C3M: A Classification Model for Multivariate Motion Time Series

Dengyuan Wu[1,2,4], Ying Liu[1,5] , Ge Gao[3], Zhendong Mao[1,2,4], Tao He[2,4]
1.Graduate University of Chinese Academy of Sciences
2. Instittue of Computing Technology, CAS
3. University of Virginia 4. Beijing Zhongke Fulong Computer Technology Co. Ltd.
5.Fictitious Economy and Data Science Research Center, CAS
barlooker@gmail.com, yingliu@gucas.ac.cn, gg5j@virginia.edu,
maxdog_mao@hotmail.com, hetao@fulong.com.cn

Abstract

The problem of time series classification has drawn intensive attention from the data mining community. Conventional time series model may be unsuitable for multivariate motion time series because of the large volume of the data, highly correlated dimensions and rapid growth nature. In this paper, we propose C3M, an effective classification model for motion time series classification, which consists of segmentation, dimension ranking and selection, and classification. We propose new segmentation and dimension selection scheme that reduce the storage volume but keep enough valuable information and correlation between different dimensions. Experimental results show that C3M achieves significant performance improvements in terms of both classification accuracy and execution time over conventional schemas.

1. Introduction

The problem of motion time series mining has drawn intensive attention from the data mining community due to its prevalence in numerous applications. Recently, with the prosperity of multimedia technology, more and more types of domain-specific time series data have emerged including motion capture data, video surveillance, animation production, virtual reality, etc.. As a result, time series data retrieval techniques are in strong demand.

Motion time series classification/clustering has been extensively studied. However, due to its unique characteristics, classifying multivariate motion time series poses several challenges:

1) Most of multivariate motion time series are collected from sensors. which usually in high dimensionality and long sequence with large amount of noise and error, retaining which works out to undermine the performance of overall classification model.

2）Since motion time series are aggregate data, the dimensions are highly correlated. Methods that processing each data dimension separately would result in large loss of useful information.

3) The size of multivariate motion time series data grows very fast. Highly efficient representation methods are in strong demand.

To address the challenges from multivariate motion time series, we propose an effective classification model, called C3M, which consists of three phases: feature extraction, dimension ranking & selection and classification. In the feature extraction phase, we perform multivariate segmentation on the entire set of the dimensions of the time series data, that is, we split a time series item into small sub-time series, called "segment". The boundaries of these segments are consistent with each other. For each segment, C3M extracts only one pair of coefficients of the Chebyshev polynomials and use their quotient as the local feature. Dimension ranking & selection phase adopts a supervised manner, where feature dimensions are ranked based on their combined information gain ratio, an indicator of their predictive capability. C3M selects the Top-K predictive dimensions to represent the original multivariate data. Finally, in the classification phase, we use the selected features to build a decision tree model and make classification.

The rest of this paper is organized as follows: in Section 2, the related work is introduced. Section 3 provides the background knowledge for our approach. Our proposed methods are described in Section 4, followed by the experiments and results in Section 5. Conclusions and future work are presented in Section 6.

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.716

2. Related works

Many researchers have been working on the classification problem of time series. Typical approaches involve segmentation, feature extraction, similarity measures, and the deployment of a machine learning model.

The most essential part of these research focuses on feature extraction and similarity computation. In [1], Discrete Fourier Transform (DFT) is introduced to map time sequence to frequency domain. Discrete Fourier Coefficients are extracted as features and the similarity of two time series with equal length is defined as the distance in the feature space. DFT can achieve good accuracy but not perfect, and the algorithm in [1] can not be used in treating time series of different length. PAA [2] is easy to be understood and implemented, fast in calculation, with equivalent effectiveness. Its basic idea is to decompose the time series into a sequence of equal-length-box basis function and measure the similarity of time series under the Euclidean distance metric. PAA achieves certain level of success in prediction accuracy, but the time series data lost lots of useful local features. Two "boxes" of time series that have same mean but different variance will be represented by the same features. Berndt and Clifford introduces Dynamic Time Warping (DTW) [3] as a similarity measure, which allows an elastic scaling of the time axis to accommodate time series that are similar but out of phases in the time axis. Due to its flexibility, DTW is more robust and can provide better performance in terms of classification accuracy compared to the Euclidean distance based techniques mentioned above. However, it has $O(n^2)$ time complexity, which is intolerable for large databases or long sequences.

