Multi-Channel Similarity Based Compression

Sergey Voronin[1]

[1] Amathsolutions, United States of America

Correspondence: svoronin@amathsolutions.org, United States of America.

Received: November 18, 2019 Accepted: January 2, 2020 Online Published: January 20, 2020

doi:10.5539/cis.v13n1p80 URL: https://doi.org/10.5539/cis.v13n1p80

Abstract

Many situations arise where data is collected continuously across multiple channels or over multiple similar subjects. In many cases, transmission of the data across all channels is necessary, but the process can be made more efficient by making use of present similarity between data across different channels. We present here a combined compression approach which exploits approximate linear dependence and high correlation coefficient values between pairs of transformed and sorted channel data vectors. By exploiting this similarity, substantial compression gains can be achieved compared to compression of data per each individual channel.

Keywords: similarity compression, interpolative decomposition, high correlation modeling

1. Introduction

Data compression is of ever growing importance, with increasing data sizes driven by the use of high sensitivity sensors recording in high definition formats. There are many examples of multi-channel data which needs to be stored and/or transmitted for analysis such as recordings with acoustic arrays or medical sensors. Two particular examples we will analyze are acoustic recordings and electrocardiogram (ECG) data. In both cases, there are many sets of long data vectors of floating point numbers, which exhibit some similarity when plotted. Storing these vectors individually for each channel and even compressing them with e.g. deflate or Burrows-Wheeler type lossless methods, would yield large output sizes. However, there are two similarity aspects which can be exploited. One is approximate linear dependence. When these vectors across multiple channels are put together in a matrix, the singular value decay would typically be nonlinear. Specialized low rank factorizations such as the Interpolative Decomposition can be adapted to this purpose (Voronin & Martinsson , 2017). In case the data is lagged from channel to channel, such as in case of acoustic recordings made with sensors a significant distance apart, lag adjustments can be performed prior to matrix formation and small portions of data per channel can be retained for separate processing. The other aspect which can be exploited is high correlation of transformed and sorted data (Gutton et al. , 2007). In many cases, for every channel, the absolute values of the sorted transform coefficients would display an exponentially decaying shape. Upon application of the log transformation, the decay becomes approximately linear and a high correlation between nearby pairs of coefficients can be exploited via a low order polynomial model for the data of one of the channels in terms of the other. To improve performance, the coefficient curve can be subdivided in several parts, and the log transformed portions can be separately fitted with respect to the corresponding part of one of the retained transformed reference signals.

1.1 Acoustic and ECG Examples

Some examples to which our approach is applicable come from generated acoustic and ECG data sets. In particular, the multi-channel acoustic recording system is simulated by means of the k-wave (Treeby & Benjamin , 2010) tools (`http://www.k-wave.org`) and the ECG data was obtained from the PhysioNet (Moody et al. , 2001) (`https://physionet.org/content/ptbdb/1.0.0/`) database. Examples are shown in Figure 1. There, a sample acoustic geometry is shown with a main emitter source, nearby interference, and receiver positioning (arranged as a vertical array of black dots on the right). Some of the measured channels are then plotted. Also shown are the processed ECG signals from the above database coming from a multi-patient data set. Throughout this document we refer to channel as an individual entity, which can be a physical object (such as a microphone recorder) or patient for which data is obtained. The goal at hand is to store floating point data efficiently across all channels.

1.2 Singular Value Decay and Transformed Signal Correlation

For both applications, a combined strategy taking advantage of high degree of linear dependence and correlation under transformation can be employed to improve on the simple compression ratio, obtained by lossless compressing the floating point data for each channel. Figure 2 plots the associated singular value decay and pairwise correlation values under transformation of the first 350 ECG signals, of a set of 550. That is, we have taken a matrix consisting of 550 rows, with one ECG signal per row, and computed its singular values. We observe that the decay is nonlinear, suggesting that a

suitable low rank factorization can be utilized to store the data more efficiently. Next, we compute a four level CDF 9/7 Wavelet transform of each data row. We then sort the absolute values of the resulting coefficients for each row, take the log transformation, and compute their pairwise correlation coefficient values. We can see that many pairs of transformed and sorted coefficient vectors have high values (> 0.95), which suggests that the coefficients vectors for a few channels can be stored and the rest represented by a linear ($ax + b$) or low order polynomial model with respect to one of the stored vectors x. With the additional storage of the permutation set from the sort and the vector of signs, the original transformed coefficients for the remaining channels can be recovered.

2. Method

We present here a combined approach based on the observations noted in section 1.2. We discuss the relevant low rank factorization and the approaches for high correlation based compression via low order polynomial modeling against retained pillar data.

