Learning and storing the parts of objects: IMF

Ruairí de Fréin[†] [††]

[†] Amadeus SAS, Sofia Antipolis,
Nice, France
[††]KTH Royal Institute of Technology,
Sweden
web: https://robustandscalable.wordpress.com

in: Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on.
See also BibTeX entry below.

BibTeX:

```
@article{deFrein14Learning,
author={Ruair\'{i} de Fr\'{e}in$^\dagger$ $^{\dagger\dagger}$},
booktitle={Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on},
title={{Learning and storing the parts of objects: IMF}},
year={2014},
pages={1-6},
keywords={matrix decomposition;signal processing;Barbara;IMF;Peppers;compression ratio;learning
algorithm;nonnegative matrix factorization;Approximation methods;Dictionaries;Encoding;
Quantization (signal);Signal processing algorithms;Signal to noise ratio;Vectors;compression;
matrix factorization},
doi={10.1109/MLSP.2014.6958926},
month={Sept},
url = {http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6958926&isnumber=6958838},}
```


document created on: January 26, 2016
created from file: rdefreinMLSP14.tex
cover page automatically created with CoverPage.sty
(available at your favourite CTAN mirror)

2014 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2014, REIMS, FRANCE

LEARNING AND STORING THE PARTS OF OBJECTS: IMF

Ruairí de Fréin

Telecommunications Software and Systems Group
Ireland

ABSTRACT

A central concern for many learning algorithms is how to efficiently store what the algorithm has learned. An algorithm for the compression of Nonnegative Matrix Factorizations is presented. Compression is achieved by embedding the factorization in an encoding routine. Its performance is investigated using two standard test images, Peppers and Barbara. The compression ratio (18:1) achieved by the proposed Matrix Factorization improves the storage-ability of Nonnegative Matrix Factorizations without significantly degrading accuracy ($\approx$ 1-3dB degradation is introduced). We learn as before, but storage is cheaper.

Index Terms— matrix factorization; compression

1. INTRODUCTION

Nonnegative Matrix Factorization (NMF) uncovers feature vectors and vectors of activations of these features in a signal ensemble with structural regularity via non-negativity constraints [1]. The application of NMF is found in many different types of communications and data processing data-sets. To date the compression of NMF, which would allow for efficient storage of the derived factorization in resource constrained applications, has not been considered. Given the regularity of the structures produced by NMF and the size of many modern data-sets, we consider the storage gains achievable by proposing a compressible NMF.

NMF learns element-wise real-valued nonnegative factors. The problem of data compression is not *encoded* as an optimization constraint in NMF. This is problematic if many factorizations are required for subsequent processing. However if NMF was to learn a factorization which was amenable conversion into an integer sequence, the factorization could then be compressed by 1) identifying the limitations of the factorization and 2) devising a coding scheme which compressed the factorization, using a scheme based on [2], resulting in an integrated factorization-compression system. We propose an Integer Matrix Factorization (IMF) as the missing step for the system described above.

NMF learns a lossy factorization. In general the derived factorization is non-unique [3, 4] but *satisfactory* for most applications. Satisfaction is generally based on some distortion measure, e.g. the Frobenius-norm, Kullback Leibler Divergence [1]. Usage of the α-divergence [5] and β-divergence [6] has been motivated for speech. If NMF were unique and exact, the optimal factorization would give rise to identical distortion scores. Given that the solution is non-unique and non-exact, but typically satisfactory, we posit that we have the freedom to choose an arbitrary satisfactory lossy factorization with respect to the distortion constraint of choice (for example, a candidate solution might be acceptable if it is within 1-3dB of the expected NMF Signal-to-Noise-Ratio) if it is amenable to efficient coding. Once a suitable factorization has been produced, it is crucial that any post-factorization-coding process is loss-less. Minimum redundancy coding has received extensive treatment in the literature [7, 8, 9, 10, 11, 12, 13]. The statistics of the data ensemble may be used to compress the data ensemble; however, prior knowledge of the statistics of the factorization may not be assumed as NMF is non-unique (without performing multiple passes through the factorization, which is not practical in a alternating minimization routine). Universal coding schemes address this problem by coupling learning with the coding process for varying source characteristics [14, 15]. We take the 1977 approach of Lempel and Ziv [2] (LZ77) for post-factorization-coding, even though encoding ratios may be slightly improved using more recent extensions. This decision is justified because utilities such as gzip [16] are widely available which also increases the portability of our approach.

IMF algorithm has three features that distinguish it from previous algorithms and are worthy of comment: 1) we provide the ability to learn element-wise integer-valued NMFs. Element-wise integer constraints are preserved by interlacing two new update rules with the existing NMF updates. 2) Integer-NMF combined with a universal loss-less coding scheme advances the state-of-the-art as many existing factorizations may be adapted in a similar manner, yielding significant compression ratio improvements, whilst incurring little additional distortion. Once an integer factorization can be learned, the use of a standard compression technique ensures that the factorization is portable and efficiently stored. 3) The updates proposed by integer NMF are multiplicative and come with the same monotonic convergence guarantees as traditional NMF updates. This is because the error introduced by *integerizing* the factors is folded into the traditional NMF updates. In summary NMF can be learned with approximately the same levels of accuracy as before (within 1-3dB) but we demonstrate that ≈ 18 NMFs can be stored in the same space as 1 traditional NMF.

