2012 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 23–26, 2012, SANTANDER, SPAIN

PROBABILISTIC INTERPOLATIVE DECOMPOSITION

İsmail Arı, A. Taylan Cemgil, Lale Akarun

Bogazici University, Computer Engineering Department, 34342 Bebek, Istanbul, Turkey

ABSTRACT

Interpolative decomposition (ID) is a low-rank matrix decomposition where the data matrix is expressed via a subset of its own columns. In this work, we propose a novel probabilistic method for ID where it is expressed as a statistical model within a Bayesian framework. The proposed method considerably differs from other ID methods in the literature: It handles the model selection automatically and enables the construction of problem-specific interpolative decompositions. We derive the analytical solution for the normal distribution and we provide a numerical solution for the generic case. Simulation results on synthetic data are provided to illustrate that the method converges to the true decomposition, independent of the initialization; and it can successfully handle noise. In addition, we apply probabilistic ID to the problem of automatic polyphonic music transcription to extract important information from a huge dictionary of spectrum instances. We supply comparative results with the other proposed techniques in the literature and show that it performs better. Probabilistic interpolative decomposition serves as a promising feature selection and de-noising tool to be exploited in big data problems.

Index Terms— Interpolative decomposition, CUR Decomposition, Bayesian inference, SVD, Simulated Annealing, Importance Sampling, Polyphonic Music Transcription.

1. INTRODUCTION

In *interpolative decomposition (ID)*, the aim is to represent a matrix by a subcollection of its columns [1]. In particular, given a matrix with N columns, we would like to select k linearly independent columns that span the column space of the matrix. Hence, ID is also called *spanning columns*. It is usually applied also on the rows of the matrix and so, two IDs can be combined to construct the matrix using a subcollection of its columns and a subcollection of its rows (*CUR decomposition*) [1, 2].

The main motivation in interpolative decomposition (and CUR decomposition) is to determine fewer number of columns (rows) out of a huge collection of columns (rows) that approximate the range (corange) of the data matrix [1]. It serves as a feature selection tool that extracts the essence and enables working with big data which is originally too large to fit into the RAM. In addition, it removes the non-relevant parts of the data which consist of error and redundant information. Compared to SVD, CUR is better in terms of reification issues since it uses the actual columns (rows) of the matrix whereas SVD uses some artificial singular vectors that may not represent physical reality [3]. Another important advantage is that CUR maintains sparsity if the data is sparse. Like SVD, CUR decomposition may be used as a tool for data compression, feature extraction or data analysis in many application areas [3, 4, 5].

As stated, ID is studied implicitly in CUR decomposition. Thus we give related research in the context of CUR decomposition which is studied extensively in the literature [1, 6, 7]. Proposed methods mainly differ in their way on the selection of column/row indices. Halko *et al.* use plain column-pivoted Gram-Schmidt method based on the vector norms [1]. This algorithm selects the vector with the largest norm, then it projects all the remaining vectors onto the complement of the spanning space of this selected vector and selects the one with the largest norm in this new space in an iterative manner. Other methods can be grouped as Monte Carlo (MC) sampling based methods where each column/row is assigned a probability proportional to its Euclidean norm [8, 9], norm of right singular vectors [3] or vector sparseness value [5]. Then, corresponding indices are selected randomly from a multinomial distribution of these probabilities. In [10], Bien *et al.* investigate CUR from a sparse optimization viewpoint. They show that CUR is implicitly optimizing a sparse regression objective, and compare it to sparse PCA. A randomized algorithm for CUR approximation with controlled absolute error is given in [8] and a relative error algorithm appears in [11].

On the other hand, matrix decompositions can also be seen as statistical models where we seek for the decomposition that gives the maximum marginal likelihood (MML) for the underlying data [12]. Probabilistic interpretations are investigated for many popular matrix decompositions in the literature such as Non-negative Matrix Factorization (NMF) [13] and Principal Component Analysis (PCA) [14], and generalized to tensor factorizations [15].

A better interpretation of ID (and CUR decomposition) is

This work has been supported by the TUBITAK project 108E161. A.T.C. is funded by TUBITAK grant 110E292.