Our C3M differs with the above models and methods in the following aspects:

1) It provides a complete solution for the multivariate motion time series classification, including feature extraction, dimension ranking & selection and classification, rather than a single algorithm.

2) It applies Chebyshev approximation in the segmentation and uses the quotient of the first two Chebyshev coefficients as the features, thus reducing the data volume while still maintaining the correlation between different dimensions.

3) It proposes a new supervised dimension ranking and selection strategy based on information theory that removes noise dimension and further reduces the data size.

3. Chebyshev Polynomials and Coefficients

Before introducing C3M, we would like to introduce Chebyshev approximation, a very important definition we use in C3M. For more details, please refer [4][5][6].

Chebyshev Polynomials are often used to approximate the functions that are difficult or impossible to compute. The Chebyshev Polynomial $\{T_n(x)\}$ in x of degree n is defined as $T_n(x)=\cos(n\cos^{-1}x)$ for $x\in[-1,1]$ .The definition can be easily extend to any interval [a, b] by various schema which we will not discuss here. Combining the trigonometric identity:

$$\cos n\theta+\cos(n-2)\theta=2\cos\theta\cos(n-1)\theta,$$

we obtain the fundamental recurrence relation:

$$T_n(x)=2xT_{n-1}(x)-T_{n-2}(x),\ n=2,3,\ldots.$$

with initial conditions $T_0(x)=1,\ T_1(x)=x.$

The orthogonal property of Chebyshev approximation motivates us to use Chebyshev Polynomials in C3M. The set of Chebyshev Polynomials is orthogonal on (-1,1) with respect to the weight function:

$$\omega(x)=(1-x^2)^{-1/2}.$$

That is, the Chebyshev Polynomials satisfy:

$$\int_{-1}^{1}\frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}dx=\begin{cases}0, & n\neq m\\ \frac{\pi}{2}, n=m\neq 0\\ \pi, n=m=0\end{cases}$$

The inner product of two Chebyshev Polynomials is unequal to zero only when they are identical. The orthogonality of Chebyshev Polynomials allows them to be used to approximate any function. Given a function $f(x)$, the best polynomial approximation of degree n is

$$f(x)\approx\sum_{k=0}^{n}c_kT_k(x)$$

where c_k is the Chebyshev Coefficient of $T_k(x)$. Theoretically, c_k can be calculated as ,

$$c_k=\frac{1}{n}\textstyle\sum_{k=0}^{n}f(x) \qquad (k=0);$$

$$c_k=\frac{2}{n}\textstyle\sum_{k=0}^{n}f(x)T_k(x) \qquad (k>0);$$

Another important property that distinguishes Chebyshev Polynomails from other polynomials is its minimax property. That is, for all monic polynomials of degree n, $P_n(x)$ (monic polynomials are those polynomials the highest order of which is 1).

$$2^{1-n}=\max_{x\in[-1,1]}|2^{1-n}T_n(x)|\leq\max_{x\in[-1,1]}|P_n(x)|$$

We use Chebyshev Approximation in our C3M for the following reasons:

1) Chebyshev Approximation has been used in the trajectory data indexing to extract global feature of trajectory data [16] and achieves great success.

2) Its superior property in both $\mathcal{L}_2$ approximation and $\mathcal{L}_\infty$ approximation makes it a good candidate for time series segmentation in first segments' generation and final feature extraction.

3) The computation of Chebyshev coefficients does not include time consuming complex computation. We only need the low order Chebyshev coefficients. In our experiments we employ the NAG library [7] to calculate it.

4. C3M MODEL

4.1 Overview

In this section, we introduce our C3M model systematically. C3M consists of three phases: feature extraction, dimension ranking & selection, and classification. The first two phases actually perform the data preprocessing which is a necessary step since due to large size of raw multivariate motion time series. After data preprocessing, we feed the extracted feature vectors, along with their corresponding class labels, into the decision tree to train a classifier. When C3M is used to classify time series, we compute the feature vectors of the data in the testing set and apply classifier to predict the class labels. Table 1 lists the notations used in the remainder of this paper. Figure 1 illustrates the entire flow of C3M.

