$$\begin{aligned} \mathbf{H}_{n+1} &= \left(diag(\mathbf{1}^T\mathbf{W}_n)\right)^{-1}\left(\mathbf{H}_n \odot {\mathbf{W}_n}^T\frac{\mathbf{V}}{\mathbf{V}_n}\right) \qquad (79) \\ \mathbf{W}_{n+1} &= \left(\mathbf{W}_n \odot \frac{\mathbf{V}}{\mathbf{V}_n}{\mathbf{H}_n}^T\right)\left(diag(\mathbf{H}_{n+1}\mathbf{1})\right)^{-1} \qquad (80) \end{aligned}$$

Once the updates above converge to a final estimate $\mathbf{V} \approx \mathbf{WH}$, the final estimates of $P(x|z)$ are obtained as the entries of the matrix $\mathbf{W}$, $P(z)$ values are obtained as the

entries of $\mathbf{H1}$ (the elements of which represent the terms $\sum_y P(y,z)$), and $P(y|z)$ terms are obtained as the entries of $\left(diag(\mathbf{H1})\right)^{-1}\mathbf{H}$.

The astute reader would realize that Equations 79 and 80 are identical to NMF update rules given by Equations 5 in reference (6). We have thus shown that one may utilize NMF update rules to perform PLSI and obtain numerically identical results to those obtained by the PLSI rules of Equations 66-69. The only requirements here are that the matrices $\mathbf{V}$ and $\mathbf{H}_0$ (the initial estimate of $\mathbf{H}$) must represent a distribution, *i.e.*, their elements must sum to 1.0, and the columns of $\mathbf{W}_0$ (the initial estimate of $\mathbf{W}$) must also sum to 1.0.

More generally, in NMF $\mathbf{H}_0$ and $\mathbf{W}_0$ are *not* initialized to conform to the requirements of PLSI. Nevertheless, as we show below, the iterations of NMF update rules can still be used to obtain PLSI decompositions that are numerically identical to those obtained through the EM solution of Equations 66-69. Let $\mathbf{W}_n$ and $\mathbf{H}_n$ represent the estimate of $\mathbf{W}$ and $\mathbf{H}$ respectively, obtained in the n^{th} iteration of NMF updates. We define the normalizing constants

$$\begin{aligned} \mathbf{D} &= diag(\mathbf{1}^T\mathbf{W}_0) && (81)\\ c &= \mathbf{1}^T\mathbf{D}\mathbf{H}\mathbf{1} && (82) \end{aligned}$$

c is simply the sum of all the elements of the matrix $\mathbf{DH}$. Using these terms we can now define *scaled* versions of $\mathbf{W}_0$ and $\mathbf{H}_0$ as:

$$\begin{aligned} \bar{\mathbf{W}}_0 &= \mathbf{W}_0\mathbf{D}^{-1} && (83)\\ \mathbf{W}_0 &= \bar{\mathbf{W}}_0\mathbf{D} && (84)\\ \bar{\mathbf{H}}_0 &= \mathbf{D}\mathbf{H}_0c^{-1} && (85)\\ \mathbf{H}_0 &= c\mathbf{D}^{-1}\bar{\mathbf{H}}_0 && (86) \end{aligned}$$

$\bar{\mathbf{H}}_0$ and $\bar{\mathbf{W}}_0$ conform to the requirements of PLSI, *i.e.*, the sum of all elements of $\mathbf{H}_0$ sum to 1.0, while the columns of $\bar{\mathbf{W}}_0$ sum to 1.0. Using the above normalizations, we get $\mathbf{W}_0\mathbf{H}_0 = c\bar{\mathbf{W}}_0\bar{\mathbf{H}}_0$.

As shown earlier, initializing $\mathbf{W}$ and $\mathbf{H}$ with $\bar{\mathbf{W}}_0$ and $\bar{\mathbf{H}}_0$ respectively in the update rules of Equations 79 and 80 will give us PLSI estimates of $\mathbf{W}$ and $\mathbf{H}$ (*i.e.*, of $P(x|z)$ and $P(y,z)$). On the other hand, initializing the update rules directly with $\mathbf{W}_0$ and $\mathbf{H}_0$ will give us NMF updates.