978-1-4673-1026-0/12/$31.00

crucial for any sequent algorithm that makes use of the selected indices. This work aims to express interpolative decomposition as a statistical model within a Bayesian framework. For the maximization of the MML, we use a Simulated Annealing technique. We derive the analytical solution for the normal distribution and we generalize the approach for other observation models by providing numerical solution using importance sampling. The method can easily be adapted for problem-specific interpolative decompositions by giving prior knowledge about the data and choosing suitable distributions. In addition, model order selection (determination of the number of columns and rows) to represent the data is handled automatically within the proposed system via Bayesian inference. Our work, in our knowledge, is the first interpolative decomposition and CUR decomposition technique using Bayesian inference: Previous probabilistic approaches use heuristics to select bases and the number of bases is known a priori [1, 2, 5, 8, 9]. We do not explicitly discuss error bounds since the proposed approach differs from the works in the literature in the way that it already enables modeling of the error. We perform experiments on synthetic and real life data. We apply probabilistic ID to reduce the size of the training data in the problem of polyphonic music transcription by removing more than 99% of the samples and yet maintain successful results.

2. PROBABILISTIC INTERPOLATIVE DECOMPOSITION AND MODEL SELECTION

The aim in interpolative decomposition (ID) is to represent the range of a matrix $\mathbf{X} \in \mathbb{R}^{d\times N}$ using k of its columns ($k \ll N$). The selected vectors stand for the bases and the remaining ones are expressed as linear combinations of them. The decomposition can fully represent the matrix by selecting columns that span the whole column space of $\mathbf{X}$ as well as approximating it with fewer number of columns than its rank. That is,

$$\mathbf{X} \approx \mathbf{CZ} \tag{1}$$

where $\mathbf{C}$ is a submatrix of columns of $\mathbf{X}$, and $\mathbf{Z}$ involves the interpolation coefficients corresponding to each column in $\mathbf{X}$. Let J be the set of the indices of the selected columns. Then, we write $\mathbf{C} = \mathbf{X}(:, J)$ where the colon operator implies all indices.

Let $\mathbf{x}_n$ be the n^{th} column of $\mathbf{X}$ and let $\mathbf{r} \in \{0, 1\}^N$ be the state vector with each element indicating the type of the corresponding column. If $r_n = 1$, then $\mathbf{x}_n$ is a basis column. If $r_n = 0$, then $\mathbf{x}_n$ is interpolated using the basis columns plus some error term. In addition to J, let $\tilde{J}$ be the set of indices of the remaining columns. The relation between $\mathbf{r}$ and the index sets is simple; $J = J(\mathbf{r}) = \{n|r_n = 1\}_{n=1}^N$ and $\tilde{J} = \tilde{J}(\mathbf{r}) = \{n|r_n = 0\}_{n=1}^N$. The generative model for ID is given in Fig 1: Columns J of the data matrix are the same as the base matrix, that is $\mathbf{X}(:, J) = \mathbf{C}$. The remaining columns are generated using as a linear combination of the bases plus some error term: $\mathbf{X}(:, \tilde{J}) = \mathbf{CZ} + \mathbf{E}$.

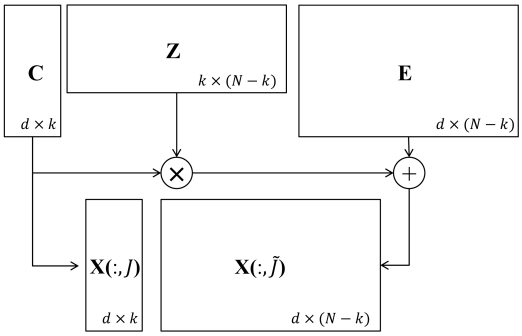


Fig. 1. Generative model for interpolative decomposition

We are interested in the state $\mathbf{r}^*$ giving the maximum marginal likelihood for the underlying data $\mathbf{X}$:

$$\mathbf{r}^* = \underset{\mathbf{r}}{\operatorname{argmax}}\, p(\mathbf{X}, J(\mathbf{r})) \tag{2}$$

$$= \underset{\mathbf{r}}{\operatorname{argmax}}\, p(\mathbf{X}|J(\mathbf{r}))p(J(\mathbf{r})) \tag{3}$$