By way of preparation we introduce some frequently used notation. The vector $\mathbf{1}_M$ is a column vector of M ones. We distinguish between element-wise nonnegative $\boldsymbol{X}$ and element-wise nonnegative integer matrices, $\hat{\boldsymbol{X}}$, by a hat. We denote the r-th column entry of a matrix by $\boldsymbol{X}(:, r)$. It is frequently convenient to consider the sequence X which is generated from the column vector $\boldsymbol{X}(:, r)$, by arranging the entries according to *column ordering* in the derived sequence.

In terms of paper organization, in Section 2 we illustrate how our approach is new, we highlight related results, and we evaluate their strengths and weaknesses. In Section 3 we introduce the new factorization-compression algorithm. In Section 4 we empirically evaluate the success of the factorization-compression routine. Finally, in Section 5 we discuss the impact our results will have on the Machine Learning and Signal Processing community.

2. RELATED WORK

The use of dictionaries for compression is analogous to the type of representation or encoding of an observation ensemble achieved by NMF and sparse coding [17, 18]. Sliding-window compression is appealing given the sparsity of NMF basis functions and their structural regularity. In many cases non-negativity constraints are enough to recover the original signals if the signals are sparse enough [19]. Moreover, results on the conditions for establishing the uniqueness of NMF rely on zero-groundedness of the signals [4], or determining binary closure patterns in the signal ensemble [20], which typically explains the success of sparse coding techniques [18]. Learning sparse factors is generally a good indicator for achieving good compression, much in the same way as integer sequences with many zeros are typically well compressed.

The family of Lempel-Ziv source coding algorithms [2, 21] is: popular (due to [22]); universally optimal (their asymptotic compression rate approaches the entropy rate of the source for any stationary ergodic source [23]); and simple to implement. We adopt a "string-matching on a sliding window" procedure [2] (LZ77) to encode IMF. The alternative, an adaptive dictionary compression algorithm [21] (LZ78), has been in use for longer (cf. UNIX compress) but does not compress as well as LZ77 (which forms the basis for pkzip, gzip, Stacker, Microsoft Windows). However, LZ77 is more computationally intensive than LZ78. Modern implementations of gzip are arguably as fast as many LZ78 variants.

In this paper we use variable-to-variable-length codes to represent IMF. Similar in spirit to LZ77, both the number of source symbols encoded and the number of encoded bits per code-word are variable; the symbol set is also learned via a lossy integer matrix factorization. Unlike Shannon and Huffman codes, our approach does not require prior knowledge of the sources statistics. Using a LZ77 mechanism, adaptation ensures that the average code-word length per source letter is minimized.

3. FACTORIZATION-COMPRESSION ALGORITHM

The proposed factorization-compression algorithm consists of 1) learning an integer nonnegative matrix factorization algorithm whose elements are drawn from a finite set of integers $\mathcal{I} \subset \mathcal{Z}$; 2) a rule for arranging the factorization parameters as a sequence composed of integers (and a small set of non-integers); 3) a rule for parsing strings of integers from the finite alphabet, $\mathcal{I}$, into sub-strings whose lengths do not exceed a prescribed integer L_s (the encoder's look-ahead buffer, cf. [2]); and 4) a coding scheme which maps these sub-strings sequentially into uniquely decipherable code words of fixed length L_c over the alphabet $\mathcal{I}$. We describe the mechanism of the factorization-parsing-coding procedure in the following subsections.

3.1. Factorization - Integer Matrix Factorization

Definition 1 *NMF factorizes a matrix $\boldsymbol{V} \in \mathbb{R}^{M\times N}$ into the product of two matrices $\boldsymbol{W} \in \mathbb{R}^{M\times R}$ and $\boldsymbol{H} \in \mathbb{R}^{R\times N}$, yielding the estimate $\tilde{\boldsymbol{V}} = \boldsymbol{W}\boldsymbol{H}$. All matrices have nonnegative elements and $R < M, N$. The Kullback Leiber Divergence (KLD) is a frequently used objective function.*

$$D(\boldsymbol{V}||\tilde{\boldsymbol{V}}) = \sum_{m,n} \boldsymbol{V}(m,n)\log\frac{\boldsymbol{V}(m,n)}{\tilde{\boldsymbol{V}}(m,n)} - \boldsymbol{V}(m,n) + \tilde{\boldsymbol{V}}(m,n) \tag{1}$$

Lee and Seung introduced a step-size function for gradient descent and two alternating multiplicative updates (element-wise multiplication and division are denoted, $\odot$ and $\oslash$).

$$\boldsymbol{H} \leftarrow \boldsymbol{H} \odot \boldsymbol{W}^T\frac{\boldsymbol{V}}{\boldsymbol{W}\boldsymbol{H}} \oslash (\mathbf{1}_M^T\boldsymbol{W})^T\mathbf{1}_N^T, \tag{2}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} \odot \frac{\boldsymbol{V}}{\boldsymbol{W}\boldsymbol{H}}\boldsymbol{H}^T \oslash \mathbf{1}_M(\boldsymbol{H}\mathbf{1}_N)^T. \tag{3}$$