2.1 Interpolative Decomposition

There are several available matrix decompositions which can exploit numerical rank deficiency in a matrix of similar signals (Voronin & Martinsson , 2017). We review here the low rank singular value decomposition (SVD) and the interpolative decomposition (ID), illustrated in Figure 2. The rank-k SVD ($A_k = U_k \Sigma_k V_k^T$) of a general $m \times n$ matrix A yields an optimal approximation of rank k to A, in the sense that $\|A - A_k\| \leq \|A - M_k\|$ for any rank k matrix M_k, both in the operator (spectral) and Frobenius norms. However, with the use of the SVD, the eigenvectors may often be difficult to interpret and do not contain direct entries of the signals in the matrix. The ID is a different factorization based on the QR decomposition, which returns a factor containing a selection of columns (or rows) of the original matrix. We can start with the pivoted QR decomposition, $AP = QS$, which can be written as $A(:, J_c) = QS$ for a column index J_c. From this, the rank-k ID can be efficiently constructed, with details given in (Voronin & Martinsson , 2017). Below we outline the main steps from expanding the Q and S factors into portions:

$$A(:, J_c) \quad = \quad {}_m \begin{bmatrix} {\scriptstyle k} & {\scriptstyle r-k} \\ Q_1 & Q_2 \end{bmatrix} \times {}_{\substack{k \\ r-k}} \begin{bmatrix} {\scriptstyle n} \\ S_1 \\ S_2 \end{bmatrix} = Q_1 S_1 + Q_2 S_2.$$

$$S_1 = {}_k \begin{bmatrix} {\scriptstyle k} & {\scriptstyle n-k} \\ S_{11} & S_{12} \end{bmatrix} \quad \text{and} \quad S_2 = {}_k \begin{bmatrix} {\scriptstyle k} & {\scriptstyle n-k} \\ 0 & S_{22} \end{bmatrix}, \quad \text{such that} \quad S = {}_{\substack{k \\ r-k}} \begin{bmatrix} {\scriptstyle k} & {\scriptstyle n-k} \\ S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}.$$

Inserting the above expressions for S_1, S_2 into $A(:, J_c)$, we get:

$$A(:, J_c) \quad = \quad Q_1 \begin{bmatrix} S_{11} & S_{12} \end{bmatrix} + Q_2 \begin{bmatrix} 0 & S_{22} \end{bmatrix} = {}_m \begin{bmatrix} {\scriptstyle k} & {\scriptstyle n-k} \\ Q_1 S_{11} & Q_1 S_{12} + Q_2 S_{22} \end{bmatrix}.$$

Next, we set $C := A(:, J_c(1:k)) = Q_1 S_{11}$, so that:

$$Q_1 S_1 = \begin{bmatrix} Q_1 S_{11} & Q_1 S_{12} \end{bmatrix} = Q_1 S_{11} [I_k \quad S_{11}^{-1} S_{12}] = C \, [I_k \quad T_l],$$

where T_l is the solution to the matrix equation $S_{11} T_l = S_{12}$, a set of matrix-vector systems. This results in the approximate factorization:

$$A \approx C V^T, \quad \text{where} \quad C = A(:, J_c(1:k)), \quad V^T = \begin{bmatrix} I_k & T_l \end{bmatrix} P^T,$$

with C a subset of the columns of A based on the pivoting strategy in the QR factorization and the remaining V factor a well-conditioned matrix, a portion of which is a diagonal identity sub-matrix. The permutation matrix P does not need be formed explicitly and is represented by vector J_c. With access to A, this vector and matrix T_l give complete information for the ID. The V factor can be constructed as below:

```
Iinv=zeros(length(I),1);
for i=1:length(I)[ ind = I(i); Iinv(ind) = i; ] end;
V1 = [I_k; T']; V = V1(Iinv,:);
```

The runtime can be accelerated by employing randomization at the expense of accuracy, which essentially projects the matrix A from the left (RA) and uses the smaller matrix for the ID. The alternative is to build a QB factorization (Martinsson & Voronin , 2016) and obtain an ID from that which allows rank to be chosen based on specified tolerance. For $m \times n$

input A, C is $m \times k$ and V is $n \times k$. Notice that when we apply the ID to the matrix transpose, we get $A^T \approx A(J_r(1:k),:)\tilde{V}^T$, where $A(J_r(1:k),:)$ represents a subset of k rows of the matrix, corresponding to the employed pivoting strategy. The use of the ID allows us to retain only a subset of all available channels via direct application to the transpose of the matrix containing the channel data, one per row. Of course, the V factor (or T_l and J_c or J_r) also needs to be stored. It should be noted that the data may need some pre-processing prior to being inserted into the matrix, such as in the case of substantial lag between channels. In that case, data vectors can be shifted and adjusted with respect to one another so that a portion of each channel vector would separately remain along with the ID factors.

