Fast Algorithms for Coevolving Time Series Mining

Lei Li
supervised by Christos Faloutsos
Computer Science Department,
Carnegie Mellon University
5000 Forbes Ave, PA 15213, USA
leili@cs.cmu.edu

***Abstract*— In this paper, we present fast algorithms on mining coevolving time series, with or with out missing values. Our algorithms could mine meaningful patterns effectively and efficiently. With those patterns, our algorithms can do forecasting, compression, and segmentation. Furthermore, we apply our algorithm to solve practical problems including occlusions in motion capture, and generating natural human motions by stitching low-effort motions. We also propose a parallel learning algorithm for LDS to fully utilize the power of multicore/multiprocessors, which will serve as corner stone of many applications and algorithms for time series.**

I. Introduction and Motivation

Time sequences appear in numerous applications, like sensor measurements, mobile object tracking, data center monitoring, computer network monitoring, motion capture sequences, environmental monitoring (like automobile traffic and chlorine levels in drinking water) and many more.

In these scenarios, it is very important to understand the patterns in the data such as correlation and evolving behavior. By better mining the patterns, it will help make better prediction and many further tasks in individual scenario. Our goal is to develop algorithms for mining and summarizing any time series data, and we list here a few motivation settings.

Motion capture sequences Motion capture (mocap) is a technique for creating realistic motion animations. Such technique is not a puppy creature, but widely used in several multi-billion industries such as computer game and movie industry. The revenue merely in game industry spans over 10 billion in US dollars.

We are particularly interested in the following important problems:

- How to create new and natural human motions from a motion capture database?
- How to characterize natural human motions?
- How to recover the occlusion that is common in mocap sequences?

Sensor data Wireless sensors are useful in many situations where resource (e.g. power) are limited for measurement, such as monitoring chlorine levels in drinking waters systems and automobile traffic in major infrastructure roads . Sensor data are usually in streaming fashion, and well suited in the context of our time series mining algorithms.

Some typical problems in sensor data include:

- How to find patterns, such as correlation and time shifting, in sensor data?
- How to detect anomalies in sensor data? For example, detecting the outbreak in drinking water by monitoring the chlorine levels in drink water system.
- How to find incorrect observations or recover missing values in sensor data? It is common to have missing observations due to various factors, say out of battery.

Computer network traffic Another important time series data comes from the computer communication, such as port to port tcp/ip traffic and web click streams . We are particularly interested in the following problems:

- How to find patterns in such time series? How to group similar traffic patterns together? The challenge lies in the bursty nature of these data sequences.
- How to identify intrusion/anomalies in such computer network traffic data?

In this paper, we present our work on some of these problems. At large, we focus on the theme of mining multiple co-evolving sequences, with the goal of developing fast algorithms for finding patterns, summarization, and anomalies. In the following sections, we will describe our approaches to mine meaningful patterns from multiple coevolving time sequences and use those patterns for solving real problems in motion capture and sensor data monitoring. Particularly, we present three pieces of work here:

1) Mining with missing values;
2) Natural motion stitching;
3) Parallelizing on multicore/multiprocessor computers.

For each work, we will describe the basic problem setting, the main idea of our proposed methods, and the results.

II. Mining with Missing Values

A. Problem Definition

Given multiple time sequences with missing values, we propose DynaMMo which summarizes, compresses, and finds latent variables. The idea is to discover hidden variables and learn their dynamics, making our algorithm able to function even when there are missing values.

B. Main Idea

Our main idea is to simultaneously exploit smoothness and correlation. Smoothness is what splines and linear interpolation exploit: for a single time-sequence (say, the left-elbow x-value over time), we expect successive entries to

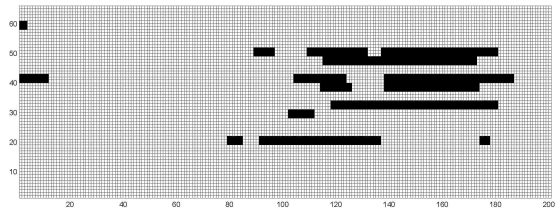


Fig. 1. Occlusion in handshake motion. 66 joint angles (rows), for ≈ 200 frames. Dark color indicates a missing value due to occlusion. Notice that occlusions are clustered.

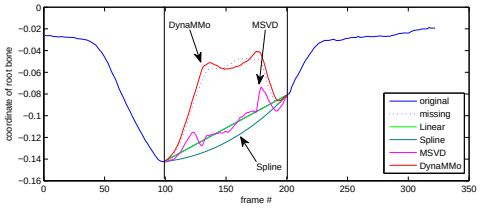


Fig. 2. Reconstruction for a jump motion with 322 frames in 93 dimensions of bone coordinates. Blue line: the original signal for *root bone* z-coordinate - the dash portion indicates occlusion from frame 100 to 200. The proposed DynaMMo, in red, gets very close to the original, outperforming all competitors.