NMF is non-unique and inexact, which implies that the mixing model may represent any member of the family

$$\{\boldsymbol{W}, \boldsymbol{H} | \boldsymbol{V} \approx (\boldsymbol{W}\boldsymbol{A})(\boldsymbol{A}^{-1}\boldsymbol{H})\}, \tag{4}$$

where $\boldsymbol{A}$ is a permutation times a diagonal matrix. Satisfaction can be decided upon using the inequality, $D(\boldsymbol{V}||\tilde{\boldsymbol{V}}) < \epsilon$, where ϵ is a user specified distortion (or convergence) criteria which is meaningful in the application domain. Many factors $\boldsymbol{W}$ and $\boldsymbol{H}$ may satisfy a given inequality constraint. A central concern in many learning applications is how to efficiently store what has been learned for later use. The first sub-problem we solve is how to select a pair of factors which are amenable to efficient storage without significant loss of accuracy from the family referred to in Eqn. 4. We call this Integer Matrix Factorization (IMF). The crucial property these factors should have is that they are drawn element-wise from a finite integer symbol set.

IMF can be achieved by taking a nonlinear approximation of each element of the factor estimates produced by the Lee and Seung updates at convergence using the quantizers

$$\hat{\boldsymbol{H}} \leftarrow \left\lfloor \frac{\mathbf{1}}{\Delta^\star}\boldsymbol{H} + \frac{1}{2} \right\rfloor \quad \hat{\boldsymbol{W}} \leftarrow \left\lfloor \boldsymbol{W}\frac{\mathbf{1}}{\Delta^\star} + \frac{1}{2} \right\rfloor; \tag{5}$$

The problem with this approach is that 1) one quantization step-size, $\Delta^\star$ is not appropriate for all rows and columns of $\boldsymbol{H}$ and $\boldsymbol{W}$; and more importantly 2) fixed non-alternating quantization does not allow the factors to absorb the distortion introduced by quantization, and the quantization functions to absorb the distortion introduced by a poor approximation and quantization. A sensible approach is to learn the quantization functions as well as the factors so that the quantized approximation tends to improve as the $\boldsymbol{W}$ and $\boldsymbol{H}$ converge; we interleave the optimization processes. This process is illustrated Fig. 1, where a floor function $\lfloor\cdot\rfloor$ is used for quantization. The quantization functions are adapted for each R as the factors and quantizers converge. The elements of the factor $\boldsymbol{H}$ assume one of the dashed line quantization levels. The cardinality of the symbol set of $\hat{\boldsymbol{H}}$ is smaller than $\boldsymbol{H}$.

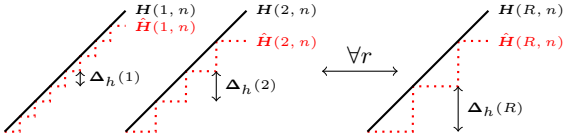

Fig. 1. Quantization functions are adapted for each $H(r,:)$.

Definition 2 ***IMF*** *factorizes a matrix $\boldsymbol{V} \in \mathbb{R}^{M\times N}$ into the product of two element-wise integer matrices $\hat{\boldsymbol{W}} \in \mathcal{I}^{M\times R}$ and $\hat{\boldsymbol{H}} \in \mathcal{I}^{R\times N}$ and two diagonal matrices $diag(\boldsymbol{\Delta}_w)$ and $diag(\boldsymbol{\Delta}_h)$. The operation $diag(\cdot)$ places the elements of the vectors $\boldsymbol{\Delta}_w \in \mathbb{R}^{R\times 1}$ and $\boldsymbol{\Delta}_h \in \mathbb{R}^{R\times 1}$ on the diagonal of an $R \times R$ matrix. The factorization is expressed as*

$$\tilde{\boldsymbol{V}} = (\hat{\boldsymbol{W}}diag(\boldsymbol{\Delta}_w))(diag(\boldsymbol{\Delta}_h)\hat{\boldsymbol{H}}) = \boldsymbol{W}\boldsymbol{H}. \tag{6}$$

By grouping the multiplications we have $\boldsymbol{W} = (\hat{\boldsymbol{W}} diag(\boldsymbol{\Delta}_w))$ *and* $\boldsymbol{H} = (diag(\boldsymbol{\Delta}_h)\hat{\boldsymbol{H}})$*, which are related to the factorizations in (Eqn. 4). The quality of the factorization is computed using the KLD*

$$D(\boldsymbol{V}||\tilde{\boldsymbol{V}}) = \sum_{m,n} \boldsymbol{V}(m,n) \log \frac{\boldsymbol{V}(m,n)}{\tilde{\boldsymbol{V}}(m,n)} - \boldsymbol{V}(m,n) + \tilde{\boldsymbol{V}}(m,n) \tag{7}$$

