Effective Pattern Similarity Match for Multidimensional Sequence Data Sets

Seok-Lyong Lee^{1,*} and Deok-Hwan Kim^{2,**}

¹ School of Industrial and Information Engineering,
Hankuk University of Foreign Studies, 449-701, Korea
sllee@hufs.ac.kr

² School of Electronics and Electrical Engineering,
Inha University, 402-751, Korea
deokhwan@inha.ac.kr

Abstract. In this paper we present an effective pattern similarity match algorithm for multidimensional sequence data sets such as video streams and various analog or digital signals. To approximate a sequence of data points we introduce a *trend vector* that captures the moving trend of the sequence. Using the trend vector, our method is designed to filter out irrelevant sequences from a database and to find similar sequences with respect to a query. Experimental results show that it provides a lower reconstruction error and a higher precision rate compared to existing methods.

Keywords: Pattern match, Multidimensional data sequence, Trend vector.

1 Introduction

A multidimensional data sequence (MDS) is a sequence of data elements, each element being represented as a multidimensional vector. In [5], we have defined an MDS S with k points in d -dimensional space as a sequence of its component vectors, $S = \langle S_0, S_1, \dots, S_{k-1} \rangle$, where each vector $S_i (0 \leq i \leq k-1)$ is composed of d scalar entries, $S_i = (S_i^1, S_i^2, \dots, S_i^d)$. Time-series data, a sequence of *one-dimensional* real numbers, is obtained by replacing S_i with a single scalar value. An MDS is partitioned into segments of equal or varying lengths, each of which is represented as a *trend vector*. It holds the information on the moving trend of points within a segment. The similarity between segments is defined using the trend vector, and the similarity between sequences is defined in terms of the similarity between segments of the sequences. The pattern similarity search problem in this paper is defined as follows: Given a query Q , a set of data sequences SET , and a similarity threshold ζ ($0 \leq \zeta \leq 1$), the goal is to retrieve sequences S 's from SET such that the similarity between S and Q is equal to or greater than ζ .

---

* This work was supported by Korea Research Foundation Grant (KRF-2004-041-D00665).

** Corresponding author.

Recently, Keogh et al. [2] and Yi et al. [9] independently proposed a mean-based method approximating a time sequence by dividing it into equal-length segments and recording a mean value of data points that fall within the segment. In this method a sequence is represented as a feature vector that holds s means when the sequence is divided into s segments. By extending it Keogh et al. [3] introduced a new dimensionality reduction technique called an adaptive piecewise constant approximation (APCA) that approximates each time sequence by multiple segments of varying lengths such that their individual reconstruction errors are minimal. They showed that APCA could be indexed using a multidimensional index structure, and proposed two distance measures, D_{LB} and D_{AE} , to provide a lower bounding Euclidean distance approximation and a non-lower bounding tight approximation, respectively.

However, these methods basically address the similarity search for one-dimensional time-series data and thus do not handle multidimensional data sequences effectively. Furthermore, they represent each segment as a mean, causing the trend information of the segment to be lost. In mean-based approaches [2,3,9], the trend information may be maintained with a small segment size, but it increases the number of mean values, leading to a processing overhead.

2 Representation of a Sequence

The moving trend of a series of data points is usually represented as the slope of a straight line that approximates the points. Consider a segment SEG with a sequence of k points, $\langle P_0, P_1, \dots, P_{k-1} \rangle$, as shown in Fig. 1. To represent the trend of SEG we usually use a *start-end vector* SE . It is however not desirable to choose SE since it causes a large reconstruction error with respect to an original sequence. We may compute an *optimal vector* OP that is parallel to SE and minimize a reconstruction error. But, to avoid a possible overhead to find OP , we use a vector that is parallel to SE and passes through the middle of a mean, called *trend vector* TV . Fig. 1 depicts TV for a one-dimensional time sequence. For convenience, we added a horizontal time-axis and thus the mean is drawn as a line. In multidimensional data space, the mean is represented as a point without the time axis. All data points in mean-based approach is approximated by a single value, *mean*, while in TV -representation points, $P_0, \dots, P_{k-1}$ is approximated by points, $P'_0, \dots, P'_{k-1}$, on TV respectively.