Let us represent the estimates of $\mathbf{W}$ and $\mathbf{H}$ obtained in the n^{th} PLSI iteration as $\bar{\mathbf{W}}_n$ and $\bar{\mathbf{H}}_n$ respectively. Let us represent the estimates obtained in the n^{th} iteration of the NMF updates as $\mathbf{W}_n$ and $\mathbf{H}_n$ respectively. Prior to proceeding, we note that for the Hadamard multiplication operator $\odot$ and for any diagonal matrix $\mathbf{D}$, $\mathbf{DW} \odot \mathbf{D}^{-1}\mathbf{H} = \mathbf{WH}$ and similarly $\mathbf{WD} \odot \mathbf{HD}^{-1} = \mathbf{WH}$. For $n = 1$ we get

$$\mathbf{H}_1 = \left(diag(\mathbf{1}^T\mathbf{W}_0)\right)^{-1}\left(\mathbf{H}_0 \odot {\mathbf{W}_0}^T\frac{\mathbf{V}}{\mathbf{W}_0\mathbf{H}_0}\right) \quad (87)$$

$$= \mathbf{D}^{-1}\left(c\mathbf{D}^{-1}\bar{\mathbf{H}}_0 \odot \mathbf{D}\bar{\mathbf{W}}_0^T\frac{\mathbf{V}}{c\bar{\mathbf{W}}_0\bar{\mathbf{H}}_0}\right) \quad (88)$$

$$= \mathbf{D}^{-1}\left(\bar{\mathbf{H}}_0 \odot \bar{\mathbf{W}}_0^T\frac{\mathbf{V}}{\bar{\mathbf{W}}_0\bar{\mathbf{H}}_0}\right) \quad (89)$$

$$\mathbf{H}_1 = \mathbf{D}^{-1}\bar{\mathbf{H}}_1 \quad (90)$$

In the above we have utilized the PLSI update rule of Equation 77 (which is identical to that of Equation 79 since $diag(\mathbf{1}^T\bar{\mathbf{W}}_0)$ is an identity matrix). Similarly, we find that the corresponding NMF update rule for $\mathbf{W}_1$ can be written as

$$\mathbf{W}_1 = \left(\mathbf{W}_0 \odot \frac{\mathbf{V}}{\mathbf{W}_0\mathbf{H}_0}{\mathbf{H}_0}^T\right)\left(diag(\mathbf{H}_1\mathbf{1})\right)^{-1} \quad (91)$$

$$= \left(\bar{\mathbf{W}}_0\mathbf{D} \odot \frac{\mathbf{V}}{c\bar{\mathbf{W}}_0\bar{\mathbf{H}}_0}c{\bar{\mathbf{H}}_0}^T\mathbf{D}^{-1}\right)\left(\mathbf{D}^{-1}diag(\bar{\mathbf{H}}_1\mathbf{1})\right)^{-1} \quad (92)$$

$$= \left(\bar{\mathbf{W}}_0 \odot \frac{\mathbf{V}}{\bar{\mathbf{W}}_0\bar{\mathbf{H}}_0}\bar{\mathbf{H}}_0^T\right)\left(diag(\bar{\mathbf{H}}_1\mathbf{1})\right)^{-1}\mathbf{D} \quad (93)$$

$$\mathbf{W}_1 = \bar{\mathbf{W}}_1\mathbf{D} \quad (94)$$

Here we have utilized the fact that $diag(\mathbf{D}^{-1}\mathbf{H}_1\mathbf{1}) = \mathbf{D}^{-1}diag(\mathbf{H}_1\mathbf{1})$. Note that the constant c no longer figures in either Equation 90 or Equation 94. It is now straightforward to extend the above development to generalize the relationship between PLCA and NMF to:

$$\mathbf{W}_n = \bar{\mathbf{W}}_n\left(diag(\mathbf{1}^T\mathbf{W}_0)\right) \quad (95)$$

$$\mathbf{H}_n = \left(diag(\mathbf{1}^T\mathbf{W}_0)\right)^{-1}\bar{\mathbf{H}}_n \quad (96)$$

In the above equations we have used the expanded form of $\mathbf{D}$ from Equation 81.