2.2 High Correlation Modeling

Once the ID has been used to potentially reduce the number of retained channels, high correlation modeling can be employed to further decrease floating point storage requirements. The approach described here is loosely based on the method with the Fourier transform described in (Gutton et al. , 2007). In (Goffman-Vinopal & Moshe , 2002), the authors describe correlation compression for color image data. Here, we describe the use of the sorted and scaled Wavelet and log transforms, portion splitting, and low order polynomial modeling for a set of floating point signals. First, a matrix $M = [w_1; \ldots; w_m]$ is formed with $w_i = Thr(Wr_i)$. W can represent a CDF 9/7 or other Wavelet transform (Rao , 2002), with the optimal choice depending on the smoothness properties of the data. Thr() is an optional hard or soft thresholding operation (Kowalski , 2014), which retains only a portion of the largest by absolute value coefficients, resulting from the transformation W applied to a row vector (floating point channel data) r_i, such that $r_i \approx W^{-1}Thr(Wr_i)$ to a specified tolerance, corresponding to classic lossy signal compression in the signal processing sense. Clearly, if we are able to approximately reconstuct M from some compressed set, then we can obtain the original channel data by applying the inverse Wavelet transform to each row. Once M is constructed, three matrices are formed: $\bar{M}, M_{num}, M_{sgn}$. To do so, a descending order sort is applied to the absolute values of the transformed coefficients: $[v_i, I_i] = sort(abs(w_i), 'd')$ to get the corresponding sorted coefficient set to be stored in row i of $\bar{M}$ and the integer permutation index information I_i (for where the values occurred in the original unsorted Wavelet transformed vector) to be stored in M_{num}. The signs of $w_i(I_i)$ (the permuted coefficients) are stored in M_{sgn} as a bit array of 0s and 1s for each channel, corresponding to negative or positive transform coefficients in the permuted order. If we plot the entries of v_i, then for every channel i, they would have a similar, exponentially decaying shape as in Figure 2, and hence roughly linear, under a log transform. We can scale each vector v_i by the max entry, and save in addition, one scaling factor per row (or modeled portion), so that the log values of the transformed and sorted coefficient vectors for all channels start at 0. The basic idea to high correlation compression is to store v_i only for some i (forming the so called reference pillar set), and for the other i, use a low order model to model $\log(v_i)$ with respect to the log of one of the retained pillar vectors (of sorted absolute values of the Wavelet coefficients). Then, for the non-pillar channels, only the fitting coefficients are stored in place of the floating point vectors v_i. To reconstruct the original signal r_i for each channel, we employ the linear model reconstruction and the integer index and sign sets to form the approximate Wavelet coefficients of correct sign, followed by the application of the inverse transform. The pillar vectors can either be chosen as a small subset of all available data or as one or more members of a cluster, when waveform clustering is applied to cluster the data in groups. In that case, high correlation modeling can be performed separately over each cluster.

2.3 Compression and Decompression Algorithms

The general compression and decompression approaches are summarized in Algorithms 1 and 2, using a mix of lossy compression (as achieved via the ID, thresholding, and high correlation modeling) and later, optional lossless compression of the remaining parts. First, the ID is applied on the transpose of channel data $M = [r_1, \ldots, r_n]^T$, resulting in a subset of channels C (rows of M) and an extra factor V (typically small compared to C). The subsequent steps aim to compress the data in C via high correlation based modeling. For this, the three M matrices are formed from the data in C. Notice that M_{num} and M_{sgn} are integer and bitwise matrices with lower storage cost, while $\bar{M}$ contains the floating point sorted absolute values of the transform coefficients, only a portion of which will be retained. Some of the rows of $\bar{M}$ are retained in floating point format in E as pillar data. The rest of $\bar{M}$ is not stored. Choosing the pillar data can be accomplished by keeping a set portion of $\bar{M}$ as reference. Alternatively, rows can be added in blocks until a specified error tolerance is met or clustering can be used to make a set of pillars (e.g. taking the first member of each cluster). The resulting matrix E contains the pillar signal transformed and sorted absolute valued Wavelet coefficients against which other channels coefficients will be fitted. For the rest of rows j of $\bar{M}$, stored in matrix F, are the coefficients for a fitting model with respect to $\log(\bar{M}(i,:))$ in E (the optimal reference model to fit against $\log(\bar{M}(j,:))$), the scaling factors, and the index i to the reference in E. Once done, matrix F contains a small amount of floating point data per non-pillar channel and the combined approach can represent substantial savings over storing the floating point data for each channel. As a final step, lossless compression is employed over all retained data, including the integer and bitwise sets M_{num} and M_{sgn} and the V factor from the ID. Burrows-Wheeler based bzip2 can be used for any of the remaining data, and special methods e.g. (Ratanaworabhan et al. , 2006), can be employed for integer permutation sets. To improve performance, the data in $\bar{M}$

can be subdivided into several portions for each row and fitted separately with respect to the reference data in E (such that different portions of a row can be fitted against portions of possibly different pillar vectors). A simplified version of the transform correlation based encoding code is shown below, where a set of data is retained as reference pillars:

```
% transform and sort data per channel from C
% store abs vals, permutation info, signs, scaling factor array
wj = wavecdf97(xj,4);
[vals,inds] = sort(abs(wj),'d'); sfactors(j) = max(vals);
M_bar(j,:) = vals/sfactors(j);
M_num(j,:) = inds;
M_sgn(j,:) = sign(wj(inds));
% take a portion of M_bar as pillar data
E = M_bar(1:nsave,:);

% encode the rest with respect to saved signals
for j=(nsave+1):k
    sig_new = log(M_bar(j,:));
    errs = zeros(nsave,1);
    % choose best reference signal to use
    for ind=1:nsave
        sig_ref = log(E(ind,:));
        p = polyfit(sig_ref,sig_new,2);
        yfit = p(1) * sig_ref.^2 + p(2) * sig_ref + p(3);
        err_val = norm(yfit - sig_new)/norm(sig_new);
        if err_val < min_err
            p1u = p(1); p2u = p(2); p3u = p(3); pind = ind;
            min_err = err_val;
        end
    end
    % store model fit and reference to pillar signal in E
    F(j,:)=[p1u,p2u,p3u,sfactors(j),pind];
end
```

