AN IMPROVED FEATURE EXTRACTION TECHNIQUE FOR HIGH VOLUME TIME SERIES DATA

Jonathan S. Anstey and Dennis K. Peters
Electrical and Computer Engineering
Memorial University of Newfoundland
St. John's, NL Canada
Email: {anstey, dpeters}@engr.mun.ca

Chris Dawson
INSTRUMAR Limited
St. John's, NL Canada

ABSTRACT

The field of time series data mining has seen an explosion of interest in recent years. This interest has flowed over into many applications areas, including fiber manufacturing systems. The volume of time series data generated by a fiber monitoring system can be huge. This limits the applicability of data mining algorithms to this problem domain. A widely used solution is to reduce the data size through feature extraction. Four of the mostly commonly used feature extraction techniques are Fourier transforms, Wavelets, Piecewise Aggregate Approximation, and Piecewise Linear Approximation (PLA). In this paper, we first empirically demonstrate that PLA techniques produce the highest quality features for this problem domain. We then introduce a novel PLA algorithm that is shown to produce higher quality features than any other currently available techniques.

KEY WORDS
Signal processing, segmentation, feature extraction, manufacturing, and time series.

1 Introduction

The field of time series data mining has seen an explosion of interest in recent years. Researchers in this field are typically faced with two related problems: many data mining algorithms have a high time complexity and time series databases are often very large. Together, these problems make practical use of data mining technologies such as novelty detection in time critical systems difficult. The solution is to either reduce algorithm complexity or reduce data size. Much work has been devoted to both problems in the data mining community. We will focus on the aspect of reducing data size.

Reducing data size is often referred to as the process of feature extraction. Simply put, feature extraction is the process of identifying important data features while removing unimportant ones. Features could be actual data points, statistics on several data points, lines, clusters, or even coefficients of functions. The goal is to end up with fewer features than original data points so that data mining algorithms can run in a reduced amount of time.

Many feature extraction techniques have been proposed for time series data mining, including Fourier/Wavelet transforms [1, 2], Singular Value Decomposition (SVD) [3], Adaptive Piecewise Constant Approximation (APCA) [4], Symbolic Mappings [5, 6], Piecewise Aggregate Approximation (PAA) [7, 8], and Piecewise Linear Approximation (PLA) [9, 10, 11, 12]. In this paper, we will evaluate the four most popular techniques on datasets from an industrial synthetic polymer fiber monitoring application [13]. Synthetic polymer fiber (hereafter referred to as fiber) in this application is used to make carpet and other industrial fibers. Our motivation for evaluating these techniques here is to use the best technique for data preprocessing in a fiber property novelty detection application [14]. Time series novelty detection is an active area of data mining research and to our knowledge has not been applied to a fiber monitoring application.

The main result of these experiments is that the PLA approach produces the highest quality features at all compression levels tested. This motivated us to create a new PLA algorithm that produces higher quality features than all other currently available techniques on fiber property datasets.

The remainder of the paper is organized as follows. In Section 2, we describe the major feature extraction techniques and the optimizations used. In Section 3 we empirically evaluate the techniques and show that PLA methods produce the highest quality features. In Section 4 we detail our new PLA algorithm and validate its results. We conclude in Section 5 with a summary and directions for future work.

2 Related Work

Many feature extraction algorithms have been presented in the literature. The purpose of these algorithms is to reduce data size while keeping only the important details intact (i.e., the data features). In this section, we discuss the four most popular feature extraction techniques that have been used for time series data mining. Figure 1 shows these techniques in action. We refer the interested reader to works by Keogh et al. [15, 5] for a more extensive survey of feature extraction techniques.