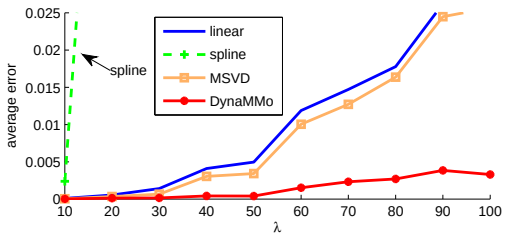


Fig. 3. Average error for missing value recovery on a sample mocap data (subject#16.22). Average rmse over 10 runs, versus average missing length λ(from 10 to 100). Randomly 10.44% of the values are treated as "missing". DynaMMo (in red solid line) wins. Splines are off the scale.

have nearby values ($x_n \approx x_{n+1}$). Correlation reflects the fact that sequences are not independent; for a given motion (say, "walking"), the left-elbow and the right-elbow are correlated, lagging each other by half a period. Thus, when we are missing x_n, say, the left elbow at time-tick n, we can reconstruct it by examining the corresponding values of the right elbow (say, y_{n-1}, y_n, y_{n+1}). This two-prong approach can help us handle even "*black-outs*", which we define as time intervals where we lose track of all the time-sequences.

The main contribution of our approach is that it shows how to exploit both sources of redundancy (smoothness and correlation) in a principled way. Specifically, we show how to set up the problem as a Dynamic Bayesian Network and solve it efficiently, yielding results with the best reconstruction error and agreeing with human intuition. Furthermore, we propose several variants based on DynaMMo for additional time series mining tasks such as forecasting, compressing, and segmentation.

One benefit of DynaMMo is that it helps compress time series more compactly and accurately. The basic compression idea is to store the learned parameters by DynaMMo and values of hidden variables for a subset of time ticks. Based on different strategies to choose the subsets, hence consequently different compression ratio and accuracy, we proposed three variants of DynaMMo compression, fixed compression (DynaMMo$_f$), adaptive compression (DynaMMo$_a$), and optimal compression (DynaMMo$_d$).

As a further merit, DynaMMo is able to segment the data sequence. Intuitively, this is possible because DynaMMo identifies the dynamics and patterns in data sequences, so segments with different patterns can be expected to have different model parameters and latent variables. We use the reconstruction error as an instrument of segmentation, making pieces whenever the error spikes.

We refer reader to [1] for more technical details on DynaMMo, its compression, decompression and segmentation algorithms.

C. Results

We presented experiments on motion capture sequences[1] and chlorine measurements and demonstrated that our proposed DynaMMo method and its extensions (a) can successfully learn the latent variables and their evolution, (b) can provide high compression for little loss of reconstruction accuracy, and (c) can extract compact, but powerful features, for sequence forecasting, interpretation and segmentation, (d) scalable on duration of time series.

Recovering missing values Figure 2 shows the reconstructed signal for an occluded jumping motion. DynaMMo gives the best result close to the original value. Figure 3 shows the reconstruction error versus the occlusion length on motion capture dataset: the error grows little with increasing occlusion length. Compared with other alternative methods, DynaMMo achieves the best performance among the four methods. The results are confirmed by experiments on other additional datasets [1].

Compression Figure 4 shows the decompression error (in terms of reconstruction square error) with respect to compression ratio compared with the baseline compression using a combined method SVD and Linear Interpolation. DynaMMo$_d$ wins especially in high compression ratio.

Segmentation Figure 5 shows the reconstruction error from segmentation experiment on a real human motion sequence in which an actor running to a complete stop. Two (y-coordinates of left hip and femur) of 93 joint coordinates are shown in the top of the plot. Note the spikes in the error plot coincide with the slowdown of the pace and transition to stop.

III. Natural Motion Stitching

A. Problem Definition

Given two motion-capture sequences that are to be stitched together, how can we assess the goodness of the stitching?

[1] http://mocap.cs.cmu.edu

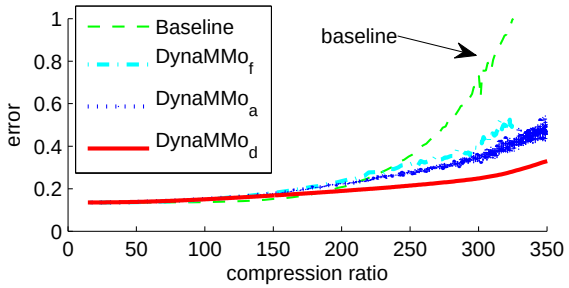


Fig. 4. Compression for Chlorine dataset: Reconstruction error versus compression ratio. Lower is better. DynaMMo$_d$ (in red solid) is the best.