For decompression, assuming the use of a second order model, given model terms a_j and b_j and scaling factor s_j in F and the index i to the reference data in matrix E (corresponding to the reference channels for which the transformed and sorted coefficients are stored), we can compute $v_j = s_j \exp\left(a_j x_i^2 + b_j x_i + c_j\right)$ with $x_i = E(i,:)$ to model the sorted absolute values of the coefficients for channel j. Then, to construct approximate coefficients of the Wavelet transformed result for each channel (unsorted and with proper sign), we must make use of the permutation index and sign information stored in M_{num} and M_{sgn}. To do so, we initialize an empty vector for each channel, and insert v_j in entry specified by index $M_{\text{num}}(:, j)$ with sign specified by $M_{\text{sgn}}(:, j)$. Following the construction of the transformed vector with the correct sign entries, the inverse transform can be applied to yield the approximate signal for each channel. The sequence of steps is summarized by the pseudocode below, where the log model is fitted with respect to the best reference, then the coefficients are inserted in correct order with relevant sign, and subsequently inverse transformed. If the rows of $\bar{M}$ were split in several portions, then F would be composed of several sets of records for the different models of the respective row portions and the reconstructions for each portion would be performed separately. A simplified version of the code appears below, where we read the pillar and model data, evaluate the second order polynomial model, and use the permutation and sign information to construct the approximate transform coefficients for the channel.

```
% We must recover all but the pillar signals in E
for j=(nsave+1):k
        % read stored model and reference to pillar
        model_data1 = F(j,:);
        % get pillar data
        sig_ref_thr = log(E(model_data1(5),1:round(N)));
        % evaluate polynomial model and rescale
        sig_recov = sfactors(j)*exp(model_data1(1)*sig_ref_thr.^2 + model_data1
            (2)*sig_ref_thr + model_data1(3));
```

```
        % construct transform coefficients
        % based on permutation and sign info and invert
        sig_recov_wt = zeros(1,N);
        for s=1:N
                if M_sgn(j,s) > 0
                        sig_recov_wt(M_num(j,s)) = sig_recov(s);
                else
                        sig_recov_wt(M_num(j,s)) = -sig_recov(s);
                end
        end
        sig_recov_iwt = wavecdf97(sig_recov_wt,-4);
end
```

The use of the ID above improves compression ratio by reducing the number of retained channels. Similarly, splitting the transformed and sorted absolute value coefficients into several portions before modeling can significantly improve performance. This can be done in four parts as below or adaptively based on the behavior of the sorted absolute values.

```
sig_ref1 = log(E(ind,1:round(N/4))); sig_ref2 = log(E(ind,(round(N/4)+1):round(N
    /2))); ... ; sig_ref4 = log(E(ind,(round(3*N/4)+1):end));
```

Notice that the transformed and sorted absolute values of the Wavelet transform of channel data when plotted (see e.g. the plot in Figure 2) generally exhibit the greatest mean curvature (change) in the first half or quarter of the interval and it is the largest coefficients in that region that are most critical to model accurately for good reconstructions. For this reason, it may be beneficial to split the data non-uniformly, utilizing more portions for the first half of the interval. The advantage of this splitting is that it computes different model coefficients per portion and allows different portions to be fitted with respect to different pillar data vectors. Another method which we discuss in the examples is to precluster the original channel data into several parts (clusters) and take one or more members of each cluster to be the pillar data. A sample clustering sequence in R is shown below using a dissimilarity matrix construction between floating point channel data pairs based on dynamic time warping metrics (Oates et al. , 1999; Mori et al. , 2016). High correlation modeling compression can then be performed over each cluster.

```
for (i in 1:nchannels){
    for (j in 1:nchannels){
        si =sensor_mat[i,]; sj =sensor_mat[j,];
        dist1 = ACFDistance(si,sj); dist2 = ARLPCCepsDistance(si,sj);
        diss_mat[i,j] = (dist1+dist2)/2
    }
}
hc <- hclust(d = as.dist(diss_mat), method="complete");
rect.hclust(hc,nclusters); for(cn in 1:nclusters){sensor_nums = hc2[[cn]]}
```

In some cases, the behavior of signals may change significantly over time. Thus, it is plausible to first subdivide the signals over all channels into several portions and then cluster and compress those portions separately. Similarly, quite (low amplitude areas) of the channels may be removed and separately processed.