From the above it is clear that NMF update rules may be used in the place of the PLCA update rules of Equations 66-69, and numerically identical results may be obtained by scaling the final $\mathbf{W}$ and $\mathbf{H}$ matrices obtained through NMF by $\left(diag(\mathbf{1}^T\mathbf{W}_0)\right)^{-1}$ and $diag(\mathbf{1}^T\mathbf{W}_0)$ respectively. Conversely, PLSI updated rules may be utilized to perform NMF by preliminarily scaling the $\mathbf{W}_0$ and $\mathbf{H}_0$ matrices into probability densities of the appropriate form, and factoring the scaling factor $diag(\mathbf{1}^T\mathbf{W}_0)$ back into the final estimates. It is worthy of note that while the $\left(\mathbf{1}^T\mathbf{W}_n\right)^{-1}$ term in Equation 79 is merely cosmetic within the normalized PLSI framework, within the NMF update formulation it actually fulfils the role of a scaling term that maintains a constant relation between the NMF and PLSI updates.

Finally, while PLSI has been defined in this paper as applying primarily to *distributions*, *i.e.*, $\mathbf{V}$ matrices whose elements sum to 1.0, the update rules do not change if $\mathbf{V}$ is unnormalized. Let v represent the scaling factor that would convert $\mathbf{V}$ to a distribution, such that

$\mathbf{V} = v\bar{\mathbf{V}}$, where $\bar{\mathbf{V}}$ is normalized, *i.e.*, its elements sum to 1.0. The relationship between the component matrices obtained through the decomposition of $\mathbf{V}$, which we represent by $\mathbf{W}_n^\dagger$ and $\mathbf{H}_n^\dagger$ and those obtained from the decomposition of $\bar{\mathbf{V}}$, which we represent as before by $\mathbf{W}_n$ and $\mathbf{H}_n$, starting from the *same* initial value $(\mathbf{W}_0^\dagger, \mathbf{H}_0^\dagger) = (\mathbf{W}_0, \mathbf{H}_0)$, is easily shown to be:

$$\mathbf{H}_{n+1}^\dagger = v\mathbf{H}_{n+1} \tag{97}$$

$$\mathbf{W}_{n+1}^\dagger = \mathbf{W}_{n+1} \tag{98}$$

We briefly outline the proof for Equation 97 below:

$$\mathbf{H}_1^\dagger = \left(diag(\mathbf{1}^T\mathbf{W}_0^\dagger)\right)^{-1}\left(\mathbf{H}_0^\dagger \odot {\mathbf{W}_0^\dagger}^T \frac{\mathbf{V}}{\mathbf{W}_0^\dagger\mathbf{H}_0^\dagger}\right) \tag{99}$$

$$= \left(diag(\mathbf{1}^T\mathbf{W}_0^\dagger)\right)^{-1}\left(\mathbf{H}_0^\dagger \odot {\mathbf{W}_0^\dagger}^T \frac{v\bar{\mathbf{V}}}{\mathbf{W}_0\mathbf{H}_0}\right) \tag{100}$$

$$= v\left(diag(\mathbf{1}^T\mathbf{W}_0)\right)^{-1}\left(\mathbf{H}_0 \odot {\mathbf{W}_0}^T \frac{\mathbf{V}}{\mathbf{W}_0\mathbf{H}_0}\right) \tag{101}$$

$$= v\mathbf{H}_1 \tag{102}$$

It can similarly be shown that $\mathbf{W}_1^\dagger = \mathbf{W}_1$. Now assuming Equations 97 and 98 to be true for some n, it is easy to show that they must also hold for $n+1$, thereby completing the proof through induction.

Consequently, the relationship between $\mathbf{W}_n^\dagger$ and $\mathbf{H}_n^\dagger$ and the corresponding matrices obtained through PLSI decomposition of $\bar{\mathbf{V}}$ is given by

$$\mathbf{W}_n^\dagger = \bar{\mathbf{W}}_n\left(diag(\mathbf{1}^T\mathbf{W}_0^\dagger)\right) \tag{103}$$

$$\mathbf{H}_n^\dagger = v\left(diag(\mathbf{1}^T\mathbf{W}_0^\dagger)\right)^{-1}\bar{\mathbf{H}}_n \tag{104}$$

Thus, given any matrix $\mathbf{V}$, NMF decomposition of it can be obtained from the PLSI decomposition of the normalized matrix $\bar{\mathbf{V}}$, similarly the PLSI decomposition of the normalized matrix can be computed exactly from the NMF decomposition of the unnormalized matrix.

