Published in final edited form as:

Int Conf Pervasive Technol Relat Assist Environ. 2010 June ; 2010: . doi:10.1145/1839294.1839359.

Linear Frequency Estimation Technique for Reducing Frequency Based Signals

Jonathan Woodbridge,
UCLA, Computer Science, Los Angeles, California USA, jwoodbri@cs.ucla.edu

Alex Bui, and
UCLA, Medical Imaging Informatics, Los Angeles, California USA, buia@mii.ucla.edu

Majid Sarrafzadeh
UCLA, Computer Science, Los Angeles, California USA, majid@cs.ucla.edu

Abstract

This paper presents a linear frequency estimation (LFE) technique for data reduction of frequency-based signals. LFE converts a signal to the frequency domain by utilizing the Fourier transform and estimates both the real and imaginary parts with a series of vectors much smaller than the original signal size. The estimation is accomplished by selecting optimal points from the frequency domain and interpolating data between these points with a first order approximation. The difficulty of such a problem lies in determining which points are most significant. LFE is unique in the fact that it is generic to a wide variety of frequency-based signals such as electromyography (EMG), voice, and electrocardiography (ECG). The only requirement is that spectral coefficients are spatially correlated. This paper presents the algorithm and results from both EMG and voice data. We complete the paper with a description of how this method can be applied to pattern types of recognition, signal indexing, and compression.

Keywords

Pattern Recognition; Signal Indexing; Compression

1. INTRODUCTION

This paper presents a linear frequency estimation (LFE) technique for data reduction of frequency-based signals. LFE is applicable to a wide variety of signals including voice, electromyography (EMG), and electrocardiography (ECG) data. This technique relies on one fundamental property: spatial correlation of frequency components. LFE relies on the ability of a small subset of frequency components to accurately estimate the entire frequency band of a signal. This method allows an extremely aggressive removal of frequency components resulting in a representation based on a number of vectors far less than the original input size. This paper presents the basic algorithm and initial results. As we discuss later, this algorithm is applicable to many fields of computer science including compression and signal indexing.

LFE not only reduces the amount of data, but allows for the reconstruction of a signal with a configurable amount of data loss. This property yields a technique that is not only applicable to pattern recognition and signal indexing, but to compression as well. We later show results that demonstrate this technique's applicability in the all three of the aforementioned fields.

There are three main contributions of this paper. First, we show a data reduction algorithm for reducing frequency based signals based on a minimization of error per spectral coefficient. A full description of the minimization function, algorithm (and its inverse), and complexity are provided. Second, we show results of this algorithm as applied to EMG and voice data. Finally, we demonstrate the applicability of this technique to compression, pattern recognition, and indexing of frequency based signals.

2. ALGORITHM

LFE is based on Fourier transforms. The reduction is accomplished by converting the signal to the frequency domain and extracting the most significant points to allow an estimation that minimizes the error per spectral coefficient. These most significant components construct a vector representation of the original signal.

The data is first separated into bins of size m. We then convert each bin to the frequency domain using a Fast Fourier Transform (FFT) similar to that described in [2]. This transformation results in a complex array of numbers that consist of both real and imaginary parts. These two parts are separated and stored in two arrays of size $\frac{m}{2}$. Note that the two arrays only require a size of $\frac{m}{2}$ as opposed to m. This is true since we generally deal with real valued data. An FFT applied to a real data set results in a symmetrical output where the negative frequencies are the complex conjugate of the positive. Hence, half the frequency output is redundant. Further details are provided in [4].

Next, we determine the most significant points in both the imaginary and real domains as computed by the FFT for each bin independently. The most significant points are defined as the minimum number of points to estimate the frequency domain within an error measure of Δ (A full description of variables are defined in table 1). Δ is derived by finding the minimum error per frequency component between an estimation function and the original function. We define $m-1$ estimation functions; one for each size of α_i from $2 \Rightarrow m$. An optimal estimation function is defined by a series of lines connecting to each consecutive point in α_i'. Any point in M, but not in α_i' is estimated by these lines on reconstruction. The optimal estimation function is defined by:

$$F_{\text{opt}} = \underset{\forall \alpha_i \in M}{\operatorname{argmin}} \sum_{j=1}^{i-1} \sum_{k=\alpha_i x(j)}^{\alpha_i x(j+1)} |(s_j \times k + b_j) - f(k)| \quad (1)$$

$$s_j = \frac{\alpha_y(j+1) - \alpha_y(j)}{\alpha_x(j+1) - \alpha_x(j)} \quad (2)$$

$$b_j = \alpha_y(j) - s_j \times \alpha_x(j) \quad (3)$$

This calculation is done for all i from $2 \Rightarrow m$ for both the real and imaginary parts for all bins. However, the above equation results in a non-polynomial runtime bounded by $O(\sum_{i=2}^{m} \binom{m}{i})$. We define a dynamic programming algorithm to achieve a polynomial runtime.