4.2 Feature Extraction

In this phase, we first perform segmentation, and then extract features from the local segments. The motivation behind is our observation that local features make more sense for motion data. Motion data of different classes often differ slightly in the overall sequence shape but are quite different in local sub-time series. Therefore, the features of the segments after segmentation may enlarge the difference of motion data from different classes.

From our observation, different dimensions of multivariate motion time series may have strong correlations for example, if we put two sensors on our knees when we walk and we carefully analyze the y-axis position of our knees, we would found that their positions change with each other in a regular pattern. In order to keep the correlation between different dimensions of motion time series, we consider all the dimensions together: we set the segmentation boundary of different dimensions consistently when we perform segmentation. In this way, we succeed in keeping the correlations between different dimensions. The feature extraction process of C3M is described in Figure 2. C3M first divides a sequence into S segments (S is a user specified parameter) using bottom-up segmentation algorithm [8], which gives more satisfactory results than other segmentation algorithms. Similar to existing bottom-up algorithms, segmentation in C3M begins with constructing an initial set of segments representing the finest possible approximation of the data, so that n/2 segments are obtained for an n-length multivariate time series (lines 1-6). Then, it iteratively merges the pair of adjacent segments that involves the lowest cost according to some cost function until only S segments are remained (lines 7-13).

Table 1. Notations Used in This Paper

Symbol	Definition
n	Length of a multivariate time series item
N	Number of multivariate time series items in the training set
d	Number of dimensions of a multivariate time series item
S	Number of segments of a multivariate time series item
K	Number of dimensions selected
$T_n(x)$	Chebyshev Polynomial of degree n
c_n	Chebyshev Coefficient of $T_n(x)$ in the approximation

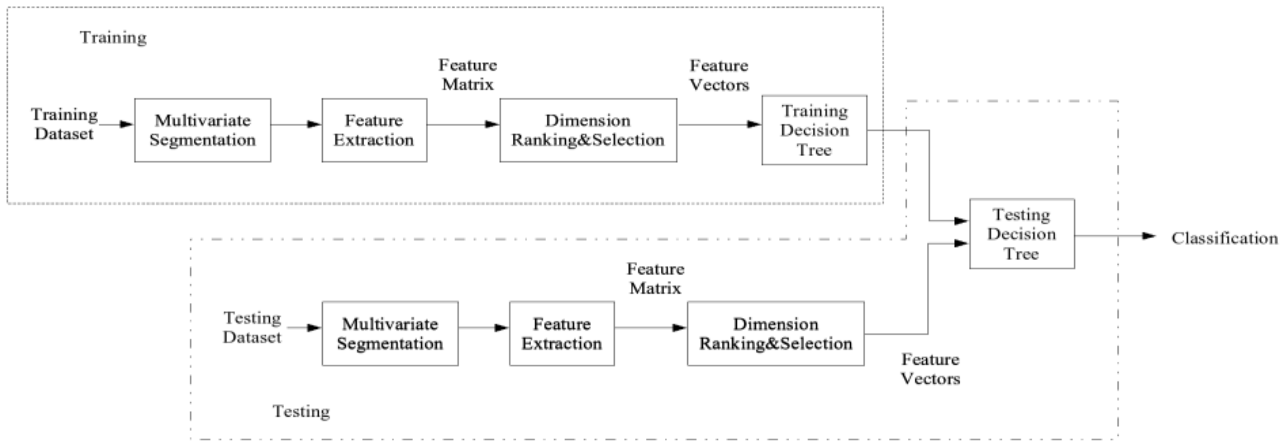

Figure 1 Overview of C3M

Input: T {One multivariate time series item of length n with d dimensions}, S {the required number of segments}

```
1:  $Seg \leftarrow \emptyset$;
2:  $FeatureMatrix \leftarrow \emptyset$;
3:  for i=1 to n/2 do
4:      $Seg(i) \leftarrow T[2i-1:2i]$;
5:      $Chebyshev(i)$ ←approximate $Seg(i)$
        with Chebyshev Polynomials;
6:      $Error(i) \leftarrow$
        $calculateError(Seg(i), Chebyshev(i))$;
7:  end for
8:  for i=n/2 to S+1 do
9:      for j=1 to i-1 do
10:         $[j, cost(j)]$ ←compute the cost of
            $merge(Seg(j), Seg(j+1))$ according to the cost
            function;
11:     end for
12:     $J$ ←choose j with $min(cost(j))$;
13:     $Seg(J) \leftarrow merge(Seg(J), Seg(J+1))$;
14: end for
15: for i=1 to S
16:     $FeatureMatrix[,i] \leftarrow Chebyshev(i)$;
17: end for
```

Figure 2. Algorithm for extracting features from a multivariate time series item.