Algorithm 1. Compression sequence

Data: Floating point data from multiple channels. Tolerance and pillar block parameters ($\epsilon_{1,2,3}, L$), Wavelet transform, and thresholding function.
Result: Compressed representation of data for all channels.
Insert floating point data into matrix A, one channel per row.
Perform ID decomposition on the transpose of the matrix, $A^T \approx A(J_r(1:k),:)V$ with rank chosen per ϵ_1 tolerance.
Set $C = A(J_r(1:k),:)$ to be the subset of retained channels.
Form matrices $\bar{M}, M_{num}, M_{sgn}$ from C.
for $j = 1, \ldots, k$ **do**
  Compute $w_j = transform(C(j,:))$
  $[v_j, I_j] = sort(abs(w_j), 'd')$
  Store permutation inds I_j from sort and signs of $w_j(I_j)$ in $M_{\text{num}}(j,:)$ and $M_{\text{sgn}}(j,:)$.
  Set $\bar{M}(j,:) = Thr(v_j)$ per ϵ_2.
end
Set $M_E = 1e6, i = 0$. Initialize E to hold subset (the pillars) of $\bar{M}$ and F to hold linear fitting information.
while $M_E > \epsilon_3$ **do**
  Add $C(i+1, \ldots, i+L,:)$ to E.
  for $j = i + L + 1, \ldots, k$ **do**
    Compute low order polynomial fit model between $\log(\bar{M}(j,:))$ and each of the saved channels $\log(E(i,:))$.
    Record scaling factor s_j, modeling coefficients a, b and index to pillar model corresponding to smallest error against $E(i,:)$ in $F(j,:) = [a, b, s_j, i]$.
    Record reconstruction error as e_j.
  end
  Let $M_E = \max(e_j), i = i + L$.
end
Lossless compress saved floating point data E, fitting coefficient set F, as well as the integer and bit sign matrices M_{num} and M_{sgn} and ID matrix V.

Algorithm 2. Decompression sequence

Data: Compressed data set from Algorithm 1.
Result: Approximate floating point data for each channel.
Run lossless decompression routines on E, F, M_{num}, M_{sgn}, and V.
Utilize the resulting matrices to reconstruct C. To do so, the sorted magnitude coefficients are reconstructed by means of the evaluated model with x from the specified (best fit row) in E. The vector is then scaled and exponentiated, and its entries are inserted into appropriate locations using M_{num} index set, with appropriate sign assigned from the bitwise M_{sgn} information. The inverse transform is then applied over both the reconstructed transform data and pillar data in E to yield the approximate select channel data C.
Multiply C by V to obtain approximation to A^T.
Extract floating point channel information from the columns for A^T.

3. Numerical Results

We present results from two applications: medical ECG signals and synthetic acoustic data receiver. As discussed in the introduction, both data sets contain data for multiple channels. For the ECG data set, data from 25 patients was used, for a total of 1020 records, each of length 115200 samples. In double precision format, the storage size for the 1020×115200 matrix is 940 MB. With a rank 400 ID (resulting in a relative error of 3.7%), the retained channels take up 368 MB, along with a small dense matrix V of size 1020×400. Following the ID computation, correlation based compression on transformed and sorted data is performed. That is, the data is first transformed and sorted. The indices and signs are also recorded. Lossy compression (thresholding) is performed to reduce the size of coefficients retained. This is done, by looping over every channel and measuring the error that results from retaining only a portion of the largest coefficients. A suitable portion to retain is then determined over the maximal error over all channels. In our case, 3/4 of the largest magnitude coefficients were retained in every channel. Next, we save 40 of 400 transformed and thresholded channels as reference pillars and encode the remaining values in terms of the optimal paired saved channel. In order to

improve performance, we have broken up signals in subsets of length 7000. The full matrix for one portion is then 57 MB and the ID factors (C and V) are 22.4 and 3.2 MB, with 1.2% error in the approximation to A^T. We then divided the sorted absolute values of the Wavelet coefficients (the decaying exponential curves) into four parts indexed by the sets $(1 \to \text{round}(N/4)), \ldots, (\text{round}(3N/4) + 1 \to N)$, which are separately fitted via a third degree polynomial model based on the optimal pairing with one of the transformed and sorted reference signals.

Following this, we have the following saved data:

- Saved reference channels 40×7000 in double, 2.24 MB.
- Saved sort indices 400×7000 in uint32, 11.2 MB.
- Saved entry signs 400×7000 in int8, 2.8 MB.
- Small model coefficient and reference signal index array for the 4 transformed signal parts, each of size 400×4.

The total size is thus around 16.2 MB, down from 25.6 MB for the ID factors and 57 MB for the original data. Combining these pieces to cover the entire 115200 sample length requires around 267 MB total storage. Lossless compression can be used to further compress retained data. Error results for the transformed coefficient set reconstructions per channel are shown in Figure 3 along with a sample fit of a waveform based on the model derived from one of the 40 retained pillars. Notice that we can choose to retain less pillars (floating point data) via clustering, as we show in the next example.