The KLD objective function is minimized –such that $\hat{\boldsymbol{W}} \in \mathcal{I}^{M \times R}$ *and* $\hat{\boldsymbol{H}} \in \mathcal{I}^{R \times N}$*– by interlacing two new update rules for the parameters* $\boldsymbol{\Delta}_w$ *and* $\boldsymbol{\Delta}_h$*, with the updates (Eqns 2 and 3). In terms of algorithm structure, first, we minimize the KLD with respect to the products* $\boldsymbol{W}$ *(*$\hat{\boldsymbol{W}}$ *and* $diag(\boldsymbol{\Delta}_w)$*) and* $\boldsymbol{H}$ *(*$diag(\boldsymbol{\Delta}_h)$ *and* $\hat{\boldsymbol{H}}$*), and then with respect to the arguments of these products,* $diag(\boldsymbol{\Delta}_w)$ *and* $diag(\boldsymbol{\Delta}_h)$*. This is achieved by introducing a step-size function for gradient descent minimization of the vectors* $\boldsymbol{\Delta}_w$ *and* $\boldsymbol{\Delta}_h$*. This function results in two alternating multiplicative updates*

$$\boldsymbol{\Delta}_h(r) \leftarrow \boldsymbol{\Delta}_h(r) \frac{\mathbf{1}_M^T \left[\frac{\mathbf{V}}{\tilde{\mathbf{V}}} \odot \mathbf{W}(:,r)\hat{\mathbf{H}}(r,:) \right] \mathbf{1}_N}{\mathbf{1}_M^T \left[\mathbf{W}(:,r)\hat{\mathbf{H}}(r,:) \right] \mathbf{1}_N} \tag{8}$$

$$\boldsymbol{\Delta}_w(r) \leftarrow \boldsymbol{\Delta}_w(r) \frac{\mathbf{1}_M^T \left[\frac{\mathbf{V}}{\tilde{\mathbf{V}}} \odot \hat{\mathbf{W}}(:,r)\mathbf{H}(r,:) \right] \mathbf{1}_N}{\mathbf{1}_M^T \left[\hat{\mathbf{W}}(:,r)\mathbf{H}(r,:) \right] \mathbf{1}_N} \tag{9}$$

At the end of each iteration, we scale $\boldsymbol{\Delta}_w$ *and* $\boldsymbol{\Delta}_h$ *so that they sum to one.*

The mappings $\boldsymbol{W} \leftarrow (\hat{\boldsymbol{W}} \text{diag}(\boldsymbol{\Delta}_w))$ and $\boldsymbol{H} \leftarrow (\text{diag}(\boldsymbol{\Delta}_h)\hat{\boldsymbol{H}})$ warrant further attention. Fig. 1 illustrates the nonnegative element $\boldsymbol{H}(1,n)$ and its quantized counter-part $\hat{\boldsymbol{H}}(1,n)$. The multiplicative update rules (Eqn. 8 and 9) adjust the size of the quantization step-sizes $\boldsymbol{\Delta}_h$ for all rows of $\boldsymbol{H}(r,n)$. The smaller the step-size $\boldsymbol{\Delta}_h(r)$ the finer the resolution of $\boldsymbol{H}(r,:)$. IMF trades-off the resolution in the rank-1 approximations of $\boldsymbol{V}$, e.g. $\hat{\boldsymbol{W}}(:,r)\boldsymbol{\Delta}_w(r)\boldsymbol{\Delta}_h(r)\hat{\boldsymbol{H}}(r,:)$ with the other $R-1$ rank-1 approximations. The intuition is that certain rank-1 approximations of an NMF solution tend to a small symbol set cardinality, whereas other rank-1 approximations tend to a slightly larger symbol set cardinality. As motivation, the seminal NMF paper describes an algorithm for "learning the parts of objects" [24]; in practice these parts consists of a few non-zero entries in the factors, and many close-to-zero values. IMF attempts to reduce the total symbol set cardinality $|\mathcal{I}|$ by adjusting the quantization step-size parameters so that the quantization functions match the underlying tendencies of each rank-1 approximation.

The Lee and Seung updates guarantee that the objective is monotonically decreased each time they are applied. We make a similar claim about the quantization step-size updates. No such claim can be made about the application of a nonlinear quantizer which introduces uniformly distributed noise. Moreover, signal independence, and a uniformly distributed noise model is not strictly valid [25] for quantization error. The assumption fails when the quantization step-size is comparable with the signal magnitudes. To address this problem after each iteration we scale $\boldsymbol{\Delta}_w$ and $\boldsymbol{\Delta}_h$ so that they sum to one. Moreover, this normalization ensures $\boldsymbol{\Delta}_w$ and $\boldsymbol{\Delta}_h$ do not converge to a small number (for all r), causing IMF to revert to tradition NMF.

All that remains is to justify our quantizer design decisions in light of the monotonic convergence goal. We use a uniform quantizer: the authors of [26] demonstrate that it has an output entropy which is asymptotically smaller than that of any other quantizer, independent of the density function or the error criterion. In addition, we choose a uniform mid-tread quantizer to keep the factors in the nonnegative orthant. Finally, to improve convergence behaviour, we introduce a randomized component into the quantization functions. The scheme is described in Table 1 for the $\boldsymbol{H}$ factor. We perform No_ptrials Bernoulli trials for each probability of success $p \in B_p$. The sequence of trial probabilities is B_p –an ordered list from large to small of probabilities of success. In each trial we select a binary matrix, $\mathcal{U}_{RN}$, by drawing RN uniformly distributed values in the range $(0,1)$ and comparing them with the current probability of success, p. We generate a matrix of quantization functions by replacing the ceil function $\lceil \cdot \rceil$ in our quantizer with a floor function $\lfloor \cdot \rfloor$ according to $\mathcal{U}_{RN}$. We evaluate if the matrix of quantizers has decreased or increased the objective by comparing the new factorization with its previous value D_{t-1}. The process is stopped once the first suitable quantization function is found, the quantization steps are updated (using Eqn. 8) and normalized so that they sum to one. In summary, each IMF iteration applies (Eqn. 2); it then quantizes $\boldsymbol{H}$ using Tab. 1; it applies (Eqn. 3); and finally, it quantizes $\boldsymbol{W}$ using Tab. 1. Four variants of this approach are evaluated: 1) we call the floor/ceil