Fig. 1. Approximation of a sequence by mean, start-end vector (SE), optimal vector (OP), and trend vector (TV)

Definition 1: A trend vector TV in d -dimensional space is defined as $TV = \langle k, \mathbf{m}, \boldsymbol{\alpha} \rangle$, where k is the number of points that fall within a segment, $\mathbf{m} = (m_1, m_2, \dots, m_d)$ is a mean point, each element of which is a mean of k points in each dimension, and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_d)$ is a slope of a start-end vector. ■

By replacing $\boldsymbol{\alpha} = \text{NULL}$, this representation is reduced to APCA representation [3], while it is reduced to the segmented mean representation [2,9] by replacing $k = \text{NULL}$ and $\boldsymbol{\alpha} = \text{NULL}$. A trend-vector TV has various desirable properties: (1) It approximates well the moving trend of points in a segment, (2) It achieves a very low reconstruction error with respect to an original sequence, (3) Computing TV is straightforward and thus very fast, and (4) Using it for the similarity search prunes irrelevant sequences effectively. Fig. 2 shows sequences with a mean value approximation (dotted lines) and a trend-vector approximation (directional solid lines) for a one-dimensional sequence. We can observe that a trend-vector provides better approximation than the mean.

Fig. 2. Mean and trend-vector approximation of a sequence

3 Pattern Match Using a Trend Vector

3.1 Similarity Between Segments

We introduce two distance measures between segments. One is d_{hr} considering hyper-rectangles derived from TV to prune irrelevant segments from a database and the other is d_{seg} considering points on TV to refine candidate segments selected by d_{hr} . When segments are indexed and stored into a database for the post retrieval process, an MBR(minimum bounding rectangle) that bounds points in a segment is usually used for indexing since it provides the property of ‘no-false dismissal.’ It however suffers from many ‘false hits,’ and for our method, needs an extra storage to store low and high points of the rectangle. Thus we utilize the hyper-rectangle that

bounds TV for indexing, called *TV-rectangle*. Even though it does not preserve the property of ‘no-false dismissal,’ it reduces the search space, achieving a fast retrieval compared to MBR indexing. There exists a trade-off between the efficiency and the correctness. In reality many applications may not strongly insist on the correctness. The distance d_{hr} and d_{seg} are defined as follows:

Definition 2: Distance $d_{hr}(SEG_1, SEG_2)$ between two segments, SEG_1 and SEG_2 , is defined by the minimum Euclidean distance between two TV-rectangles that bounds TV_1 and TV_2 of SEG_1 and SEG_2 , respectively. That is

$$d_{hr}(SEG_1, SEG_2) = \sqrt{\sum_{j=1}^d u_j^2}, \quad (1)$$

$$\text{where } u_j = \begin{cases} |SEG_1.H_j - SEG_2.L_j| & \text{if } SEG_1.H_j < SEG_2.L_j \\ |SEG_1.L_j - SEG_2.H_j| & \text{if } SEG_2.H_j < SEG_1.L_j \\ 0 & \text{otherwise.} \end{cases}$$

where $SEG.L$ and $SEG.H$ are high and low points of a TV-rectangle of SEG respectively. ■

Observation 3: d_{hr} is shorter than the distance between any pair of points, one on TV_1 and the other on TV_2 . Thus, when TR_1 and TR_2 is sets of points on TV_1 and TV_2 respectively, the following holds:

$$d_{hr}(SEG_1, SEG_2) \leq \min_{P_1' \in TR_1, P_2' \in TR_2} d(P_1', P_2') \quad (2) \quad \blacksquare$$

Definition 4: $d_{seg}(SEG_1, SEG_2)$ between SEG_1 and SEG_2 of equal lengths, is defined as:

$$d_{seg}(SEG_1, SEG_2) = \frac{1}{k} \cdot \sum_{i=0}^{k-1} d(P_{1,i}', P_{2,i}') \quad (3)$$

where $P_{1,i}'$ and $P_{2,i}'$ are i th points on TV_1 and TV_2 , respectively. ■

Computing d_{seg} between different-length segments is straightforward. Since two segments cannot be compared using Equation 3, a shorter segment is compared to a longer one by sliding the former from the start to the end of the latter, in order to make a pair of equal-length comparison units. The shortest of produced distances is selected as the distance between the two segments.