Let each combination be equally likely and let us marginalize the conditional probability using the generative model given in Fig. 1 as follows and call it ℓ:

$$\ell = p(\mathbf{X}|J(\mathbf{r})) = p(\mathbf{X}_J, \mathbf{X}_{\tilde{J}}) = p(\mathbf{X}_{\tilde{J}}|\mathbf{X}_J)p(\mathbf{X}_J)$$
$$= \iint p(\mathbf{X}_{\tilde{J}}|\mathbf{C}, \mathbf{Z})p(\mathbf{X}_J|\mathbf{C})p(\mathbf{Z})p(\mathbf{C})\, d\mathbf{Z}\, d\mathbf{C} \tag{4}$$

where $\mathbf{X}_J = \mathbf{X}(:, J)$ and $\mathbf{X}_{\tilde{J}} = \mathbf{X}(:, \tilde{J})$. With $p(\mathbf{X}_J|\mathbf{C}) = \delta(\mathbf{X}_J - \mathbf{C})$, we get:

$$\ell = \iint p(\mathbf{X}_{\tilde{J}}|\mathbf{C}, \mathbf{Z})\delta(\mathbf{X}_J - \mathbf{C})p(\mathbf{Z})p(\mathbf{C})\, d\mathbf{Z}\, d\mathbf{C}$$
$$= \int p(\mathbf{X}_{\tilde{J}}|\mathbf{X}_J, \mathbf{Z})p(\mathbf{Z})p(\mathbf{X}_J)\, d\mathbf{Z} \tag{5}$$

2.1. Analytical Solution

We may choose particular distributions for the interacting factors to acquire different decompositions for the data. The MML can be computed analytically for certain kinds of distributions. We give an example case here and give the generic approach as a numerical technique in the next subsection. Let j be the index traversing through the indices in J, and similarly, $\tilde{j}$ be the index traversing through the indices in $\tilde{J}$. Assume the columns are independent; then Eq. 5 becomes:

$$\ell = \prod_{\tilde{j}\in\tilde{J}} \left(\int p(\mathbf{x}_{\tilde{j}}|\mathbf{X}_J, \mathbf{z})p(\mathbf{z})\, d\mathbf{z} \right) \prod_{j\in J} p(\mathbf{x}_j) \tag{6}$$

The interpolated vectors can be written in vector form as $\mathbf{x}_{\tilde{j}} = \mathbf{X}_J \mathbf{z}_{\tilde{j}} + \epsilon_{\tilde{j}}$. Let $\mathbf{x}_j$, $\mathbf{z}_{\tilde{j}}$, and $\epsilon_{\tilde{j}}$ be distributed with zero mean Gaussians and the variances be $\mathbf{\Sigma}_J = \sigma_J^2 \mathbf{I}$, $\mathbf{\Sigma}_z = \sigma_z^2 \mathbf{I}$, and $\mathbf{\Sigma} = \sigma^2 \mathbf{I}$, respectively. The likelihood is computed on the centralized data and is as follows:

$$\ell \propto \prod_{\tilde{j} \in \tilde{J}} \exp\left(-\frac{1}{2} \mathbf{x}_{\tilde{j}}^\intercal \mathbf{\Sigma}_{\tilde{J}}^{-1} \mathbf{x}_{\tilde{j}}\right) \prod_{j \in J} \exp\left(-\frac{1}{2\sigma_J^2} \|\mathbf{x}_j\|^2\right) \quad (7)$$

where $\mathbf{\Sigma}_{\tilde{J}} = \mathbf{X}_J \mathbf{\Sigma}_z \mathbf{X}_J^\intercal + \mathbf{\Sigma} = \sigma_z^2 \mathbf{X}_J \mathbf{X}_J^\intercal + \sigma^2 \mathbf{I}$. The log-likelihood, which is numerically more stable, is

$$\mathcal{L} \propto \sum_{\tilde{j} \in \tilde{J}} \Big(-\frac{1}{2} \mathbf{x}_{\tilde{j}}^\intercal \mathbf{\Sigma}_{\tilde{J}}^{-1} \mathbf{x}_{\tilde{j}} \Big) + \sum_{j \in J} \Big(-\frac{1}{2\sigma_J^2} \|\mathbf{x}_j\|^2 \Big) \quad (8)$$