2.1. Dynamic Programming Algorithm

We introduce a dynamic programming algorithm inspired by [6] to compute these points. This dynamic programming algorithm finds the minimum number of points to estimate a signal with a local error per spectral coefficient below a predefined threshold δ. These minimum points are considered the most significant points and construct a series of vectors that best estimate the original real and imaginary domains.

Our algorithm iteratively creates an $\frac{m}{2} \times \frac{m}{2}$ table T_i and T_r (for both real and imaginary components) where $T(i, j)$ denotes the minimum error per spectral coefficient to travel from point i to point m (endpoint) that include j edges. This is done for each bin computed by the FFT. As the number of j increases, the spectral error will decrease until $j = m - 1$ (or when all edges from $i \Rightarrow end$ are included in the set). The algorithm builds this table from left to right, iterating through each column from top to bottom. The algorithm is shown in figure 1. This algorithm computes both T and S where S stores the next optimum point for each path. S is required to reconstruct the optimum path described by T.

BuildEdges constructs an $m \times m$ array that contains the absolute difference between the values defined by the edge $\overline{ij}$ and the actual values from i to j. m is the size of the input. Details of this algorithm are shown in 2. The function $Line$ creates a line between the two input points and returns a vector of values defined by this line.

We create a variable δ that determines the maximal spectral error allowed for each estimation computed in ComputeOptimalPoints. Once T is returned, we enumerate the first row of T calculating its error percentage. We start from 1 to m stopping at the first point that satisfied our constraint δ. We are guaranteed to satisfy any value of δ since in the worse case we could choose all points as most significant meaning the error per spectral coefficient would equal 0.

2.2 Reconstruction

The reconstruction algorithm is much simpler than compression. Reconstruction is accomplished by inserting previously omitted values by their linear estimations computing during data reduction. Once a bin is constructed, we convert the signal to the time domain through an inverse Fast Fourier Transform (iFFT). Each bin is appended to construct the full time domain signal.

2.3. Runtime Analysis of Reduction

The reduction algorithm starts by splitting the input signal into bins of size m (Fourier window size) and computing the Fourier transform for each bin. When using the FFT as defined by [2] we achieve a runtime bounded by $O(\frac{n}{m} m \log m) = O(n \log m)$. Next, we loop through each bin. During this loop we extract the real and imaginary parts and find their corresponding most significant points. Extracting the real and imaginary parts is trivially bounded by $O(n)$. In fact, this step can be entirely avoided by implementing the FFT algorithm to return both real and imaginary parts separately.

Next we calculate the most significant points by minimizing the error per spectral coefficient. *ComputeOptimalPoints* starts by calling *BuildEdges*. This function builds an $m \times m$ matrix (as described earlier). This function consists of an outer and inner loop both bounded by m yielding an $O(m^2)$ runtime. Next, *ComputeOptimalPoints* has 3 nested loops each bounded by m and include all constant time operations. This gives a total runtime of *ComputeOptimalPoints* of $O(m^2 + m^3) = O(m^3)$. This yields a total runtime for compression of $O(\frac{n}{m}(m^3) + n \log m) = O(n(m^2))$.

2.4. Runtime Analysis of Reconstruction

Analysis of the reconstruction algorithm starts with the insertion of omitted points. The algorithm loops through each bin constructed during compression and inserts each missing point. The number of bins is derived by the total number of points divided by the Fourier window size or $|bins| = n/m$ where n is the size of the input and m is the size of the Fourier window. For each bin we recreate the imaginary and real curve and calculate the iFFT. The iFFT runs in $m \log m$ as shown in [2]. Each bin has at most m points with a total of n/m bins. For each bin we insert omitted points than perform an iFFT for a bounded runtime of $O(m + m \log m) = O(m \log m)$. This yields a total runtime of $O(n/m(m \log m)) = O(n \log m)$

3. RESULTS

We used two distinctly different sets of data to assess LFE. The first set of data included EMG data acquired from rats surgically implanted with EMG electrodes. The data included 20 different readings from 10 different rats and a heterogeneous mix of biceps and triceps from both the hind limbs and forelimbs. Samples ranged from 45 seconds to 3 minutes and were sampled at 2kHz. The second set of data were voice samples taken from the Open Speech Repository [8]. Speech included 20 samples of both male and female speaking in English, and Mandarin. Speech was sampled at 8kHz and samples ranged from 20 to 45 seconds. To note, speech data was chosen to show LFE's flexibility and not as a replacement

to current speech specific algorithms. An example reconstructed signal versus its original signal is shown in figure 3.