Lemma 5 (Lower Bounding Distance: $d_{hr}(SEG_q, SEG_s) \leq d_{seg}(SEG_q, SEG_s)$): $d_{hr}(SEG_q, SEG_s)$ between two TV-rectangles of a query segment SEG_q and a database segment SEG_s is the lower bound of distance $d_{seg}(SEG_q, SEG_s)$ of the two segments.

Proof: Let SEG_q, SEG_s have k data points, and P_q', P_s' be arbitrary points on trend vectors of SEG_q, SEG_s , respectively. Then:

$$d\_seg(SEG_q, SEG_s) = \frac{1}{k} \cdot \sum_{i=0}^{k-1} d(P_{q,i}', P_{s,i}') \geq \min_{P_q' \in TR_q, P_s' \in TR_s} d(P_q', P_s')$$

By Observation 3, the following holds:

$$d\_hr(SEG_q, SEG_s) \leq \min_{P_q' \in TR_q, P_s' \in TR_s} d(P_q', P_s')$$

Therefore, we conclude: $d\_hr(SEG_q, SEG_s) \leq d\_seg(SEG_q, SEG_s)$. ■

It is also straightforward to extend Lemma 5 for different-length segments. It implies we may use d_hr for index search *safely*, i.e., *without false-dismissal* with respect to d_seg , to prune irrelevant segments from a database. A set of segments obtained by d_hr is a superset of that obtained by d_seg . Since a data space is normalized in a $[0,1]^d$ hyper-cube, the maximum allowable distance is $\sqrt{d}$, a diagonal of the cube. Then the similarity $sim\_seg(SEG_1, SEG_2)$ between two segments SEG_1 and SEG_2 is obtained as follows:

$$sim\_seg(SEG_1, SEG_2) = 1 - \frac{1}{\sqrt{d}} \cdot d\_seg(SEG_1, SEG_2) \quad (4)$$

Accordingly a user-provided similarity threshold ζ can be transformed into a distance threshold ε for index search using a spatial access method and *vice versa*. That is, $\zeta = 1 - \varepsilon / \sqrt{d}$ and $\varepsilon = \sqrt{d} (1 - \zeta)$.

3.2 Similarity Between Sequences and Pattern Similarity Match Algorithm

A sequence is partitioned into equal- or varying-length segments¹⁾. Consider two equal-length sequences S_1 and S_2 , each of which is partitioned into varying-length segments. To compute $sim\_seq(S_1, S_2)$ between S_1 and S_2 , we first divide a whole time interval into sub-intervals with respect to segment boundaries of both sequences. And next, we compute sim_seg 's for the sub-intervals and merge them. A following example shows this.

Example 6: Consider two sequences S_1 and S_2 with 2 and 3 segments respectively as shown in Fig. 3: $S_1 = (SEG_{1,1}, SEG_{1,2})$ and $S_2 = (SEG_{2,1}, SEG_{2,2}, SEG_{2,3})$, where $SEG_{1,1} = \langle k_{1,1}, m_{1,1}, \alpha_{1,1} \rangle$, $SEG_{1,2} = \langle k_{1,2}, m_{1,2}, \alpha_{1,2} \rangle$, $SEG_{2,1} = \langle k_{2,1}, m_{2,1}, \alpha_{2,1} \rangle$, $SEG_{2,2} = \langle k_{2,2}, m_{2,2}, \alpha_{2,2} \rangle$, $SEG_{2,3} = \langle k_{2,3}, m_{2,3}, \alpha_{2,3} \rangle$. Assume that $k_{1,1} = 18$, $k_{1,2} = 14$, $k_{2,1} = 10$, $k_{2,2} = 14$, and $k_{2,3} = 8$. This segmentation yields 4 sub-intervals, I_1, I_2, I_3 , and I_4 . To compare