For the purposes of our experimentation, all tech-

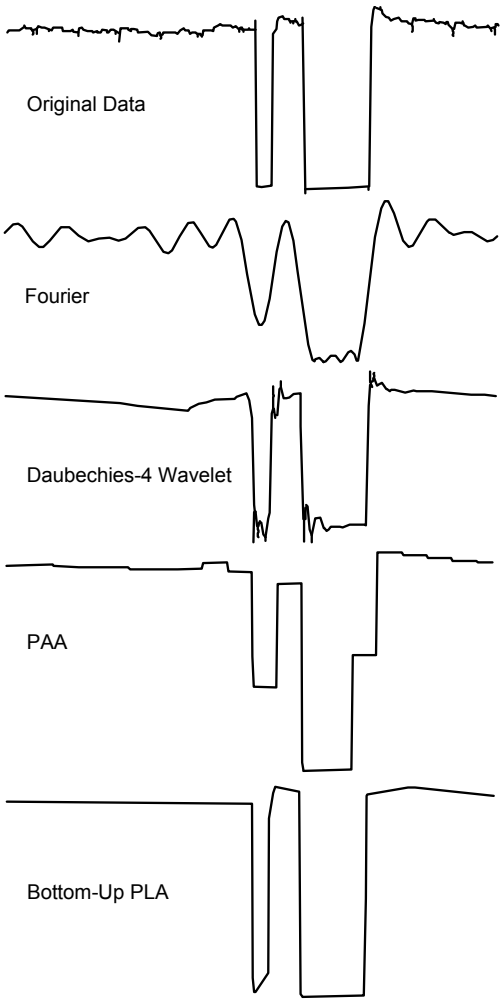

Figure 1. Major feature extraction techniques reducing a 512 point sample to 20 points.

niques will need to reduce a time series of length n to N data points. Data points include anything required to store the compressed representation; this includes actual values, indices, or other bookkeeping data.

Table 1 contains the notation used throughout this paper.

2.1 Discrete Fourier Transform

The first technique presented to the time series data mining community for data reduction was the Discrete Fourier Transform (DFT) [1]. The DFT breaks down a time series of length n into n sine/cosine waves, which when composed together form the original signal. Each wave is represented by a single complex number, known as a Fourier coefficient.

The DFT of a time series $\vec{x} = [t = 0, ..., n-1|x_t]$ is defined as a sequence of complex coefficients $\vec{X} = [f = 0, ..., n-1|X_f]$, represented by

$$X_f = \frac{1}{\sqrt{n}} \sum_{t=1}^{n-1} x_t \exp\left(\frac{-i2\pi ft}{n}\right)$$

where i is the imaginary unit $i = \sqrt{-1}$ and $f = 0, ..., n-1$ [1]. Computing X_f directly is a $O(n^2)$ operation. Fortunately, X_f can be computed in $O(n \log n)$ time by the Fast Fourier Transform (FFT) algorithm.

Table 1. Notation used in this paper

Symbol	Definition
$\vec{x}$	A time ordered series of real values.
n	The length of $\vec{x}$.
x_i, $x[i]$	The i^{th} value of $\vec{x}$ where $0 \leq i < n$.
$\vec{X}$	A vector of features.
N	The length of $\vec{X}$ plus any bookkeeping data.
C_{ratio}	The compression ratio, $C_{ratio} = \frac{N}{n}$.
$maxError$	The maximum distance between an approximating line and the corresponding points in $\vec{x}$.
`merge(s1, s2)`	Returns a new segment consisting of the first point of segment `s1` and the last point of segment `s2`.
`error(s, t)`	Returns the distance between an approximating segment `s` and the original time series `t`.

Referring back to Figure 1, the Fourier result shown is not a direct plot of $\vec{X}$. All plots shown are time domain signals reconstructed from the feature vector $\vec{X}$ (which is in the frequency domain).

One may notice that the resulting data size after the DFT is $2n$. This is because each complex number needs two values for storage: one real and one imaginary. By taking advantage of the symmetric property of the DFT [7], half of the resultant coefficients can be discarded. This still leaves us with the same data size as we started with. The ability to compress data comes from the fact that for most data sets, many coefficients contribute little to the reconstructed signal and can be discarded. Another useful property of the DFT is that the Euclidean distance between two time series in the frequency domain is the same as in the time domain [1]. This allows all data mining operations to be performed in the much smaller feature domain.

Since the DFT essentially measures the global frequency content of the signal, it does not preserve localized time domain events. That is, elements that occur at a specific time and do not repeat are lost in the transformation. As a result, processing large time series with the DFT is expected to produce poor approximations.

In Agrawal's paper [1], he describes the process of discarding Fourier coefficients as simply removing all but the first few. However, the lower frequencies that are kept

in this scheme may not always be the frequencies that contribute most to the signal. A technique proposed by Mörchen [16] describes keeping only the largest coefficients; this results in the most energy preservation from the original signal. Using this scheme, the resulting coefficient array will most likely be sparse and value indices will need to be stored. This means that fewer coefficients can be stored to make room for indices. After some experimentation on fiber property datasets, we discovered that keeping the first few coefficients actually produced higher quality features than the technique proposed by Mörchen [16]. This is perhaps due to the higher number of coefficients retained in Agrawal's approach. Thus, we will use Agrawal's approach of removing all but the first few coefficients.