Table 1. Randomized Quantization Functions

for $p = B_p$
  for k=1:No_ptrials
    $i = \mathcal{U}_{RN} > p$;
    $\hat{\boldsymbol{H}} = \lceil \text{diag}(1 \oslash \boldsymbol{\Delta}_h)\boldsymbol{H} + .5 \rceil$, $A = \lfloor \text{diag}(1 \oslash \boldsymbol{\Delta}_h)\boldsymbol{H} - .5 \rfloor$
    $\hat{\boldsymbol{H}}(i) = A(i)$
    $\boldsymbol{H}' = \text{diag}(\boldsymbol{\Delta}_h)\hat{\boldsymbol{H}} + eps$
    if $D(\boldsymbol{V}||\hat{\boldsymbol{W}}\text{diag}(\boldsymbol{\Delta}_w)\boldsymbol{H}') < D_{t-1}$
      break
    end, **end**, **end**
$\boldsymbol{H} = \text{diag}(\boldsymbol{\Delta}_h)\hat{\boldsymbol{H}} + eps$; update $\boldsymbol{\Delta}_h$, normalize $\boldsymbol{\Delta}_h = \boldsymbol{\Delta}_h / \mathbf{1}_R^T \boldsymbol{\Delta}_h$

based quantization variants "floorfix" and "flooradapt". The postfix "adapt" indicates that the quantization step-sizes are adapted. For the purpose of comparison we also evaluate the cases where $\boldsymbol{\Delta}_h$ is initialized randomly and not updated by IMF to evaluate the effect of the $\boldsymbol{\Delta}_h$ and $\boldsymbol{\Delta}_w$ updates. This gives rise to the postfix "fix"; 2) in another pair of variants of the algorithm "roundfix" and "roundadapt" we use the round function instead of floor, and randomly round-up or down depending on the effect on the objective function using the scheme in Tab. 1. In general the number of Bernoulli trials required for the adaptive quantization IMFs is less than the number of trials required for the fixed quantization variants. This is because the quantizers converge as the factors converge. Bernoulli trials are typically only required during the first 1-5 iterations of IMF. In effect these trials select the initialization for the factors of IMF; once a good set of initial factors is found, the number of trials drops-off dramatically.

3.2. Parsing - Sequence of Integers Generator

Once IMF has converged, we generate a sequence of integers by re-arranging the factors. Decoding is exact and is achieved by reversing these steps. Depending on the dimensions of the factors, e.g. M, N, R it may be more computationally efficient to 1) generate a column or row vector sequence (depending on row or column major ordering) and 2) to transpose the $\boldsymbol{H}$ matrix instead of the $\boldsymbol{W}$ matrix. We assume that $N \gg M$. Parsing is performed as follows.

1) Concatenate the columns of the factor $\hat{\boldsymbol{W}}$ and take the transpose of the concatenated vector to generate a sequence

$$W = [\hat{\boldsymbol{W}}(:,1)^T | \hat{\boldsymbol{W}}(:,2)^T | \ldots | \hat{\boldsymbol{W}}(:,R)^T] \tag{10}$$

2) Concatenate the rows of the factor $\hat{\boldsymbol{H}}$ to generate H,

$$H = [\hat{\boldsymbol{H}}(1,:)|\hat{\boldsymbol{H}}(2,:)|\dots|\hat{\boldsymbol{H}}(R,:)] \tag{11}$$

3) Generate the sequence of integers, S, by concatenating the sequences, W and H, the dimensionality parameters and quantization step-sizes $\boldsymbol{\Delta}_w$ and $\boldsymbol{\Delta}_h$ (which are real-valued)

$$S = M|R|N|W|H|\boldsymbol{\Delta}_w|\boldsymbol{\Delta}_h. \tag{12}$$

We drop vector notation from here on for convenience.

3.3. Encoder - Sequential Data Compression

For completeness we describe a derivative of the well-known LZ77 scheme [2]. IMF generates the factors $\hat{\boldsymbol{W}} \in \mathcal{I}^{M\times R}$ and $\hat{\boldsymbol{H}} \in \mathcal{I}^{R\times N}$, where $\mathcal{I} \subseteq \{1, 2, \dots \max\{[\hat{\boldsymbol{W}}^T|\hat{\boldsymbol{H}}]\}\}$. These factor matrices are then parsed into S in (Eqn. 12). A sub-string of S, which starts at position i and ends at position j, is denoted $S(i, j)$. The length of a string is $\ell(S)$. The sub-sequence $S(1, j)$ is called a proper prefix of S if $1 \leq j < \ell(S)$. Given a proper prefix $S(1, j)$ of S and a positive integer i such that $i \leq j$, $L(i)$ denotes the longest match, the largest nonnegative integer $\ell \leq \ell(S) - j$ such that