Accuracy was calculated by comparing the original signal's frequency power to a reconstructed signal's frequency power. The comparison used a sliding window Discrete Fourier Transform (SWDFT). For each window, the Fourier transform is calculated for both the original and reconstructed signal. Next, the euclidean distance of the Fourier transform is calculated for both signals and compared. The error is averaged over all windows to find the total error and is returned as a percentage difference between the reconstructed and original signal. Each bin is normalized such that bin's containing little frequency power contribute less than bins containing higher frequency power.

Results for EMG and voice are shown in 4. Larger Fourier window sizes generally get better results with diminishing improvements. A window size of 1024 worked well for both EMG and voice receiving a compression of 70% and 92% respectively when using a 45% local accuracy (δ). Both had a 25% error rate on reconstruction.

4. IMPLEMENTATIONS

LFE is applicable to many fields including compression, pattern recognition, and signal indexing. The applicability to compression is quite obvious from the results in section 3. Compression rates are near 90% with high accuracy and include no optimizations. Further compression can be obtained by taking advantage of properties of the signal's receiver. For example, algorithms could remove/combine redundant frequency components by utilizing the human ears natural pitch resolution and range. These are similar mechanisms as done in MP3 and AAC [1]. However, the power of this algorithm is it ability to reduce a fairly general set of frequency based signals. Utilizing signals' observers' properties will reduce the applicability to one domain.

A less obvious application of LFE is in pattern recognition. One simplified method of pattern recognition is to treat each frequency bin as a character and a concatenation of bins as words. Algorithms such as [5] can then be used to find similarities between patterns. The difficult part is determining the definition of a character. Training using LFE can be used to define different classes of characters. The idea is quite simple. A series of training bins of the same type (or character are analysed to find values for each point in the frequency domain (1 to m). Next, we find the standard deviation at each point (1 to m) and define the upper bound of the signal to be one standard deviation from the mean and the lower bound to be one standard deviation below the mean. We treat the lower bound and upper bound as new signals and calculate their LFE representation. Next, we can take an LFE reduced signal and check bins for a specific character. The number of comparisons (on the average) would be reduced significantly since the number of comparisons is reduced from m (as in standard euclidean distance) to a number $m' << m$.

The method above trivially leads to a signal indexing method. By reducing a frequency based signal to textual-like representation, we can use several established methods of

indexing and categorization such as those in [7] [9] improve signal searching capabilities. Similar methods have been applied by [3] on non frequency-based signals.

5. CONCLUSION

This paper presented a generic algorithm for reducing frequency based signals. This algorithm is specifically applicable to index and search signals, pattern recognition and data compression. We presented examples of how this algorithm can be applied to these three areas as well as presented reduction results from both EMG and voice; two largely different signals.

Acknowledgments

This publication was partially supported by Grant Number T15 LM07356 from the NIH/National Library of Medicine Medical Informatics Training Program