2.2 Discrete Wavelet Transform

Like the DFT, the Discrete Wavelet Transform (DWT) converts a time domain signal into a frequency domain signal, represented by a set of coefficients. Instead of using sine/cosine waves to reconstruct a signal, the DWT uses many scaled and/or shifted versions of a function called the mother wavelet [16]. Low frequency versions of the mother wavelet model the global contributions of a signal while high frequency versions model local events in a signal [17]. In this way, data features that are localized to a specific time can be represented as wavelet coefficients. This is a contrast to the DFT, which can only model global contributions to the signal. This fact, along with its low time complexity of $O(n)$ [18], has made the DWT a choice feature extraction technique for many data mining applications [19].

The DWT of a time series $\vec{x}$ results in a sequence of real coefficients $\vec{X}$. Like the DFT, many of these coefficients contribute little to the overall signal and can be removed. Therefore, the compression scheme we used for the DFT was also used for all DWT experiments in this paper.

There are many mother wavelet functions to choose from, with each emphasizing different aspects of a signal. The most commonly used wavelet function for the purposes of data mining is the Haar wavelet [20, 2]. Popivanov et al. also note that wavelet functions such as Daubechies [21] may perform better for certain datasets [22]. As a result, in this paper we will use two types of wavelet functions: the Haar and Daubechies-4 wavelets. The Haar transform will be referenced as HWT (the **H**aar **W**avelet **T**ransform) and the Daubechies transform will be referenced as DWT (the **D**aubechies **W**avelet **T**ransform).

2.3 Piecewise Aggregate Approximation

The simple, yet powerful, Piecewise Aggregate Approximation (PAA) technique was first applied to time series data mining by Keogh et al. [7] and Yi et al. [8]. It has been used before in other fields under names such as span-based averaging or simply averaging. The basic idea of this approach is to first divide the input sequence $\vec{x}$ of length n into N equal sized segments. N is determined by the amount of data compression needed. The mean of each segment is then used as a data feature in the resultant series $\vec{X}$, which is calculated using the following equation

$$\vec{X} = [mean(\vec{s}_1), ..., mean(\vec{s}_N)]$$

where $\vec{s}$ is the input sequence $\vec{x}$ divided into N equal sized segments. Due to the simplicity of this method, it has a time complexity of $O(n)$. This makes it particularly attractive for large data sets.

2.4 Piecewise Linear Approximation

Keogh et al. make the suggestion that Piecewise Linear Approximation (PLA) is perhaps the most used feature extraction technique in time series data mining [9]. The basic idea of this approach is to approximate the input sequence using a desired number of straight lines, which we call segments. Since the number of segments is typically much smaller than n, a high level of compression can be achieved.

PLA techniques can be generally classified into three categories:

- **Sliding Window:** A window is grown until a specified error threshold is reached. Even though this method produces relatively poor results, it is heavily used for its online capability. Online algorithms are able to process data in a piece-by-piece fashion, without having the entire dataset available initially. This is contrasted to offline or batch algorithms which need the entire dataset initially.
- **Top-Down:** Starts initially by approximating the entire time series with one segment. It then recursively partitions the segment until all segments fall within a specified error threshold. Without any modifications, this algorithm processes data in an offline fashion.
- **Bottom-Up:** Starts initially with the finest grain approximation possible (i.e., essentially the original data). It then iteratively merges segments until some error threshold is met. This algorithm also processes data offline.

For all techniques, approximating lines can be calculated in a variety of ways. To keep computational complexity low, in this work, lines are calculated using linear interpolation (i.e., use the first and last points of the segment as the approximating line).