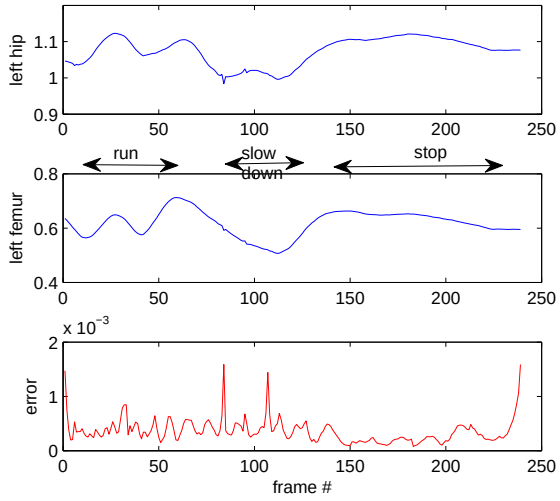


Fig. 5. Reconstruction error plot for segmentation on a real motion capture sequence in 93 dimensions (subject#16.8) with 250 frames, an actor running to a complete stop, with left hip and femur y-coordinates shown in top plots. The spikes in bottom plot coincide with the slowdown of the pace and transition to stop.

A good distance function is important for the generation of realistic character motion from motion capture databases. We propose a novel distance function to pick natural stitching points between human motions. To motivate our work, we demonstrate that a straightforward, ad-hoc approach may lead to poor stitchings like euclidean distance, time-warping and geodesic joint-angle distance, because none of them tries to capture the dynamics of the stitching as explicitly as our upcoming proposal does.

Problem 1 (Stitching Naturalness): **Given** a query sequence $\mathcal{Q}$ of T points in m-dimensional space with take-off point $\vec{q}_a$, and a data sequence $\mathcal{X}$ of $\mathrm{T_x}$ points of the same dimensionality with landing point $\vec{x}_b$, **find** a function to assess the goodness of the resulting stitched sequence, i.e. $\vec{q}_1, \ldots, \vec{q}_a, \vec{x}_b, \ldots, \vec{x}_{\mathrm{T_x}}$.
The goal is that the "goodness" metric should be low if humans consider the stitching to be natural. Once we obtain a qualified distance function, we can either do a sequential scan or use database indexing techniques to perform a fast search over the whole motion capture database to find the best stitching motions.

B. Main Idea

The main contribution of our work is that we propose an intuitive, first-principles approach, by computing the effort that is needed to do the transition (laziness-effort, or "L-score"). Our conjecture is that, the smaller the effort, the more natural the transition will seem to humans. Moreover, we propose the elastic L-score which allows for elongated stitching, to make a transition as natural as possible. We present preliminary experiments on both artificial and real motions which show that our L-score approach indeed agrees with human intuition, it chooses good stitching points, and generates natural transition paths.

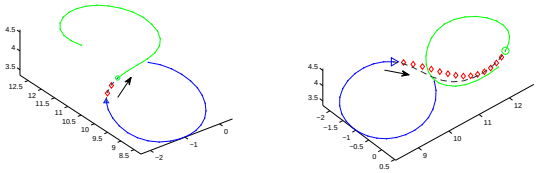
C. Results

We capture a set of waving, walking, running and jumping motions at 30 frames per second. Motions are 300 to 2000 frames in length and have m=93 dimensional joint positions in body local coordinates. We use one Kalman filter (=LDS) for each of the m=93 features, and set the parameters according to physics [2]. We have informally viewed a large variety of transitions within this database and find that our approach consistently performs as well or better than the Euclidean distance metric at generating pleasing transitions.

In order to assess the quality of the stitching found by our *elastic* L-score, we blank out a short interval (2 frames) and a long interval (11 frames) from the transition made by the human actor during 2 waving circle motions, and we compare the actual trajectory against the transition trajectories estimated by the *elastic* L-score. The processing time is around two and a half hours on a Pentium class machine. The observations (see Figure 6) are as follows:

- Our method computes the correct value of blanked-out frames, or gets very close to it.
- Our generated trajectories match very well the actual trajectories.

More results and details are in [2].