REFERENCES

1. Brandenburg, K. MP3 and AAC explained; AES 17th International Conference on High-Quality Audio Coding; 1999.
2. Cooley J, Tukey J. An algorithm for the machine calculation of complex Fourier series. Mathematics of computation. 1965; 19(90):297–301.
3. Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems. 2001; 3(3):263–286.
4. Proakis, JG.; Manolakis, DG. Digital signal processing: principles, algorithms, and applications. Indianapolis, IN, USA: Macmillan Publishing Co., Inc.; 1992.
5. Ristad ES, Yianilos PN. Learning String-Edit Distance. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1998; 20(5)
6. Schuster G, Katsaggelos A. An optimal polygonal boundary encoding scheme in the rate distortion sense. IEEE Transactions on Image Processing. 1998; 7(1):13–26. [PubMed: 18267376]
7. Sebastiani F. Machine learning in automated text categorization. ACM computing surveys (CSUR). 2002; 34(1):1–47.
8. The Open Speech Repository, VoIPTroubleShooter.com. 2010 www.voiptroubleshooter.com/openspeech/index.html.
9. Yang, Y.; Liu, X. A re-examination of text categorization methods; Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval; 1999. p. 42-49.

```
function ComputeOptimalPoints(inputBin)
  edgeWeights = BuildEdges(inputBin);

  // Initialize first column
  for j=1:size(T,2),
    T(j,1) = edgeWeights(j,size(T,2));
    S(j,1) = size(T,2);
  end

  // Build each column from left to right
  for i=2:size(T,1),
    for j=1:size(T,2)-i,

      T(j,i) = Inf;
      S(j,i) = 0;
      for k=j+1:size(T,2)-i+1,
        temp = edgeWeights(j,k) + T(k,i-1);

        if temp < T(j,i),
          T(j,i) = temp;
          S(j,i) = k;
        end
      end
    end
  end

  return T, S;
end
```

Figure 1.
A dynamic programming algorithm that calculates the optimal path from each point to the end point with path length from 1 to $m - 1$

```
function BuildEdges(inputBin)
  edgeArray = array(length(inputBin), length(inputBin));

  for i=1:size(edgeArray,1),
    for j=i+1:size(edgeArray,2),
      edgeArray(i,j) = sum(abs(inputBin(i:j) -
                               Line(i, input(i),
                                    j, input(j))));
    end
  end
end
```

Figure 2.

Computes the total spectral error by calculating the difference between the integral of every edge $\overline{ij}$ and $\sum_{k=i}^{j} \text{inputBin}(k)$