What our segmentation algorithm differs with the conventional one is the definition of cost function. Firstly, in order to implement multivariate segmentation, a cost function is constructed to report the information lost of all the dimensions by merging pairs of adjacent segments (line8-line14). Secondly, the computation of the cost function involves Chebyshev approximation. That is, for each segment, the first-order Chebyshev polynomials are used to approximate every dimension of the multivariate time series. This would naturally yield deviation of original data points from the fitting curve (discrete points). Let $d_1, d_2, \ldots, d_j$ denote the vertical distances from the j data points in a segment to their fitting curve, we can define the normalized error, for the corresponding ith dimension, as

$$e_i = \frac{\sum_{k=1}^{j} {d_k}^2}{j}$$

By summing up the errors over all the dimensions, we obtain the overall error of Chebyshev approximation for a given segment:

$$e_{seg} = \sum_{m=1}^{d} e_m$$

Informally, e_{seg} can be considered as a measure of how much information is lost by adopting multivariate segmentation. When adjacent segments Seg_s *and* Seg_{s+1} is assumed to be merged into one segment Seg_s', we generate Chebyshev approximation for Seg_s' and approximation error e_{seg} is recomputed. Then the information lost by merging Seg_s and Seg_{s+1} can be expressed as:

$$cost = e_{seg_s'} - \left(e_{seg_s} + e_{seg_{s+1}}\right) \quad \text{(line10)}$$

Smaller value of cost means the merge of the two segments may yield less information loss and better final segments. We choose to merge the two segments with the smallest cost, the merge of which may cause the lowest information loss.

After obtaining the specified number of segments, C3M starts to compute the feature for each segment. Given a Chebyshev Approximation $c_0T_0(x)+c_1T_1(x)$, we use the quotient of the first two Chebyshev Coefficients, c_1/c_0, as the feature for each dimension in each segment. In this way, only one order local feature is used to represent the sequence of each dimension in each segment. As a result, a time series item by a $d \times n$ matrix can be converted to a $d \times S$ feature matrix by feature extraction. This will result in a large reduction in volume. The complexity of our Segmentation algorithm is the same as that in [8]: $O(n^2/S)$

4.3 Dimension Ranking and Selection

Dimension ranking and selection in C3M adopts a supervised manner. It ranks each dimension based on its predictive capability. We learn from the information theory that information gain and information gain ratio can be used to measure predictive capability. Information gain ratio is superior to information gain. However, information gain ratio is only defined for single dimension attribute while every dimension of time series contains S extracted feature value. Thus information gain ratio can not be directly used in our dimension ranking and reduction. In this paper, we extend the definition of information gain ratio to combined attributes.

Given a collection S, with data items belonging to c classes, the information needed to classify a given data item $I(S)$ is:

$$I(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

where p_i is the portion of S belonging to class i.

We set S_v the subset of S for which variable V has the value v ($S_v = \{s \in S | V(s) = v\}$). Then the information gain, split information and information gain ratio of a single variable V relative to the collection S is defined as follow:

Given a data set S, the information gain of attribute V is:

$$Gain(S,V) \equiv I(S) - \sum_{v \in Values(V)} \frac{|S_v|}{|S|} I(S_v)$$

$$SplitInfo(S,V) \equiv - \sum_{v \in Values(V)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

$$GainRatio(S,V) \equiv \frac{Gain(S,V)}{SplitInfo(S,V)}$$

In order to evaluate the predictive capability of each dimension of S feature that are extracted from each dimension of origin time series and in fact combinations of features, we need to extend the definition of information gain ratio for combined variables in the following way:

1. S: multi-attribute data set
2. $\{V_1, V_2, ..., V_i, ..., V_n\}$: a set of discrete variables related to S
3. $\{N_1, N_2, ..., N_i, ..., N_n\}$: Number of possible value of $\{V_1, V_2, ..., V_i, ..., V_n\}$, respectively.
4. $S_{\langle v_1, v_2, ..., v_i, ..., v_n \rangle}$: the subset in S that satisfy: $V_1 = v_1$, $V_2 = v_2$, ..., $V_i = v_i$, ..., $V_n = v_n$
5. V: a single variable of S
6. V': a conbined variable of S that has $\{V_1, V_2, ..., V_i, ..., V_n\}$ as variables.