2.2. Numerical solution

We use Monte Carlo sampling to find an approximation when we cannot find an analytical solution to precisely compute the integral. We take N_z samples of $\mathbf{z}$ and compute the marginal likelihood. Sampling $\mathbf{z}$ values directly from the distribution with pdf $p(\mathbf{z})$ can lead to a large variance in the computed expected values. Instead, we use *importance sampling* where we sample from another distribution with pdf $q(\mathbf{z})$ and weigh the values.

$$\ell \propto \prod_{\tilde{j} \in \tilde{J}} \left(\frac{1}{N_z} \sum_{i=1}^{N_z} p(\mathbf{x}_{\tilde{j}} | \mathbf{z}_i, \mathbf{X}_J) \frac{p(\mathbf{z}_i)}{q(\mathbf{z}_i)} \right) \prod_{j \in J} p(\mathbf{x}_j) \quad (9)$$

In order to decrease the variance, a good way of sampling $\mathbf{z}_{\tilde{j}}$ is to use the information of $\mathbf{X}_J$ and $\mathbf{x}_{\tilde{j}}$ values. A good proposal distribution is $\mathbf{z}_{\tilde{j}} | \mathbf{X}_J, \mathbf{x}_{\tilde{j}}$ where $\mathbf{z}_{\tilde{j}}$ comes from a normal distribution with expectation $E[\mathbf{z}_{\tilde{j}}] = \mathbf{X}_J^\dagger \mathbf{x}_{\tilde{j}}$ and covariance matrix $\mathrm{Cov}(\mathbf{z}_{\tilde{j}}) = \mathbf{X}_J^\dagger \mathbf{\Sigma} (\mathbf{X}_J^\dagger)^\intercal$ for the given case.

2.3. Simulated annealing

The problem of finding the state $\{r_t\}_{t=1}^N$ giving the MML is combinatorial; i.e., we have 2^N different states to check. When we have two candidate states, we can decide which one is more likely the model. Simulated Annealing, which is a Markov Chain Monte Carlo method, is used for finding the state giving the MML in our setup. We start with a random state $\mathbf{r}$ and check the likelihood values $\{\ell(\mathbf{r}_i)\}_{i=1}^n$ of its neighbors. $\mathbf{r}'$ is a state randomly chosen among the neighbors proportional to their likelihoods. Acceptance probability in simulated annealing is taken as $\max\{1, \exp\left(-\rho\left(\ell(\mathbf{r}') - \ell(\mathbf{r})\right)\right)\}$ where the cooling coefficient $\rho = \frac{i}{m}$ in the i^{th} iteration, where m is the maximum number of iterations. The algorithm lets the search to branch into less likely states in the first runs but it favors staying in the best state found in later stages. It is stopped if the result converges or an allowed maximum number of iterations are passed [16].

The neighbors of a state are the states that can be accessed by one column release and/or one column addition. For example, for $N = 3$ and $\mathbf{r} = 010$, the neighbors are 000 (a release), 100 and 001 (an index change), and 110 and 011 (an addition). At each iteration we compute the likelihoods for $k_i + k_i(N - k_i) + (N - k_i) = N(k_i + 1) - k_i^2$ states. We can assume the expected value $\mathbb{E}(k_i) \approx k$ since k_i values converge to k, and thus, worst case complexity is $O(mNk)$ steps.

The model order k is determined automatically within the algorithm. At each iteration, it stays the same or is incremented/decremented by 1 according to the change in the likelihood. We first find the eigenvalues λ_i of $\mathbf{X}$ and the initial number of dimensions is randomly assigned using probabilities proportional to λ_i. So, smaller k values are more likely to be given initially for matrices whose variation is concentrated in the first few eigenvectors. For big data, we use randomized PCA [17].