(a) (b)

Fig. 6. Real motion stitching: Right-hand coordinates of a human transition motion, with the dashed part blanked out (2 blank-out frames for the left figure, 11 for the right). △/○ marks the take-off/landing frame, respectively. Red ◇ stand for our reconstructed path using *elastic* L-score; notice how close they are to the ground truth (gray dashed line). The *elastic* L-score either finds the correct k_{opt} (=2 in left) or gets very close (=14, vs 11, in right)

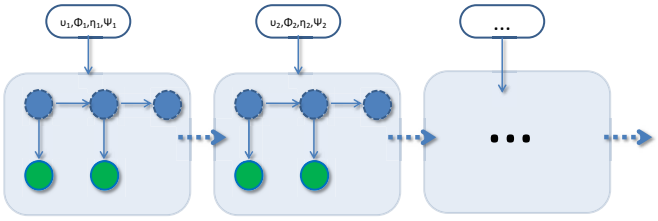
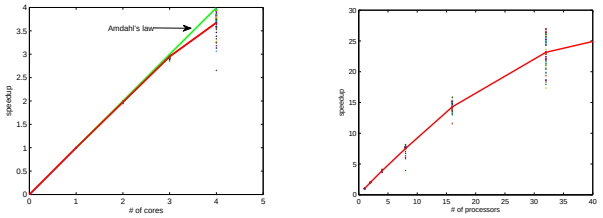


Fig. 7. Graphical illustration of dividing LDS into blocks in the *Cut* step. Note *Cut* introduces additional parameters for each block.



(a) desktop (b) supercomputer

Fig. 8. Average speedup of CAS on a commercial multi-core desktop and a supercomputer, running on 58 motions. The Sequential version is on one processor, identical to the EM algorithm.

IV. Parallelizing on Multicore

A. Problem Definition

In both problems described above, there are estimation or learning steps for Kalman filters or Linear Dynamical Systems involved in the corresponding algorithms. The well known EM algorithm for learning of Linear Dynamical Systems (LDS) iterates between computing conditional expectations of hidden variables through the forward-backward procedure (E-step) and updating model parameters to maximize its likelihood (M-step). Although EM algorithm generally produces good results, the EM iterations may take long to converge. For example, our experimental results show that on a 93-dimensional dataset of length over 300, the EM algorithm would take over one second to compute each iteration and over ten minutes to converge on a high-end multi-core commercial computer. Given multiple co-evolving sequences, our goal is to develop a parallel algorithm to learn LDS parameters, by taking advantage of the quickly developing parallel processing technologies to achieve dramatic speedup.

B. Main Idea

Traditionally, the EM algorithm for LDS running on a multi-core computer only takes up a single core with limited processing power, and the current state-of-the-art dynamic parallelization techniques such as speculative execution benefit little to the straightforward EM algorithm due to the nontrivial data dependencies in LDS.

The basic idea of CAS is to (a) *Cut* both the chain of hidden variables as well as the observed variables into smaller blocks (Figure 7), (b) perform intra-block computation, and (c) *Stitch* the local results seamlessly by summarizing sufficient statistics and updating model parameters and an additional set of block-specific parameters. The algorithm would iterate over 4 steps, where the most time-consuming E-step in EM as well as the two newly introduced steps could be parallelized with little synchronization overhead. Furthermore, this approximation of global models by local sub-models sacrifices only a little accuracy, due to the chain structure of LDS. On the other hand, it yields almost linear speedup.

C. Results

We implement the parallel learning algorithm using OpenMP, and evaluate it on a standard motion capture dataset from CMU mocap database.

Figure 8 shows the speedup on the multi-core desktop (maximum 4 cores) and a supercomputer at NCSA[2]. We also include the theoretical limit from Amdahl's law. With respect to quality. both parallel and serial achieve very small error and are similar to each other. Note the reconstruction by CAS is very close to that by sequential EM algorithm. More discussion is in [3].

V. Conclusion and Future Work

In this paper, we present fast algorithms on mining co-evolving time series, with or without missing values. Our algorithms could mine meaningful patterns effectively and efficiently. With those patterns, our algorithms can do forecasting, compression, and segmentation. Furthermore, we apply our algorithm to solve practical problems including occlusions in motion capture, and generating natural human motions by stitching low-effort motions. We also propose a parallel learning algorithm for LDS, which will serve as corner stone of many applications and algorithms for time series.

In the future, we will continue our exploration on the theme of mining large co-evolving sequences, with the goal of developing fast algorithms for finding patterns, summarizing, and detecting anomalies.

References

[1] L. Li, J. McCann, N. Pollard, and C. Faloutsos, "Dynammo: Mining and summarization of coevolving sequences with missing values," in *KDD*. New York, NY, USA: ACM, 2009. [Online]. Available: http://www.cs.cmu.edu/ leili/pubs/li-kdd09.pdf

[2] L. Li, J. McCann, C. Faloutsos, and N. Pollard, "Laziness is a virtue: Motion stitching using effort minimization," in *Short Papers Proceedings of EUROGRAPHICS*, 2008.

[3] L. Li, W. Fu, F. Guo, T. C. Mowry, and C. Faloutsos, "Cut-and-stitch: efficient parallel learning of linear dynamical systems on smps," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2008, pp. 471–479. [Online]. Available: http://www.cs.cmu.edu/ leili/paralearn/li-kdd08.pdf

[2] http://www.ncsa.illinois.edu