In the following test, we show results for a synthetic acoustic application, whereby an array of 29 microphones is recording the signal from a particular emitter and interference sources. The main setup is shown in Figure 1 (left). Figure 4 shows a typical signal obtained by one of the microphones, along with its subdivision into 6 parts. Since the measured signal at a given microphone potentially consists of many samples (as in the example shown), it is reasonable to subdivide the signal into several portions before employing compression. If the signal has both active (high magnitude) and relatively flat (low magnitude) portions, depending on the emitter and interference source profiles, then a gradient map of the indices corresponding to large magnitude entries of the floating point vector (e.g. above the third quartile after filtering the small entries away), can be used to estimate cut locations along the flat areas of the signal, as shown. Once subdivision is performed, the compression sequence can be performed in parallel over different portions on multi-processor or suitable multi-core systems, if necessary. In Figure 5, we show one particular portion (#3), as recorded at 4 different sensors. The pillar selection approach then proceeds by clustering, using all microphone signals from this portion, subdivided in 2000 point portions. In order to accomplish clustering, a dissimilarity matrix is first constructed, which shows the relative distance between pairs of recorded microphone signals. The distance approach is based on dynamic time warping (DTW) (Oates et al. , 1999) and auto regressive integrated moving average (ARIMA) process derived metrics (the average of ACF and ARLPCCeps distances from the TSdist package in R was used (Mori et al. , 2016)). The number of clusters was set to 6 and the resulting dendrogram shows the resulting microphone distribution with respect to different clusters. For each cluster, only the transformed coefficients of the first member (the pillar) were saved and the rest of the signals were encoded with respect to the pillar using the described technique. The resulting final error distribution is shown in the same figure along with per cluster statistics consisting of the original floating point size, the similarity compressed size, and the final size following Burrows-Wheeler based and entropy lossless compression. Notice that the second problem shown here is more challenging for two reasons: the more oscillatory (higher frequency) nature of the signals and the fact that encoding is done with respect to only one pillar, unlike the ECG example, where 40 pillar signals were saved and the optimal reference was used in each case. We can instead choose to save more than one pillar data per cluster. Figure 6 shows two examples with both the reconstructed signed and properly ordered Wavelet coefficients and the reconstructed signals (obtained from the inverse transform of the reconstructed Wavelet coefficients), for a higher and lower error case. As expected, when the Wavelet transform output is more oscillatory, the reconstruction mechanism gives greater error. Still the final result captures the main features of the signals, with some deviations in the magnitudes. For the signal on the right of the figure, both the Wavelet coefficient sequence and subsequent signal reconstruction are close to the originals. In this example, we have also not applied the ID since the full rank of the data matrix is relatively small, but additional gains are possible with respect to the cluster sizes reported in Figure 5.

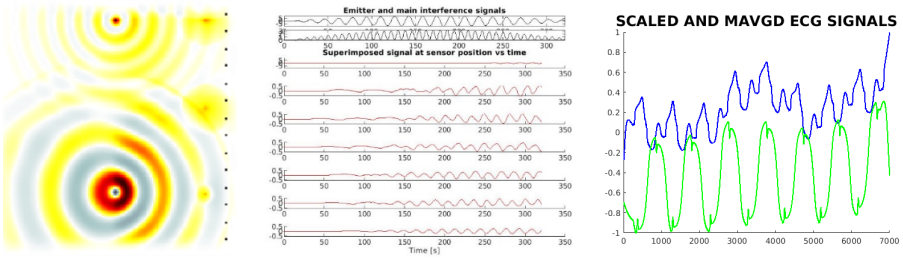

Figure 1. Example acoustic geometry and measured array signals and scaled and moving averaged ECG signals

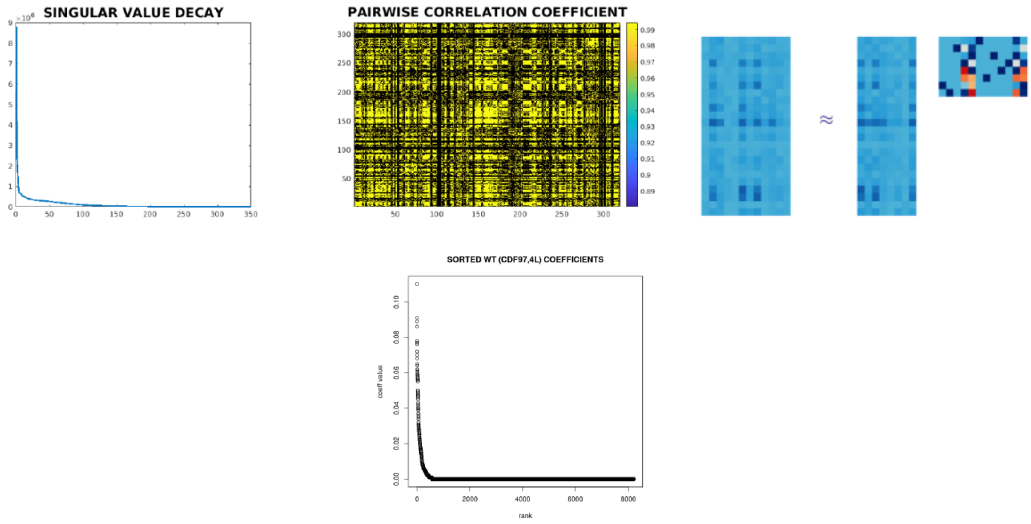

Figure 2. Singular value and pairwise correlation between transformed and sorted coefficients and illustration of the ID. Sorted magnitudes of transformed signal coefficients