2.4. Probabilistic CUR Decomposition

To express $\mathbf{X} \approx \mathbf{CUR}$, we apply ID on $\mathbf{X}$ to compute a subcollection of columns $\mathbf{C}$ and similarly on $\mathbf{X}^\intercal$ to compute a subcollection of rows $\mathbf{R}$. Then, the small linkage matrix $\mathbf{U}$ is found via solving a small least squares problem [1].

3. PROBABILISTIC ID BASED POLYPHONIC MUSIC TRANSCRIPTION

As an example application, we work on automatic music transcription that is one of the fundamental problems studied in the field of audio processing. Given polyphonic music, the aim is to recognize the notes and the time interval in which they are active. In this section, we are interested in employing probabilistic ID to reduce the size of the big training data while keeping the transcription algorithm unchanged. The work is built up on our previous work [4]. Once the data is reduced, other polyphonic music transcription algorithms may make use of this reduction as well. We briefly summarize the transcriber here and show how to make it efficient via probabilistic ID.

Inspired partially by the idea that human listeners become more successful with training in recognizing musical constructs, Smaragdis has demonstrated that it is possible to perform polyphonic pitch tracking successfully via a linear model that tries to approximate the observed musical data as a superposition of previously recorded monophonic musical data [18]: $\mathbf{Y} \approx \mathbf{XW}$ where $\mathbf{Y}$ is the observed spectrogram, $\mathbf{X}$ is the dictionary matrix obtained from the training data, and $\mathbf{W}$ contains the corresponding weights. Let $\mathbf{X}_i$, with elements $X_i(f, \tau_i)$, denote the magnitude spectrogram of monophonic music recordings belonging to I different notes. Here, $i = 1, \ldots, I$ is the note index, $f = 1, \ldots, F$ is the frequency index, and $\tau_i = 1, \ldots, N_i$ is the time index where

F is the number of frequency bins and N_i is the number of columns in $\mathbf{X}_i$. We remove the effect of sound intensity by normalizing each vector such that its elements sum up to 1 since we are interested only in the pitch of the data. We also remove the samples below some loudness level. We obtain the training data by concatenating all training vectors side by side where there are a total number of $N = \sum_{i=1}^{I} N_i$ training samples. Test data are composed of polyphonic recordings. Let $\mathbf{Y}$, with values $Y(f,t)$, be the spectrogram of the test data where $t = 1, \ldots, T$ and T is the number of time frames.

The transcription algorithm uses a basic linear model that describes the relationship between the observed and the training data where the observed spectrum is expressed as a superposition of the training vectors:

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = \mathbf{X}\mathbf{W} \tag{10}$$

The aim is to find the weight matrix $\mathbf{W}$ which minimizes $\mathcal{D}\left[\mathbf{Y} \| \mathbf{X}\mathbf{W}\right]$ where $\mathcal{D}[\cdot\|\cdot]$ is a properly selected cost function. KL divergence is used as the cost function as described in [4]. We start with random initialization of $\mathbf{W}$, and continue with the following step until convergence:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \left(\frac{\mathbf{X}^\intercal \frac{\mathbf{Y}}{(\mathbf{X}\mathbf{W})}}{\mathbf{X}^\intercal \mathbb{1}} \right) \tag{11}$$

The $\odot$ symbol denotes element-wise multiplication and the division operation is also done element-wise. $\mathbb{1}$ is a matrix of all ones of the same size with $\mathbf{Y}$. Active notes are selected by thresholding. Additionally, each weight value is smoothed by applying a median filter to remove sharp jumps (See [4] for other details).