As identified by Keogh et al., no PLA technique is best for all data sets [15]. However, the Bottom-Up approach is generally considered the best overall and is the technique that will be used for all experiments in this paper. It has a time complexity of $O\left(n\frac{n}{N-1}\right)$, where $N-1$ is the number of segments [9]. Pseudo code for the generic

Listing 1. The Bottom-Up Segmentation (BUS) algorithm

```
TimeSeries BottomUp(TimeSeries T, double maxError)
{
   Segment[] segments;
   double[]  mergeCost;
   int       i;

   for (i = 0; i < T.Length; i += 2)
      segments[i / 2] = new Segment(T[i], T[i + 1]);

   for (i = 0; i < segments.Length; ++i)
   {
      Segment approxSegment = merge(segments[i],
           segments[i+1]);
      mergeCost[i] = error(approxSegment, T);
   }

   while (min(mergeCost) < maxError)
   {
      i = indexOfMin(mergeCost);
      segments[i] = merge(segments[i], segments[i+1]);
      remove(segments[i+1]);
      mergeCost[i]= error(merge(segments[i], segments[i
           +1]), T);
      mergeCost[i-1]= error(merge(segments[i-1],
           segments[i]), T);
   }

   return new TimeSeries(segments);
}
```

Bottom-**U**p **S**egmentation (BUS) algorithm is shown in Listing 1 (adapted from [9]). In this code listing, $\vec{X}$ is essentially the return value of the `BottomUp` routine (i.e., $\vec{X} =$ `BottomUp(`$\vec{x}, maxError$`)`).

With little modification, the BUS algorithm can accept N as a parameter, instead of $maxError$. While this is desirable, we will use $maxError$ to better support comparison with our new technique in Section 4.

Combinations of the three PLA techniques have been proposed before as well. For instance, Keogh et al. have combined the Bottom-Up approach with the Sliding Window approach [9]. This novel technique produced features similar in quality to Bottom-Up with the addition of online support. Since it still did not outperform Bottom-Up, we will not include it here.

3 Empirical comparison of major feature extraction techniques

In this section we will present a detailed empirical comparison of the major feature extraction techniques described in Section 2. As stated before, the experimentation will be done on data from an industrial synthetic polymer fiber monitoring system; namely, the Attalus$^{\mathrm{TM}}$ Fiber System [13]. This system uses an electromagnetic field to measure numerous properties about the fiber being produced. We will focus on the Magnitude data property which corresponds closely with fiber quantity. Eleven datasets will be tested ranging in size from 256 to 1024 points. All datasets are normalized to produce a mean of zero and standard deviation of one. These datasets were not chosen arbitrarily; they all contain anomalous patterns of fiber quantity. Retaining such anomalous patterns in the reduced representation is a critical goal of this work for the intended novelty detection application [14]. Figure 2 illustrates the eleven datasets used.

3.1 Methodology

To evaluate the suitability of each feature extraction technique, we will adopt a procedure similar to the one used by Keogh et al. [15]. This procedure essentially measures the reconstruction error for a fixed N, where lower reconstruction error implies higher feature quality. Keogh et al. use the simple Euclidean distance function as a measure of the reconstruction error (see Table 1 for notation)

$$error(\vec{x}, \vec{X}) = \sqrt{\sum_{i=0}^{n} (x_i - X_i)^2}$$

where $n = N$. However, in all cases where $C_{ratio} > 0$, n is greater than N so this formula will not work. In addition, points in the original and reconstructed time series will not necessarily 'line up' along the time axis. In our scheme, interpolation is used to find the distance between a point and a line in the reconstructed time series. We call this distance metric the *time aligned Euclidean distance*. This is illustrated in Figure 3. Note that in this work, the time that a data event occurs is important.

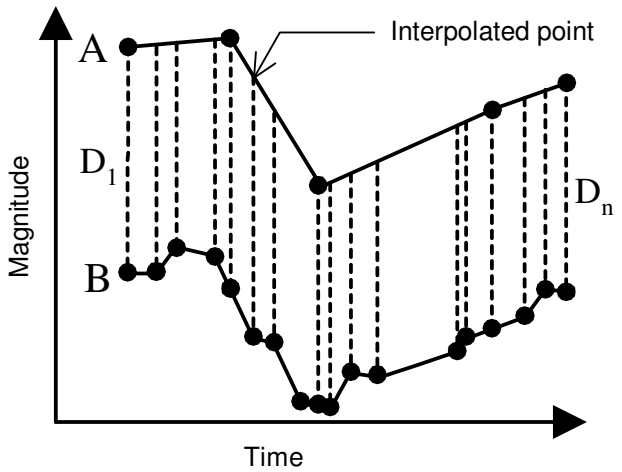

Figure 3. When all points do not line up, interpolation can be used to find the distance. In the above case, the total distance would be $\sqrt{D_1^2 + ... + D_n^2}$. Where B is the input sequence and A is the reduced representation.