Given a set of discrete variables $\{V_1, V_2, ..., V_i, ..., V_n\}$ related to a dataset S, with $N_1, N_2, ..., N_i, ..., N_n$ possible values respectively, the combined variables V' are defined to have values

$$Values(\langle V_1, V_2, \dots, V_i, \dots, V_n \rangle) = \{(v_1, v_2, \dots, v_i, \dots, v_n) | v_i \epsilon Values(V_i)\}.$$

There are $N_1 \times N_2 \times ... \times N_i \times ... \times N_n$ possible values in this set. If a data item $s \in S$ has values v_1^0, v_2^0, ..., v_i^0, ..., v_n^0 on variables $V_1, V_2, ..., V_i, ..., V_n$ respectively, it belongs to $S_{(v_1{}^0, v_2{}^0, \dots, v_i{}^0, \dots, v_n{}^0)}$, the subset of S for which variable $<V_1, V_2, ..., V_i, ..., V_n>$ has the value $(v_1^0, v_2^0, ..., v_i^0, ..., v_n^0)$. Based on the above generalization, we can define the information gain, as well as the information gain ratio, for combined discrete variables in the similar way with a single variable:

$$\text{Gain}(S, V') = Gain(S, \langle V_1, V_2, \dots, V_n \rangle) \equiv I(S) - \sum_{v \in Values(\langle V_1, V_2, \dots, V_n \rangle)} \frac{|S_{\langle V_1, V_2, \dots, V_n \rangle}|}{|S|} I(S_{\langle V_1, V_2, \dots, V_n \rangle}),$$

$$\text{SplitInfo}(S, V') = SplitInfo(S, \langle V_1, V_2, \dots, V_n \rangle) \equiv - \sum_{v \in Values(\langle V_1, V_2, \dots, V_n \rangle)} \frac{|S_{\langle V_1, V_2, \dots, V_n \rangle}|}{|S|} \log_2 \frac{|S_{\langle V_1, V_2, \dots, V_n \rangle}|}{|S|}$$

Thus,

$$\text{GainRatio}(S, V') \equiv \frac{\text{Gain}(S, V')}{\text{SplitInfo}(S, V')}$$

In C3M, each dimension is treated as a combination of multiple variables. Given a training dataset consisting of N multivariate time series items and their corresponding class labels, C3M first computes the information gain ratio for each of the d dimensions by using the methods mentioned in [17] to discretize the continuous attributes. Then, C3M ranks the dimensions according to the information gain ratio. Finally, the top K dimensions are selected. Thus, a d*n time series is compressed to K*S which result in a great volume reduction (d>>K, n>>S).

In subsequent processing, only features in the selected dimensions are utilized to represent the original multivariate time series and applied to the decision tree model for training and classification.

5. EXPERIMENTAL RESULTS

In order to demonstrate the effectiveness of C3M, we evaluate the performance of C3M in terms of precision and execution time. We conduct comprehensive experiments on real multivariate motion time series datasets [9, 10, 11]. Details of the datasets are in Table 1.

Table 1 Experimental Datasets

Data set	Number of Dimension	Number of Classes	Number	Length
Gun-Point	1	2	200	150
Hips	3	3	150	100
FSW	3	3	150	100
Human-Gait	66	15	540	133

In our experiments, we compare the performance of C3M with several art-of-state classification techniques in time series literature to show the superiority of C3M model, including DFT [1, 2], PAA [4], DTW [7, 12]..

We test the algorithms using 10-fold cross-validation. We use 10 % of dataset as testing items, and average the results of the 10 folds [15]. For DFT, PAA and DTW, we employ the classical 1-nearest neighbor classification technique. All the experiments were run on Intel 1.5 GHz and 1.5 GB RAM.