3.1. Improving efficiency via probabilistic ID

In real world applications the training data may be composed of millions of columns, and it may not fit into the RAM. Our aim is to make the method efficient in terms of both time and space complexity. The key point is to determine which columns of $\mathbf{X}$ carry the important information. We first apply PCA on the unnormalized spectrogram matrix $\mathbf{X}_i$ of each note and reduce the dimension by keeping only the coefficients corresponding to the eigenvectors representing 99% variance of the data. Let $\tilde{\mathbf{X}}_i$ involve the coefficients in the transformed space. Then, we apply probabilistic ID on $\tilde{\mathbf{X}}_i$ and find J_i and $\mathbf{C}_i$. Then, we merge $\mathbf{C}_i \ \forall i$ and get $\mathbf{C}$. As a result, we replace $\mathbf{C}$ with $\mathbf{X}$ in the rule:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \left(\frac{\mathbf{C}^\intercal \frac{\mathbf{Y}}{(\mathbf{C}\mathbf{W})}}{\mathbf{C}^\intercal \mathbb{1}} \right) \tag{12}$$

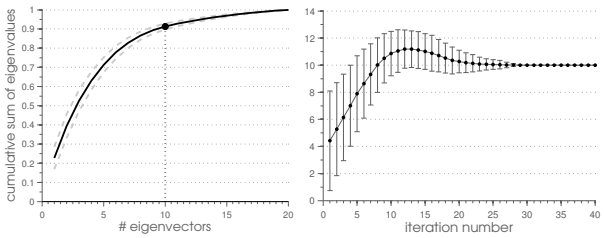
Since we discard most of the data and keep only the important information, the computational gain is very high in practice. The number of columns sufficient to approximate the range of the full matrix may be hundreds of times less the original size. For a realistic case, if we have $N \approx 10^5$, and we choose $k = 800$, the algorithm becomes nearly 125 times faster and we use only about 0.8% of the data. This method can work in real time and handle large amount of data which can not be handled by conventional methods. Alternatively, $\mathbf{C}$ can be computed via other CUR methods in the literature. The gain the space & time complexities will be the same, but the success of the algorithm will vary. We conduct experiments and provide comparative results in the next section.

4. EXPERIMENTS AND RESULTS

In order to evaluate our methods, we have conducted several experiments. First, we perform experiments on synthetic data to show that the technique converges to an approximation of the desired decomposition independent of the initial state; i.e., it is stable. Second, we perform experiments on real music data to show how to efficiently perform automatic polyphonic music transcription via probabilistic ID. Additionally, we compare the proposed technique with two popular techniques in the literature.

4.1. Experiments with synthetic data

A synthetic experiment is performed to show how probabilistic ID can automatically determine k. We first create 10 vectors using $\sigma_J = 1$. Then, we interpolate 90 vectors using $\sigma_z = 0.3$ and add noise using $\sigma = 0.75$. Fig. 2a shows the normalized cumulative sum of eigenvalues, $\sum_j^i \lambda_j / \sum_j^N \lambda_j$, found on this matrix using Eq. 7. The rank is actually 10 but it is difficult to determine since the added error spans the remaining space. Fig. 2b shows the selected number of indices k where we observe that it converges to $k = 10$ after 30 iterations.



(a) (b)

Fig. 2. Simulation results on noisy data where rank is not clear: (a) normalized cumulative sum of eigenvalues and (b) mean and variance of the best estimates of the number of the selected indices. This experiment is repeated 100 times and the bars around the points indicate the 95% confidence intervals for cumulative sum (a) and k_i (b). The decomposition gets stable after 30 iterations and the final value of k is found as 10.

4.2. Polyphonic Music Transcription

In our real data experiments, we have used the MAPS (MIDI Aligned Piano Sounds) data set [19]. The setup is similar to [4]: The training set is obtained using 440 monophonic piano sounds of 44.1 kHz sampling rate where we represent the audio by its magnitude spectrogram which is computed via DFT. The spectrogram is formed using overlapping Hanning windows of dimension 2048 with a window shift of 512. While constructing $X_i(f, \tau_i)$, we simply concatenated the spectra corresponding to the i^{th} note where $i = 1, \ldots, 88$. Note that 88 is the number of keys on a piano. About one third of the training data is removed due to low loudness and the obtained final dictionary matrix is 1025×115600 and is around 860 MB.

A test set is formed by choosing random sections from five different polyphonic pieces. It can be said that the test data is predominantly polyphonic since he polyphony orders (the number of notes played simultaneously at any time instant) in the test data are observed to be nearly equally distributed among the first five values.