For each value of C_{ratio} tested, a technique must produce N data points. For example, for $C_{ratio} = 20\%$ a $n = 1000$ point sample would be reduced to $N = 200$ points. The various techniques are tuned to produce N resultant data points from a specified C_{ratio} as follows:

- **DFT:** $N = \frac{C_{ratio} * n}{2}$, the first N coefficients are used.

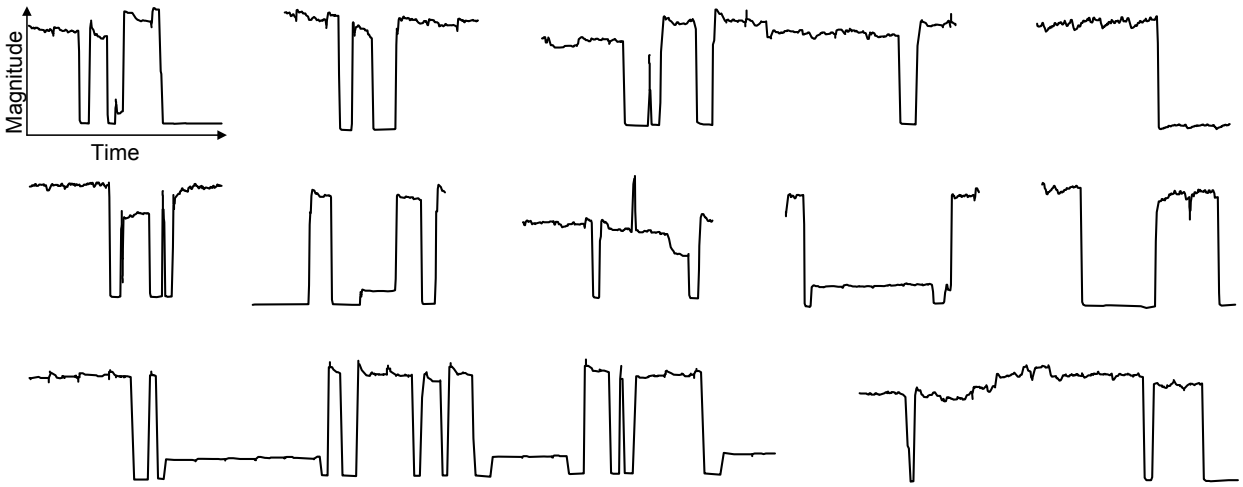

Figure 2. The eleven datasets used in the experiments. Each dataset is a recording of the Magnitude data property over time.

- **DWT, HWT:** $N = C_{ratio} * n$, like the DFT, the first N coefficients are used.
- **PAA:** $N = C_{ratio} * n$, $\vec{x}$ is divided into N segments of length $\frac{n}{N}$.
- **BUS:** N is the number of resulting data points (i.e., $N - 1$ segments) from the BUS algorithm on a pre-calculated maximum error ($maxError$). $maxError$ is defined as the maximum time aligned Euclidean distance allowed between a segment and the original data. The $maxError$ value that produces N data points is calculated by a separate utility that searches $maxError$ values in increments of 0.0001, starting from zero.

We note that the method of calculating $maxError$ values for the BUS algorithm is computationally expensive. This method is only used to facilitate uniform comparison with the other feature extraction techniques. Not all techniques support $maxError$ as a parameter, therefore we must resort to such a procedure. In practice, $maxError$ would be set to a fixed value and N would vary accordingly.

Since we are only concerned with the relative quality of a particular technique, we normalize all results by dividing by the worst technique. The technique with the lowest quality for a particular C_{ratio} has the highest reconstruction error.

3.2 Results

The results of the experiments are summarized in Figure 4. One can readily see that the DFT performs very poorly at all levels of compression. This is most likely due to its inability to preserve the high number of local events in fiber property data.

PAA, DWT, and HWT perform similarly at all compression levels and were generally much better than DFT. None performed better than BUS at any point.

The main result here is that the BUS technique produced the highest quality features at all levels of compression.