Since the computational complexity of DTW, $O(n^2)$, is often unacceptable, especially when applied to datasets with long sequences or large size. In our experiments, we use 10%-size warping window to constrain the warping path, as in [12, 13, 14].

5.1 Precision

Figure 3 compares the classification precision of C3M and DFT, PAA and DTW. The figure consists of four graphs, each for one multivariate time series dataset. For C3M, DFT and PAA, we vary *S*, the number of coefficients (segments) extracted from each dimension, from 2 to 10. Hence, the length of the feature vectors fed to the decision tree is equal to the product of the number of the dimensions and *S*. For C3M, what we reported here is the result from the optimal set of dimensions. And for DTW, we just report its classification precision, not affected by *S*.

In Figure 3, C3M shows the best classification precision on all the datasets and outperform other approach most when treating Human Gait data set that have the most dimensions. It indicates that our feature extraction and dimension selection method can keep the correlations between different dimensions better. In contrast, DFT and PAA perform the worst almost all the time.

It should also be pointed out that DTW does not always provide much better accuracy than PAA and DFT. This may be due to the involvement of redundant dimensions of DTW, which undermines its effectiveness in classification while our C3M removes the noise dimension.

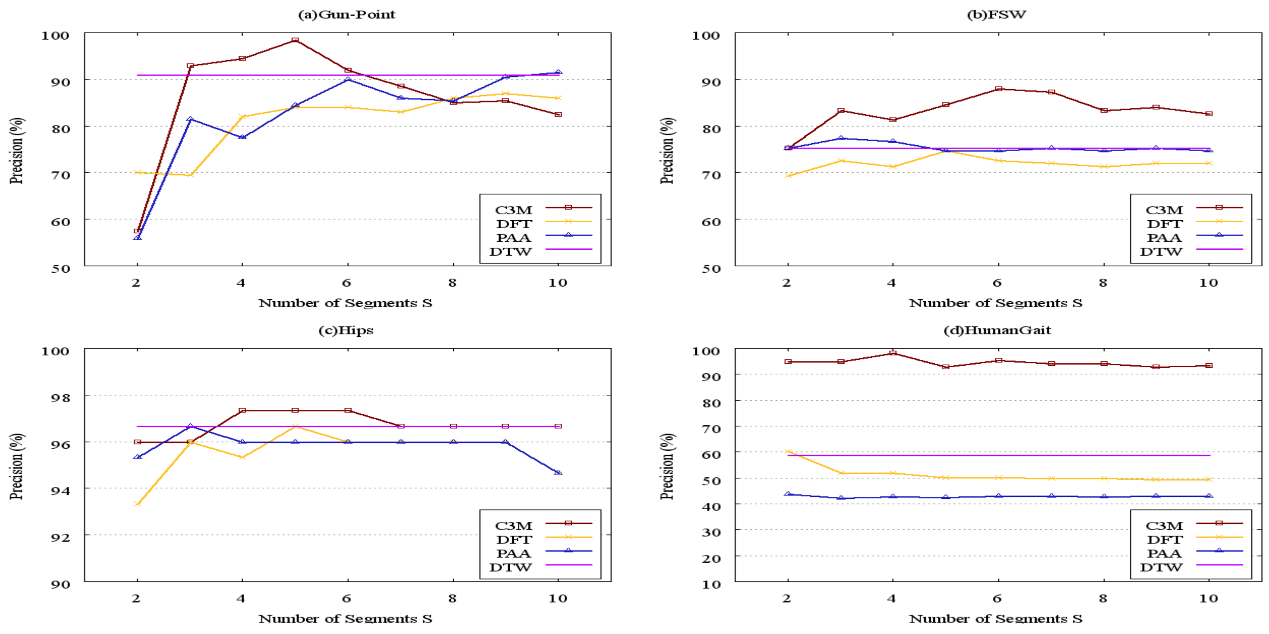

Figure 3 Classification Precision Comparison

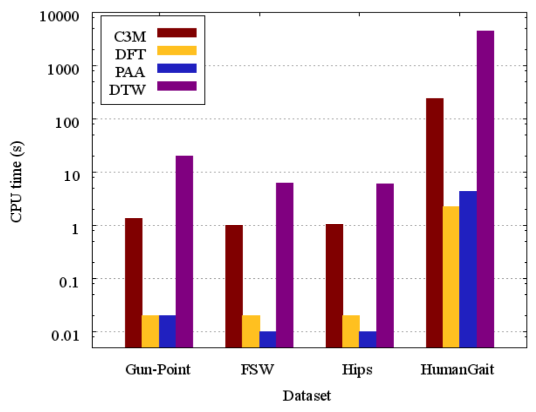

Figure 4. Execution Time Comparison

5.2 Execution Time

As mentioned above, the well-established schema DTW and its variants are almost optimal in terms of precision for most univariate time series classification scenario. But their extremely high computation cost limits their application. Our C3M model is primarily designed to reduce classification CPU cost without compromising the precision. Figure 3 compares the CPU execution time for each approach on all the four datasets. Note, all the results are reported when each approach achieves its highest precision in Figure 4.

As Figure 4 shows, DTW shows the upper bound and the naïve DFT and PAA stay at low CPU cost. Although C3M incurs more execution time than the naïve approaches, it costs one order of magnitude of CPU time less than DTW. We can also see that the classification cost of DTW suffers more from the effect of sequence length than that of C3M. The length of FSW sequence is 100 and Gun-Point sequence is 150. DTW costs 222% more CPU time when classifying Gun-Point than FSW, while our approach only costs 33% more CPU time. This result is due to the fact that the computation complexity of DTW is $O(n^2)$, while the computational complexity of C3M is lower and the CPU cost of C3M is largely dependent on the length of the feature vector applied to the decision tree, which remains relatively stable.