In order to evaluate the performance of the methods we used precision (fraction of retrieved instances that are relevant), recall (fraction of relevant instances that are retrieved) and f-measure $= \frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$ metrics.

The full solution is found using Eq. (11) and an f-measure of 78.07% is obtained (See [4] for a discussion on the result and a comparison with other polyphonic music transcription methods in the literature).

Next, we compute the ID of the dictionary. Since all the columns are valid spectrum vectors, we assumed the prior to be uniform, i.e. $p(\mathbf{x}_j) = \frac{1}{N}$ for all j.

We observe that σ_z controls the selected number of indices: The larger the σ_z, the smaller the k. Probabilistic ID is performed for each note by taking $\sigma_z = 0.2$ and σ as the quarter of the standard deviation of all data. After merging the selected indices per note, a total number of 778 columns out of 115600 are extracted. The algorithm is performed as given in Eq. (12) and the result is given in Fig. 3. We only use about 0.8% of the data; the speed is nearly 125 times faster, and we get an f-measure of 76.54% which is close to the result of the full solution.

In addition to probabilistic ID, we performed the same method using two alternative ID methods. First method is the randomized CUR proposed by Mahoney and Drineas [3] which is based on the norms of right singular vectors obtained from the matrix to be decomposed. Their algorithm randomly selects k vectors proportional to these norms. The aim here is to sample the columns which reside on the large variation directions and so, minimize the reconstruction error. Second alternative is the column-pivoted Gram-Schmidt method given in [1]. This algorithm selects the largest vector of the matrix, then it projects all the vectors onto the complement of the spanning space of this selected vector and selects the one with the largest norm in this new space. It can be thought as a greedy approach. As shown in Fig. 3, f-measure values are found to be 75.18% and 74.70% for the two alternative methods, respectively. For both methods, k should be given explicitly. We use $k = 9$ for each note, and the total number of columns in $\mathbf{C}$ is found as $9\times 88 = 792$ which is close to the number of columns found by probabilistic ID. These methods are good candidates for constructing a low rank approximation to the full matrix in terms of l_2 (or *frobenius*) norm. But we see that this does not necessarily imply a good candidate for feature selection. Probabilistic ID proves to be a good way to determine which columns are more important.

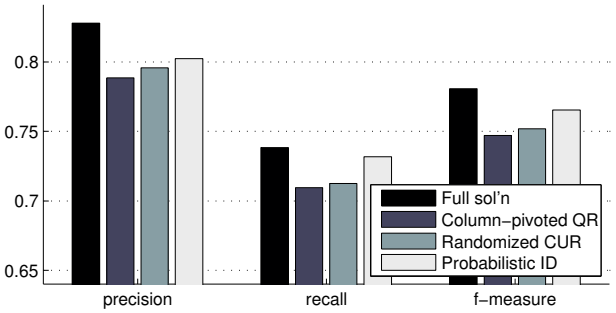


Fig. 3. Results obtained on the test set per approach: Probabilistic ID seems to perform close to the full solution and better than the alternative ones in the literature (Randomized CUR [3] and column-pivoted QR [1]).

In addition to the overall results, we give the results for each polyphony order in Fig. 4 for the probabilistic ID approach. As can be seen, recall rate is perfect in the monophonic case but the precision is low. The precision gets better while polyphony increases. Using a higher threshold on the weights may lead to selection of fewer notes as active and this can improve precision with a trade-off in recall rate if one is more interested in precision.

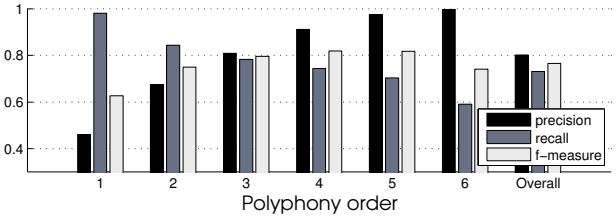


Fig. 4. Precision, recall and f-measures for each polyphony order: The figure shows the results for the probabilistic ID approach. F-measure seems to be more than 60% for each polyphony order.