$$S(i, i + \ell - 1) = S(j + 1, j + \ell). \tag{13}$$

The position for which $L(s) = \max_{1\leq i\leq j} L(i)$ is denoted by s. In Fig. 2 we define a look-ahead buffer of length $L_s = 4$ and a dictionary of length 6. Fig. 2 illustrates the sequence encoding in terms of the relative position, the length of the match, and the next integer, in a triple which is listed on the RHS. These triples are radix-$|\mathcal{I}|$ encoded. The grey integers (raised to the power of star) are the next integers after the encoded sequence. The LZ77 algorithm performs the following iteration: 1) find the position of the longest match in the dictionary (blue) with the the look-ahead buffer (red, including the grey integer) relative to the cursor j; find the length of the longest match $L(s)$; record the triple (the relative position, $L(s)$, and the next integer) encoding them in radix-$|\mathcal{I}|$. These triples are given along-side the rows in Fig. 2. Advance the cursor by $L(s) + 1$ symbols and repeat. The dictionary is initially empty which is why the topmost triple starts with a $(\cdot, 0, 1)$ entry.

Gzip, the compression algorithm we employ, has additional optimizations over the scheme above. Firstly, it Huffman codes the position, length and next integers triples. Secondly, it is non-greedy. It uses shorter matches so that next match is better. And thirdly, it uses a hash table to store the dictionary. The IMF-parsing-encoding scheme works well because gzip adapts well to changes in the integer sequence which arise because the different columns and rows of $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{H}}$ have different statistics. Modern gzip codes use probability coding as a second pass and compress much better.

4. NUMERICAL EVALUATION

We numerically evaluate the compression ratios achieved and the distortion introduced by IMF using two standard test images, Barbara and Peppers. We vectorize 8×8 blocks of the images. We arrange them in the columns of $\boldsymbol{V}$ and perform IMF over the range of ranks $10 \leq R \leq 60$ in steps of 5. We perform 50 Monte Carlo runs for each parametrization and tabulate the average SNRs and file-sizes for each IMF variant and parametrization in Tab. 2 and 3. The precision of the results is reduced here to save space. Each IMF runs for 500 iterations. We use gzip's default compression level -6 (that is, biased towards high compression at the expense of speed)

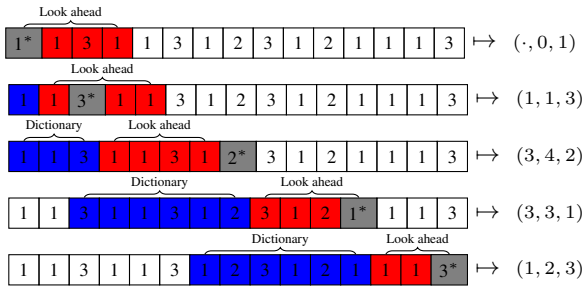

Fig. 2. Illustration of the sequence encoding scheme for S.

for all algorithms. As an aside, the maximum possible compression level is -9. Each run converges to a good solution, e.g. each factorization's SNR is within (1-3dB) of Lee-Seung's NMF. We attribute this (1-3dB) to the error introduced by quantization. We do not perform analysis of the convergence of IMF here, save to mention that $> 90\%$ of the trials for each parametrization are within $<$ 1dB of the mean SNR. This empirically supports the claim that IMF converges to a good solution and that the inaccuracy is due to quantization error. "flooradapt" IMF (rows 7-8) gives the best SNR (row 8) and compression file-size in kB (row 7) trade-off for both images of the IMF variants. Row 9 gives the file-size for uncompressed NMF and the rounded factor by which it is greater than "flooradapt" IMF in brackets. Row 10 gives the file-size for traditional NMF when it has been *gzipped* (and the rounded factor by which it is greater than "flooradapt" IMF in brackets). The SNR for NMF is listed in row 11 so that we have a benchmark accuracy score to compare IMF with. "flooradapt" IMF storage is 17-19 times more efficient than traditional uncompressed NMF. It is 7-9 times more efficient than NMF which has been gzipped. This storage gain is achieved by only degrading the SNR by 5-10%. Both images yield similar results when the results of 50 trials are averaged. This underlines the claim that the algorithms converge and compress accurately.

5. DISCUSSION

In terms of the relevance of this work to the MLSP community, steps 1 and 2 above describe a new approach for NMF. IMF's components are drawn from a finite integer alphabet. Steps 3 and 4 consist of the application of Lempel-Ziv-type data compression algorithm to NMF. We do not claim to have significantly advanced the state-of-the-art of universal algorithms for sequential data compression; however, we have demonstrated how to reformulate an increasingly prevalent factorization, such that it readily admits to one of the most pervasive sequential data compression techniques, e.g. gzip, which is based on [2]. Alternative compression routines include 7-zip, bzip2, zip, zlib, gif and compress on unix (based on [21]). The storage savings demonstrated above are undoubtedly of crucial importance to the MLSP community. Taken as a whole, steps 1-4 are new and facilitate more efficient storage of NMF without introducing significant distortion. IMF makes the efficient storage of NMFs possible. In terms of the likely adoption of these ideas, there is a strong argument to be made for the need for improving the storage of NMF in the areas of 1) Text mining: Topic Modeling; Document Clustering; Topic detection and trend tracking, email analysis. 2) Image analysis and computer vision: Feature representation and sparse coding; Video tracking. 3) Social Network Analysis: Community structure and trend detection; Recommendation systems. 4) Acoustic Signal

Table 2. Barbara: Mean Filesize and SNR (dB) vs R. Rounded compression ratios are in brackets.