6. Conclusion

In this paper, we present the C3M model, a simple but successful model for motion data classification. We analyzed the reason why C3M can have very good prediction accuracy in relatively acceptable computing time, and we defend our assertion with sufficient experiments on various motion data sets by comparing with some classic models PAA, DFT and DTW which have already been published and achieve great success.

In the further, we may extend and improve C3M to multivariate time series of different types. We may also try more feature extraction methods that may incur less computation time.

Acknowledgement
This project is partially supported by the National Natural Science Foundation of China, #90718042, and the President Fund of Graduate University of Chinese Academy of Sciences.

7. Reference

[1] Rakesh Agrawal, Christos Faloutsos, Arun N. Swami. Efficient Similarity Search In Sequence Databases. FODO 1993: 69-84
[2] Eamonn J. Keogh, Michael J. Pazzani. A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases. PAKDD 2000: 122-133
[3] Donald J. Berndt, James Clifford. Finding Patterns in Time Series: A Dynamic Programming Approach. Advances in Knowledge Discovery and Data Mining 1996: 229-248
[4] J. C. Mason and D. Handscomb. Chebyshev Polynomials. Chapman & Hall, 2003.
[5] J.R. Quinlan. Introduction of decision tree. Machine Learning, 1(1): 81-106
[6] J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
[7] www.nag.co.uk
[8] Eamonn J. Keogh, Selina Chu, David Hart, Michael J. Pazzani: An Online Algorithm for Segmenting Time Series. ICDM 2001: 289-296
[9] Keogh, E. The UCR Time Series Data Mining Archive [http://www.cs.ucr.edu/~eamonn/TSDMA/index.html]. Riverside CA. University of California - Computer Science & Engineering Department
[10] www.emotek.com
[11] Rawesak Tanawongsuwan, Aaron F. Bobick: Performance Analysis of Time-Distance Gait Parameters under Different Speeds. AVBPA 2003: 715-724
[12] Eamonn J. Keogh: Exact Indexing of Dynamic Time Warping. VLDB 2002: 406-417
[13] Rabiner, L., Rosenberg, A. & Levinson, S. (1978). Considerations in dynamic time warping algorithms for discrete word recognition. IEEE Trans. Acoustics, Speech, and Signal Proc., Vol. ASSP-26, pp. 575-582.
[14] Russell S, Norvig P (1995) Artificial Intelligence: A Modern Approach. Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, New Jersey.
[15] Steven Salzberg: On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. Data Min. Knowl. Discov. 1(3): 317-328 (1997)
[16] Yuhan Cai, Raymond T. Ng: Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials. SIGMOD 2004:599-610
[17] Usama M. Fayyad, Keki B. Irani. On the Handling of Continuous-Valued Attributes in Decision Tree Generation. Machine Learning 8,1992: 87-102