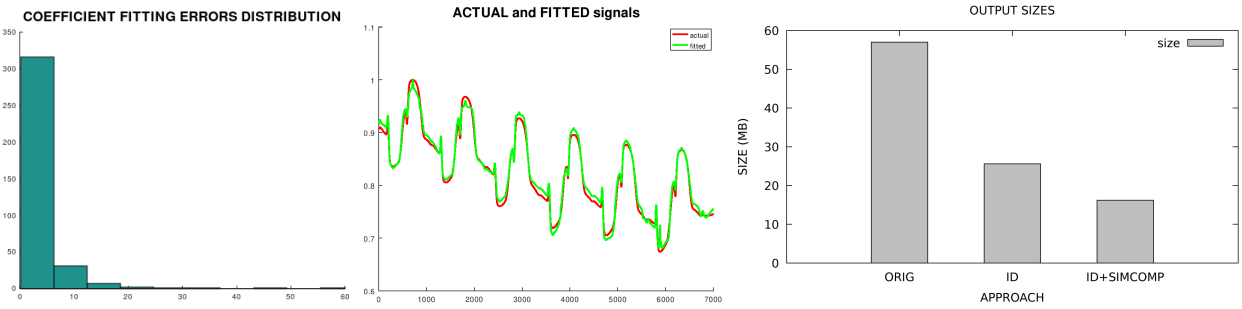

Figure 3. Reconstruction errors for ECG set and sample reconstructions

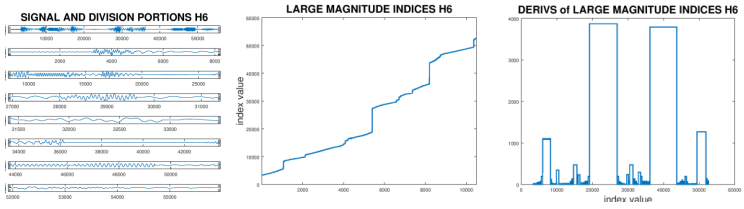

Figure 4. Sample acoustic signal subdivided into portions

4. Discussion

In conclusion, we have presented an approach useful for the compression of a set of similar time domain signals, such as those obtained from multiple test subjects (e.g. medical patients) or recorded by multi-channel processing systems.

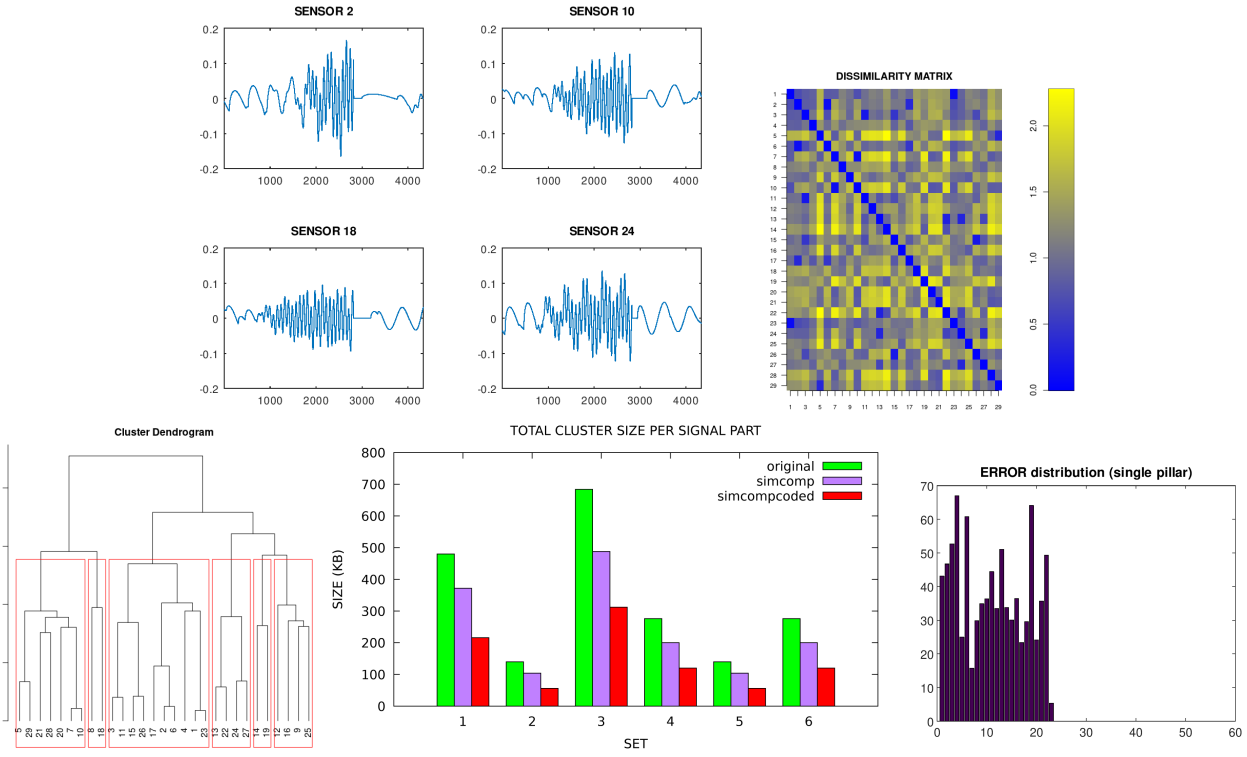

Figure 5. Acoustic signals for one portion, corresponding dissimilarity matrix over all sensors, resulting clustering dendrogram, and error distributions and per cluster stats: original, similarity compressed, entropy encoded sizes