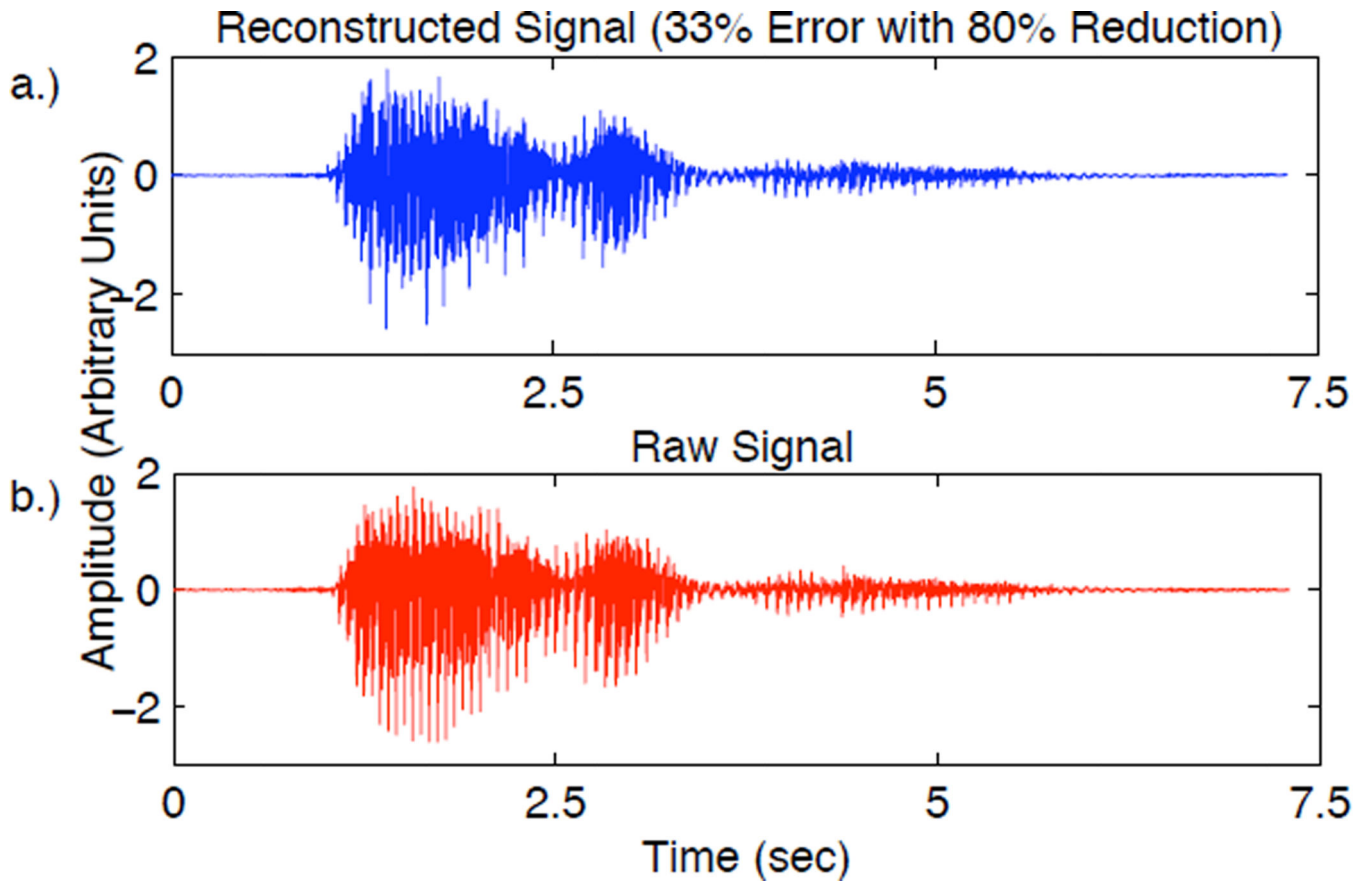

Figure 3.
A reconstructed voice signal.

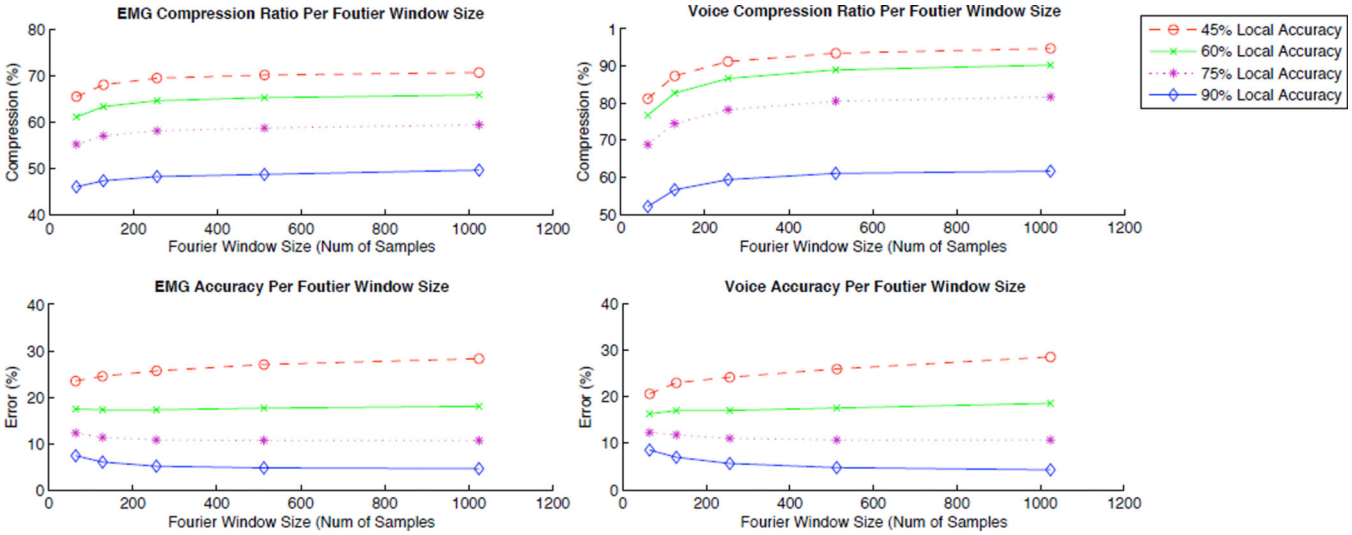

Figure 4.

Shows reduction percentage vs Fourier window size and accuracy vs. Fourier Window size for both EMG and voice. As expected, an inverse relationship is observed between accuracy and compression rate. Reduction and accuracy are computed for local accuracy of 45%, 60%, 75%, and 90%.

Table 1

Variable definitions for the LFE algorithm

Variable	Definition
M	Set of all points in the current bin (|M|=m)
f	Original signal of the current bin
α_i	An ordered set of points of size i consisting of both a domain and range component
α_{ix}	An ordered set of domain points of size *i* (corresponding to α_i)
α_{iy}	An ordered set of range points of size *i* (corresponding to α_i)
α_i'	The optimal ordered set of size *i*
Δ	Local error per spectral coefficient