4 Enhancement to the BUS technique

The results from the previous section clearly show that BUS produced the highest quality features in fiber property data. It also has a reasonable time complexity which allows it to be applied to much larger datasets. This motivated us to find further improvements to the BUS technique.

Upon examination of the features produced by BUS, we noticed that many data points appeared to be redundant. That is, many data points lay on a nearly straight line and thus did not add much value. These points exist because of the BUS algorithm's initial segmentation. Recall that initially, each segment has a length of two and so all merged segments will have an even length. However, a segment may be best represented by an odd number of data points. This was identified as a possible area for improvement by Keogh et al. [9]. To our knowledge no such improvement has been made.

To improve upon this situation we introduce a new algorithm based on BUS that removes these redundant points. We call our new algorithm IBUS (**I**mproved **B**ottom-**U**p **S**egmentation).

4.1 The IBUS algorithm

The IBUS algorithm essentially adds a post-processing step to the BUS algorithm. Thus, it first calls the BUS algorithm to determine an initial data segmentation. It then scrolls through the BUS data, discarding points that when

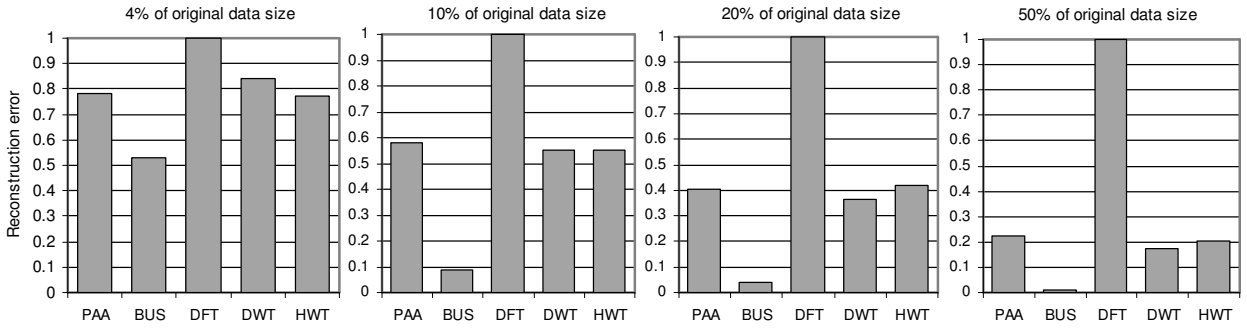

Figure 4. A comparison of the major feature extraction techniques on eleven fiber property datasets. Each histogram bar represents the sum of the reconstruction errors for each dataset.

Listing 2. The Improved Bottom-Up segmentation algorithm

```
TimeSeries ImprovedBottomUp(TimeSeries T, double
     maxError)
{
   TimeSeries resultSeries = BottomUp(T, maxError);

   for (int i = 1; i < resultSeries.Length - 1; ++i)
   {
      Segment approx = new Segment(resultSeries[i - 1],
            resultSeries[i + 1]);

      if (error(approx, T) <= maxError)
      {
         remove(resultSeries[i]);
         i--;
      }
   }

   return resultSeries;
}
```

removed, keep the error below $maxError$. This can also be described as removing points that lie on a straight line.

This process is implemented by looking at 3 points per iteration, and evaluating whether the approximation formed by the 1^{st} and 3^{rd} points have less error than $maxError$. If the approximation is below the $maxError$ threshold, then the 2^{nd} point is removed from the BUS data. The pseudo code for this procedure is shown in Listing 2.

Scrolling through the data takes just $O(N)$ time and if the underlying time series is implemented as a heap, removals take just $O(\log N)$ time [9]. Calculation of the reconstruction error is constant at each iteration. In the worst case, if every iteration produces a removal, the complexity would be $O(N \log N)$. This case is highly unlikely (if not impossible) because the BUS algorithm would need to produce a straight line with all data points lying on it. It is easily seen that the BUS algorithm can not produce such an approximation in any non-trivial case. So the actual complexity is expected to be closer to $O(N)$. In addition, since N is much less than n, even in the worst case the additional complexity of IBUS is negligible.