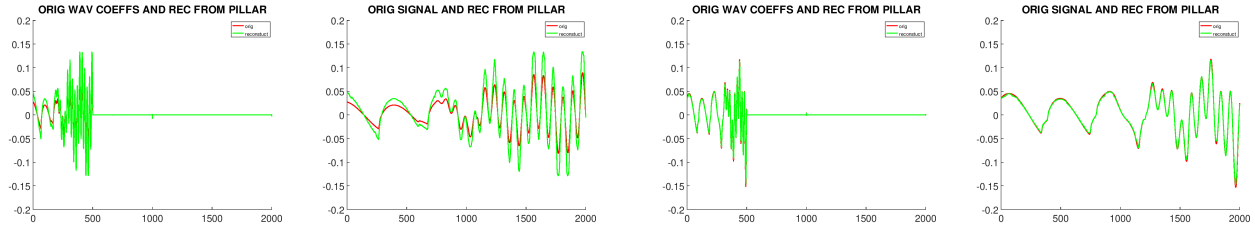

Figure 6. Sample transform coefficient set reconstructions (signed and unsorted) and corresponding signal reconstructions using a low order model with respect to one pillar in a cluster

The basic approach is to exploit both the linear dependence and correlation of data after transformation and sorting. The approximate linear dependence between some of the channel pairs can be exploited by means of the interpolative decomposition, which returns a subset of the data matrix rows when applied to its transpose. The rank can be modified adaptively based on the desired error tolerance level. Next, the remaining data can be broken up into groups via clustering and a set of pillar signals can be retained. The absolute values of the transformed and sorted coefficients of the different signals are then compared to those of the retained pillar data and a low order polynomial model is saved in place of floating point data, along with sign and index permutation sets in more efficient data types. The original data channels can then be approximately recovered by means of the model information, the retained sign and integer index sets, and the inverse transform. Lossless compression can then be applied to the resulting data sets (the ID factor, transformed pillar coefficients, polynomial model information, and the bitwise sign and integer index sets). The approach is simple and efficient to employ, adaptive with respect to specified tolerances, and can yield substantial compression gains.

Acknowledgements

We kindly acknowledge the various helpful comments and improvement suggestions from Dr. Asmae Benali and the journal editor.

References

Goffman-Vinopal, L., & Moshe, P. (2002). Color image compression using inter-color correlation. *In Proceedings. International Conference on Image Processing, 2,* II-II. IEEE, 2002. https://doi.org/10.1109/ICIP.2002.1039960

Guitton, A., Antonios, S., & Niki, T. (2007). Utilizing correlations to compress time-series in traffic monitoring sensor networks. In Wireless Communications and Networking Conference, 2007. WCC 207. IEEE, pp. 2479-2483. IEEE, 2007. https://doi.org/10.1109/WCNC.2007.462

Kowalski, M. (2014). Thresholding rules and iterative shrinkage/thresholding algorithm: A convergence study. *In 2014 IEEE International Conference on Image Processing (ICIP),* pp. 4151-4155. IEEE, 2014. https://doi.org/10.1109/ICIP.2014.7025843

Martinsson, P., & Sergey, V. (2016). A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. *SIAM Journal on Scientific Computing, 385,* S485-S507. https://doi.org/10.1137/15M1026080

Moody, G. B., Roger, G. M., & Ary, L. (2001). Goldberger. PhysioNet: a web-based resource for the study of physiologic signals. *IEEE Engineering in Medicine and Biology Magazine, 20*(3), 70-75. https://doi.org/10.1109/51.932728

Mori, U., Alexander, M., & Jose, A. L. (2016). Distance measures for time series in R: The TSdist package. *R journal, 8*(2), 451-459. https://doi.org/10.32614/RJ-2016-058

Oates, T., Laura, F., & Paul, R. C. (1999). Clustering time series with hidden markov models and dynamic time warping. In *Proceedings of the IJCAI-99 workshop on neural, symbolic and reinforcement learning methods for sequence learning,* pp. 17-21. Sweden Stockholm, 1999.

Rao, R. (2002). *Wavelet transforms.* Encyclopedia of Imaging Science and Technology. https://doi.org/10.1002/0471443395.img112

Ratanaworabhan, P., Jian, K., & Martin, B. (2006). *Fast lossless compression of scientific floating-point data.* InData Compression Conference, 2006. DCC 2006. Proceedings, pp. 133-142. IEEE, 2006. https://doi.org/10.3917/deba.142.0133

Treeby, B. E., & Benjamin, T. C. (2010). k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields. *Journal of biomedical optics, 15*(2), 021314. https://doi.org/10.1117/1.3360308

Voronin, S., & Per-Gunnar, M. (2017). Efficient algorithms for cur and interpolative matrix decompositions. *Advances in Computational Mathematics, 43*(3), 495-516.

Copyrights