R	10	15	20	25	30	35	40	45	50	55	60
roundrand (kB)	42.21	64.89	88.31	112.48	137.74	160.01	186.17	211.03	236.31	262.56	287.29
SNR (dB)	16.69	17.54	18.38	19.23	19.98	20.65	21.39	22.08	22.67	23.27	23.86
roundadapt (kB)	40.87	63.36	86.66	110.59	134.69	157.70	183.37	207.85	232.64	258.58	283.40
SNR (dB)	17.10	17.93	18.85	19.82	20.54	21.15	21.96	22.67	23.22	23.82	24.36
floorrand (kB)	43.24	67.61	92.86	119.10	145.90	171.23	199.69	227.00	254.99	283.66	311.58
SNR (dB)	18.34	19.58	20.78	21.91	22.92	23.93	24.80	25.49	26.09	26.66	27.21
flooradapt (kB)	42.79	66.71	91.52	117.82	143.92	169.80	198.19	225.67	253.41	282.04	309.97
SNR (dB)	18.61	19.85	21.10	22.28	23.31	24.32	25.23	25.92	26.53	27.07	27.58
NMF(kB)	759(18)	1150(17)	1548(17)	1951(17)	2360(16)	2770(16)	3179(16)	3587(16)	3996(16)	4406(16)	4814(16)
NMF-gzip(kB)	357.6(8)	539.4(8)	724.4(8)	910.3(8)	1099.4(8)	1287.4(8)	1474.9(7)	1661.5(7)	1848.6(7)	2035.1(7)	2219.9(7)
SNR (dB) NMF	19.6	21.1	22.5	24.1	25.1	26.1	27.0	28.0	28.6	29.2	30.0

Table 3. Peppers: Mean Filesize and SNR (dB) vs R. Rounded compression ratios are in brackets.

R	10	15	20	25	30	35	40	45	50	55	60
roundrand (kB)	41.42	62.23	84.01	106.80	131.31	155.45	176.95	200.93	220.54	251.04	272.32
SNR (dB)	18.98	19.77	20.81	21.39	22.30	22.96	22.95	23.21	23.53	24.13	24.35
roundadapt (kB)	39.90	60.59	82.67	105.81	129.09	153.93	175.47	198.59	219.85	249.28	270.99
SNR (dB)	19.62	20.05	21.12	21.62	22.54	23.32	23.25	23.40	23.66	24.36	24.58
floorrand (kB)	41.70	63.37	87.25	110.98	137.81	164.20	187.59	214.47	239.68	271.07	297.69
SNR (dB)	21.14	22.15	22.94	23.45	24.23	24.73	24.80	25.07	25.32	25.78	25.90
flooradapt (kB)	41.19	62.62	86.42	110.19	136.73	165.63	187.45	213.67	240.23	272.17	297.49
SNR (dB)	21.37	22.13	22.82	23.34	24.11	24.69	24.71	24.90	25.14	25.67	25.76
NMF(kB)	766(19)	1161(19)	1560(18)	1962(18)	2370(17)	2780(17)	3192(17)	3603(17)	4015(17)	4427(16)	4838(16)
NMF-gzip(kB)	361(9)	545(9)	731(8)	918(8)	1106(8)	1295(8)	1484(8)	1673(8)	1862(8)	2050(8)	2239(8)
SNR (dB) NMF	23.8	25.1	25.9	26.3	26.9	27.2	27.6	27.9	28.1	28.4	28.7

Processing and Blind source Separation. 5) Financial Data Analysis. In each of these areas logging of the learned factors provides the basis for each of these tasks. If hard-disk space is the limitation, larger dictionaries of factors can be saved in a (social network) task if the underlying dictionaries are compressed as part of the learning algorithm. In addition, when memory is the limiting constraint, steps 3-4 can be interspersed with the factor updates during run time. Given this explosion of NMF application areas, we believe that our contribution is timely and important as it may provide a crucial stepping-stone towards the application of NMF as a Big-data factorization.

Towards improving the encoding: We have investigated taking a pairwise-difference of the integer sequence S to improve compression; however, the resulting compression ratios were inferior. In terms of the limitations of the approach, the compression routine processes the vectors $\mathbf{\Delta}_w$ and $\mathbf{\Delta}_h$ which are generally non-integer. Placing $\mathbf{\Delta}_w$ and $\mathbf{\Delta}_h$ at the end of S does not introduce non-integers into the compression dictionary. One potential improvement of IMF would be the ability to specify the symbols set cardinality a priori. The current embodiment of IMF is unconstrained but tends to find a small cardinality. We will consider these questions in future work, along with establishing the necessary and sufficient conditions for unique IMF. We will also consider optimizing the LZ routine used to compress IMF. We have not fully explored the range of *off-the-shelf* gzip's parameters. Moreover, better compression has been reported using more modern LZ routines, for example 7zip.