5. CONCLUSION

In this work, a generic probabilistic approach to interpolative decomposition and CUR decomposition is established and supported with examples of analytical and numerical ways to solve the problem. The method can easily be adapted to get problem-specific interpolative decompositions and the number of selected columns is determined automatically within the model. The advantage of ID is that the actual columns are selected as bases instead of some artificial linear combinations as in SVD. Furthermore, it favors sparsity if the given matrix is sparse.

Experiments are conducted on both synthetic and real life data to test the proposed method. It is shown that the method is stable and it converges to the desired decomposition independent of the initial state. The experiments on polyphonic music transcription show that probabilistic ID stands to be a good technique for determining the important instances among the data and remove the redundant and non-relevant parts. We keep only 0.8% of the data matrix and it performs very close to the results obtained by using the full matrix. Furthermore, it seems to perform better than the methods used in the literature. Like other probabilistic approaches, it suffers from the running time issues compared to the randomized algorithms. But this is not a significant concern in our case since the decomposition is done only in the training phase and the selected instances are used during the test.

This work shows a novel approach to ID and CUR decomposition using Bayesian inference. As a by-product, we obtain a method for automatically selecting the rank of the approximation of the data matrix. As future work, we aim to analyze different distributions in this framework such as gamma distribution to add non-negativity constraints on the factors.

6. REFERENCES

[1] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions," *SIAM Review*, 2011.

[2] M. W. Mahoney, "Randomized Algorithms for Matrices and Data," *Foundations and Trends in Machine Learning*, pp. 123–234, 2011.

[3] M. W. Mahoney and P. Drineas, "CUR Matrix Decompositions for Improved Data Analysis," *Proc. of the National Acad. of Sci.*, vol. 106, no. 3, pp. 697–702, 2009.

[4] İ. Arı, U. Şimşekli, A. T. Cemgil, and L. Akarun, "Large Scale Polyphonic Music Transcription Using Randomized Matrix Decompositions," in *EUSIPCO*, 2012.

[5] H. Lee and S. Choi, "CUR+NMF for Learning Spectral Features from Large Data Matrix," in *IEEE Int'l Joint Conf. on Neural Networks*, 2008, pp. 1592–1597.

[6] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, "A Theory of Pseudoskeleton Approximations," *Linear Alg. and its App.*, vol. 261, no. 1-3, 1997.

[7] G. W. Stewart, "Four Algorithms for the Efficient Computation of Truncated Pivoted QR Approximations to a Sparse Matrix," *Num. Mathematik*, vol. 83, no. 2, 1999.

[8] P. Drineas, R. Kannan, and M. W Mahoney, "Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition," *SIAM Journal on Computing*, vol. 36, pp. 184–206, 2007.

[9] A. Frieze, R. Kannan, and S. Vempala, "Fast Monte-Carlo Algorithms for Finding Low-rank Approximations," *Journal of the ACM*, pp. 1025–1041, 2004.

[10] J. Bien, Y. Xu, and M. W. Mahoney, "CUR from a Sparse Optimization Viewpoint," in *Advances in Neural Information Processing Systems*, 2010.

[11] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Relative-Error CUR Matrix Decompositions," *SIAM Journal on Matrix Analysis and Appl.*, vol. 30, 2008.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007.

[13] C. Févotte and A. T. Cemgil, "Nonnegative Matrix Factorizations as Probabilistic Inference in Composite Models," in *EUSIPCO*, 2009, pp. 1913–1917.

[14] M. E. Tipping and C. M. Bishop, "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[15] Y. K. Yilmaz and A. T. Cemgil, "Algorithms for Probabilistic Latent Tensor Factorization," *Signal Processing*, vol. 92, no. 8, pp. 1853–1863, 2012.

[16] C. Robert and G. Casella, *Monte Carlo Statistical Methods*, Springer, 2 edition, 2004.

[17] N. Halko, P. G. Martinsson, Y. Shkolnisky, and M. Tygert, "An Algorithm for the Principal Component Analysis of Large Data Sets," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2580, 2011.

[18] P. Smaragdis, "Polyphonic Pitch Tracking by Example," in *IEEE WASPAA*, 2011, pp. 125–128.

[19] V. Emiya, R. Badeau, and B. David, "Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.