4.2 Comparison

We repeated all experiments in Section 3, now including the new IBUS algorithm. Since BUS performed best last time, we will compare IBUS relatively to BUS. The $maxError$ for IBUS is calculated in the same way as BUS. So both algorithms may have different $maxError$ values. As shown in Figure 5, for our application IBUS produced higher quality features than BUS at every level of compression. These results are normalized values obtained by dividing by the worst technique. The largest difference between the two methods occurred at 4% of the original data size (i.e., the highest compression level). At this level, IBUS had only half of the reconstruction error that BUS had. As the compression level is decreased, the benefits of using IBUS also decreased. It is suspected that as the compression ratio approaches 100% (i.e., no compression), BUS and IBUS will produce identical results.

In further experiments, we compared how much data reduction IBUS achieves over BUS for the same $maxError$. This is necessary to rule out any benefits derived from the $maxError$ searching utility described in Section 3.1. This also allows us to show the percent improvement for replacing an already deployed BUS solution with IBUS. The BUS $maxError$ values from the previous experiments are used here for both BUS and IBUS. Figure 6 shows the results of these experiments. At each C_{ratio}, IBUS reduces the data size considerably more than BUS. IBUS data size ranged from 61-68% of the BUS data size.

We reiterate that these claims are for fiber property datasets only. However, since no domain knowledge was used in the development of IBUS, it should be applicable to other domains as well.

While it is obvious how IBUS can lower data size by removing redundant points, it may be unclear as to how feature quality can be improved by this process. The improved feature quality comes from that fact that IBUS starts out with a higher quality BUS segmentation (i.e., lower $maxError$) and removes only redundant points to achieve a data size of N. This is contrasted to BUS which needs a higher $maxError$ to get the desired N points. The result

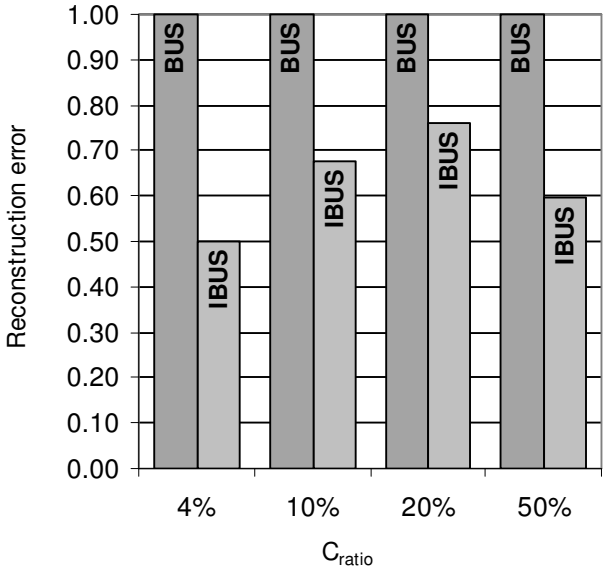

Figure 5. A comparison of the relative reconstruction errors of the BUS and IBUS algorithms.

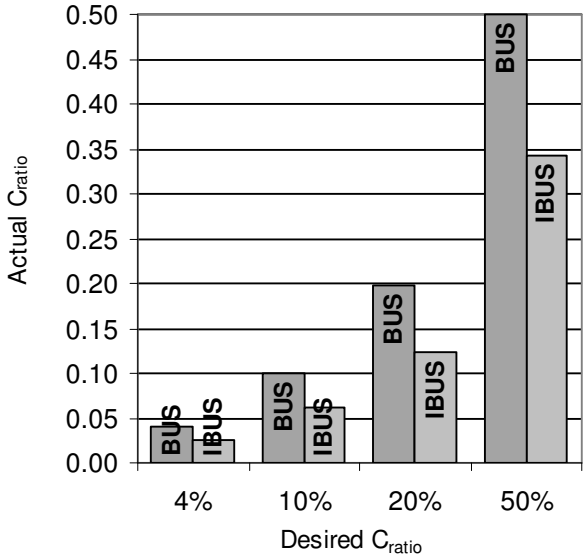

Figure 6. A comparison of the BUS and IBUS algorithms with a fixed maxError. In every case IBUS reduces the data size without reducing quality by removing redundant points.

is that IBUS has similar quality to BUS with a larger N.

5 Conclusions and Future Work

In this work we have shown that PLA methods produce the highest quality features for fiber property datasets. We also introduced a new PLA algorithm (IBUS) which, for our application produced the highest quality features and considerably more data reduction than all currently available feature extraction techniques.