We have shown how the update equations of NMF when using the KL-like distance are effectively identical to the EM updates of the PLCA model. Using multidimensional versions of PLCA we can easily extend the concept of NMF to operate on non-negative tensors of arbitrary rank, and vice versa we can apply the convolutive versions of NMF (15) on probabilistic models.

References

[1] Brown, J.C., 1991. Calculation of a Constant Q Spectral Transform, in *Journal of the Acoustical Society of America* vol. 89, pp. 425-434.

[2] Hofmann, T., 1999. Probabilistic Latent Semantic Analysis in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (UAI'99).

[3] Rost, J. and Langeheine, R. Eds., 1997. *Applications of Latent Trait and Latent Class Models in Social Sciences*. Waxmann, New York.

[4] Lazarfeld, P.F. and Henry, N.W., 1968. *Latent Structure Analysis*. Houghton Mifflin, Boston.

[5] Lee D.D. and Seung H.S., 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, Vol. 401, No. 6755. (21 October 1999), pp. 788-791.

[6] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In Advances in Neural Information Processing Systems 13, 2001.

[7] Hoyer, P.O., 2004. Non-negative Matrix Factorization with sparseness constraints, *Journal of Machine Learning Research* 5:1457-1469, 2004.

[8] Li, L. and Speed, T., 2000. Deconvolution of sparse positive spikes: is it ill-posed?, University of California at Berkeley, Department of Statistics Technical Report 586.

[9] Bro, R. "PARAFAC. Tutorial and applications", in *Chemometrics and Intelligent Laboratory Systems*, Volume 38, Issue 2 , October 1997, Pages 149-171

[10] Frey, B. J. and Jojic, N., 1999. Transformed component analysis: Joint estimation of spatial transformations and image components. *Proceedings of the IEEE International Conference on Computer Vision, September 1999.*

[11] Frey, B. J. and Jojic, N. 2003. Transformation-invariant clustering using the EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1, January 2003, pp. 1-17.

[12] Paatero, P. and Tapper, U., "Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values", in Environmentrics, v.5, pp.111-126. 1994.

[13] Smaragdis, P.; Brown, J.C., Non-negative Matrix Factorization for Polyphonic Music Transcription", IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 177-180, October 2003

[14] Smaragdis, P., "Discovering Auditory Objects Through Non-Negativity Constraints", Statistical and Perceptual Audio Processing (SAPA), SAPA 2004, October 2004

[15] Smaragdis, P., "Non-negative Matrix Factor Deconvolution; Extraction of Multiple Sound Sources from Monophonic Inputs", International Congress on Independent Component Analysis and Blind Signal Separation, ISBN: 3-540-23056-4, Vol. 3195/2004, pp. 494, September 2004 (Springer Lecture Notes in Computer Science).

[16] Papoulis, A. "Probability, Random Variables, and Stochastic Processes", McGraw-Hill, 3rd edition,1991. page 172.

[17] Cover T, and Thomas, J. "Elements of Information Theory", Wiley Interscience, 1991.

[18] M. Schmidt and M. Morup, "Nonnegative Matrix Factor 2D Deconvolution for Blind Single Channel Source Separation", Proceedings of the International Conference on Independent Component Analysis 2006.

[19] Shashanka, M.V.S. "Latent Variable Framework for Modeling and Separating Single Channel Acoustic Sources", Doctoral Dissertation, Department of Cognitive and Neural Systems, Boston University, August 2007.

[20] Shashanka, M.V.S., B. Raj and P. Smaragdis. "Sparse Overcomplete Latent Variable Decomposition of Counts Data", Neural Information Processing Systems Conference (NIPS), Vancouver, Canada, Dec 2007

[21] Brand, M. "Structure Learning in Conditional Probability Models via an Entropic Prior and Parameter Extinction", Neural Computation vol. 11, n. 5, pp. 1155-1182, 1999.