6. CONCLUSIONS

We conclude that a 18:1 compression ratio of the results of a NMF is possible if the factors are restricted to a finite integer symbol set. Restricting the factors to an integer symbol set only incurs a distortion of 5-10% of the original NMF's average SNR, typically 1-3dB. In many applications NMF basis functions are a crucial building-block for subsequent processing. We have demonstrated how to learn these functions so that they can be stored more efficiently. IMF is a member of the NMF family of solutions; we present multiplicative updates similar to those of NMF. In future work we will consider the extension of IMF to different problem domains and the effect of IMF on domain specific perceptual measures.

7. ACKNOWLEDGEMENT

This work was gratefully supported by the EU FP7 Marie-Curie IAPP SOLAS project, grant no. 612480.

8. REFERENCES

[1] Lee, D.D. and Seung, S.H., "Algorithms for Non-negative Matrix Factorization," in *NIPS*. pp., 556–62, MIT Press.

[2] Ziv, J. and Lempel, A., "A universal algorithm for sequential data compression," *Inf. Th., IEEE Trans.*, vol. 23, no. 3, pp. 337–43, May 1977.

[3] Rickard, S. and Cichocki, A., "When is non-negative matrix decomposition unique?," in *IEEE 42nd CISS*, pp., 1091–2.

[4] Laurberg, H. and Christensen, M.G. and Plumbley, M.D. and Hansen, L.K. and Jensen, S.H., "Theorems on Positive Data: On the Uniqueness of NMF," *Comput. Intell. Neurosc.*, 2008.

[5] de Fréin, R. and Rickard, Scott T., "Learning speech features in the presence of noise: Sparse convolutive robust non-negative matrix factorization," in *IEEE DSP, 16th*, pp., 1–6.

[6] Paul D. O'Grady and Barak A. Pearlmutter, "Discovering speech phones using convolutive non-negative matrix factorisation with a sparseness constraint," *Neurocomp.*, vol. 72, no. 1–3, pp. 88–101, 2008.

[7] Huffman, D.A., "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[8] Karp, Richard M., "Minimum-redundancy coding for the discrete noiseless channel," *Inf. Th., IRE Trans.*, vol. 7, no. 1, pp. 27–38, Jan. 1961.

[9] Ben Varn, "Optimal variable length codes (arbitrary symbol cost and equal code word probability)," *Information and Control*, vol. 19, no. 4, pp. 289–301, 1971.

[10] Perl, Y. and Garey, M. R. and Even, S., "Efficient generation of optimal prefix code: Equiprobable words using unequal cost letters," *J. ACM*, vol. 22, no. 2, pp. 202–214, Apr. 1975.

[11] Lempel, A. and Even, Shimon and Cohn, Martin, "An algorithm for optimal prefix parsing of a noiseless and memoryless channel," *Inf. Th., IEEE Trans.*, vol. 19, no. 2, pp. 208–14, Mar. 1973.

[12] Jelinek, F. and Schneider, K., "On variable-length-to-block coding," *Inf. Th., IEEE Trans.*, vol. 18, no. 6, pp. 765–74, Nov. 1972.

[13] Gallager, R. G., *Information Theory and Reliable Communication*, John Wiley and Sons, New York, 1968.

[14] Ziv, J., "Coding of sources with unknown statistics–II: Distortion relative to a fidelity criterion," *Inf. Th., IEEE Trans.*, vol. 18, no. 3, pp. 389–94, May 1972.

[15] Davisson, L.D., "Universal noiseless coding," *Inf. Th., IEEE Trans.*, vol. 19, no. 6, pp. 783–95, Nov. 1973.

[16] P. Deutsch, "GZIP file format specification version 4.3, RFC1952," Tech. Rep.1952, May 1996.

[17] Zibulevsky, Michael and Pearlmutter, Barak A., "Blind source separation by sparse decomposition in a signal dictionary," *Neural Comput.*, vol. 13, no. 4, pp. 863–82, Apr. 2001.

[18] Hoyer, P.O., "Non-negative sparse coding," in *Neural Networks for Signal Processing, 12th IEEE Workshop*, pp., 557–65.

[19] Bruckstein, A.M. and Elad, M. and Zibulevsky, M., "Sparse non-negative solution of a linear system of equations is unique," in *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*, pp., 762–7.

[20] de Fréin, R., "Ghostbusters: A parts-based NMF algorithm," in *24th IET ISSC*, pp., 1–8.

[21] Ziv, J. and Lempel, A., "Compression of individual sequences via variable-rate coding," *Inf. Th., IEEE Trans.*, vol. 24, no. 5, pp. 530–6, Sep. 1978.

[22] Welch, T.A., "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, June 1984.

[23] Cover, Thomas M. and Thomas, Joy A., *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*, Wiley-Interscience, 2006.

[24] Lee, Daniel D. and Seung, H. Sebastian, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–91, Oct 1999.

[25] Schobben, D. W E and Beuker, R.A. and Oomen, W., "Dither and data compression," *Sig. Proc., IEEE Trans.*, vol. 45, no. 8, pp. 2097–2101, Aug. 1997.

[26] Gish, H. and Pierce, J., "Asymptotically efficient quantizing," *Inf. Th., IEEE Trans.*, vol. 14, no. 5, pp. 676–83, Sep. 1968.