Future work will include evaluating the IBUS algorithm on more diverse datasets and applying it to other BUS based algorithms. One such algorithm mentioned earlier is a combination of the Sliding Window and Bottom-Up PLA approaches [9]. Also, in other related work we intend to evaluate several leading time series novelty detection methods using each of the feature extraction methods described here.

6 Acknowledgments

The authors gratefully acknowledge the funding support provided by INSTRUMAR Limited, the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Atlantic Canada Opportunities Agency (ACOA) through the Atlantic Innovation Fund (AIF).

References

[1] R. Agrawal, C. Faloutsos, and A. N. Swami, Efficient similarity search in sequence databases, *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, London, UK, 1993, 69-84.

[2] K. pong Chan and A. W.-C. Fu, Efficient time series matching by wavelets, *Proceedings of the 15th International Conference on Data Engineering*, Washington, DC, USA, 1993, 126-133.

[3] D. Wu, A. Singh, D. Agrawal, A. E. Abbadi, and T. R. Smith, Efficient retrieval for browsing large image databases, *Proceedings of the 5th International Conference on Information and Knowledge Management*, New York, USA, 1996, 11-18.

[4] E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani, Locally adaptive dimensionality reduction for indexing large time series databases, *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, New York, USA, 2001, 151-162.

[5] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, New York, USA, 2003, 2-11.

[6] C.-S. Perng, H. Wang, S. R. Zhang, and D. S. Parker, Landmarks: A new model for similarity-based pattern querying in time series databases, *Proceedings of the 16th International Conference on Data Engineering*, Washington, DC, USA, 2000, 33-42.

[7] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases, *Knowledge and Information Systems*, 3(3), 2001, 263-286.

[8] B. K. Yi and C. Faloutsos, Fast time sequence indexing for arbitrary lp norms, *Proceedings of the 26th International Conference on Very Large Data Bases*, San Francisco, USA, 2000, 385-394.

[9] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani, An online algorithm for segmenting time series, *Proceedings of the 2001 IEEE International Conference on Data Mining*, Washington, DC, USA, 2001, 289-296.

[10] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan, Mining of concurrent text and time series, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*, Boston, USA, 2000, 37-44.

[11] S. Park, S.-W. Kim, and W. W. Chu, Segment-based approach for subsequence searches in sequence databases, *Proceedings of the 2001 ACM Symposium on Applied computing*, New York, USA, 2001, 248-252.

[12] J. Hunter and N. McIntosh, Knowledge-based event detection in complex time series data, *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making*, London, UK, 1999, 271-280.

[13] M. Chan, Discovering the Value of Information: Online Monitoring of Fiber Characteristics, *Proceedings of the IFAI Technical Forum*, Nashville, USA, 2000.

[14] J. Anstey, D. Peters, and C. Dawson, Discovering novelty in time series data, *Proceedings of the 15th Annual Newfoundland Electrical and Computer Engineering Conference*, St. John's, Canada, 2005.

[15] E. Keogh and S. Kasetty, On the need for time series data mining benchmarks: a survey and empirical demonstration, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, USA, 2002, 102-111.

[16] F. Mörchen, Time series feature extraction for data mining using DWT and DFT, Tech. Rep. 33, Department of Mathematics and Computer Science, University of Marburg, Germany, 2003.

[17] A. Graps, An introduction to wavelets, *IEEE Computational Science and Engineering*, 2(2), 1995, 50-61.

[18] S. G. Mallat, *A Wavelet Tour of Signal Processing* (San Diego, USA: Academic Press, 1999).

[19] T. Li, Q. Li, S. Zhu, and M. Ogihara, A survey on wavelet applications in data mining, *SIGKDD Explorations Newsletter*, 4(2), 2002, 49-68.

[20] Y.-L. Wu, D. Agrawal, and A. E. Abbadi, A comparison of dft and dwt based similarity search in time-series databases, *Proceedings of the 9th International Conference on Information and Knowledge Management*, New York, USA, 2000, 488-495.

[21] I. Daubechies, *Ten lectures on wavelets*, (Philadelphia, USA: Society for Industrial and Applied Mathematics, 1992).

[22] I. Popivanov and R. J. Miller, Similarity search over time-series data using wavelets, *Proceedings of the 18th International Conference on Data Engineering*, Washington, DC, USA, 2002, 212-221.