¹⁾ For the varying-length segmentation, we use the technique that has been proposed in [6], in which the segmentation is based on geometric and semantic properties of a sequence. There are various optimal piecewise polynomial representations [1,7], but they need a considerable overhead. Meanwhile greedy approaches [4,6,8] are more efficient and thus realistic. Due to space limitation, we do not describe a segmentation algorithm in detail. Interested readers are referred to [6] for details.

segments we need to split $SEG_{1,1}$ to $SEG_{1,1,I_1}$ and $SEG_{1,1,I_2}$ in I_1 and I_2 , $SEG_{1,2}$ to $SEG_{1,2,I_3}$ and $SEG_{1,2,I_4}$ in I_3 and I_4 , and $SEG_{2,2}$ to $SEG_{2,2,I_2}$ and $SEG_{2,2,I_3}$ in I_2 and I_3 , respectively. Then we get $k_{1,1,I_1} = k_{2,1} = 10$, $k_{1,1,I_2} = k_{2,2,I_2} = 8$, $k_{1,2,I_3} = k_{2,2,I_3} = 6$, and $k_{1,2,I_4} = k_{2,3} = 8$. By splitting segments we can get similarity values at each interval and compute $sim\_seq(S_1, S_2)$ as follows:

$$sim\_seq(S_1, S_2) = [10 \cdot sim\_seg(SEG_{1,1,I_1}, SEG_{2,1}) + 8 \cdot sim\_seg(SEG_{1,1,I_2}, SEG_{2,2,I_2}) + 6 \cdot sim\_seg(SEG_{1,2,I_3}, SEG_{2,2,I_3}) + 8 \cdot sim\_seg(SEG_{1,2,I_4}, SEG_{2,3})] / 32 \quad \blacksquare$$

Fig. 3. Similarity between two sequences in each sub-interval

Consider $SEG = \langle k, \mathbf{m}, \boldsymbol{\alpha} \rangle$ to be split to r sub-segments, $SEG_1 = \langle k_1, \mathbf{m}_1, \boldsymbol{\alpha}_1 \rangle, \dots, SEG_r = \langle k_r, \mathbf{m}_r, \boldsymbol{\alpha}_r \rangle$. We can approximate $SEG_l (1 \leq l \leq r)$ using a part of TV that falls in the sub-interval of SEG_l . Thus we obtain $k = \sum_{1 \leq l \leq r} k_l$, $\boldsymbol{\alpha} = \boldsymbol{\alpha}_1 = \dots = \boldsymbol{\alpha}_r$, and $\mathbf{m}_l = (m_{l,1}, m_{l,2}, \dots, m_{l,d})$ in d -dimensional space by a simple computation. When two sequences S_1, S_2 have k, l segments respectively, the number of sub-intervals T is: $T \leq k+l-1$ where the inequality occurs when some segment boundaries of two sequences are identical. Thus the similarity $sim\_seq(S_1, S_2)$ is as follows:

$$sim\_seq(S_1, S_2) = \frac{1}{\sum_{t=1}^T k_{I_t}} \cdot \sum_{t=1}^T (k_{I_t} \cdot sim\_seg(SEG_{1,I_t}, SEG_{2,I_t})) \quad (5)$$

where $k_{I_t} = k_{1,I_t} = k_{2,I_t}$, and SEG_{1,I_t} and SEG_{2,I_t} are the segments of S_1 and S_2 in I_t , respectively. We can use a measure sim_seq safely to prune irrelevant sequences from a database with respect to a query, in the support of the following lemma:

Lemma 7: If $sim\_seq(Q, S) \geq \zeta$, then there exists at least a pair of segments (SEG_q, SEG_s) such that $sim\_seg(SEG_q, SEG_s) \geq \zeta$, for $SEG_q \in Q$ and $SEG_s \in S$.

Proof: (By contradiction) Let sequences Q, S have k, l segments respectively, i.e., $Q = (SEG_1, \dots, SEG_k)$, $S = (SEG_1, \dots, SEG_l)$. Without loss of generality, we assume $k \leq l$, since we can switch Q and S if $k > l$. The number of intervals T generated from sequences Q and S is $T \leq k+l-1$. Let us assume that when $sim\_seq(Q, S) \geq \zeta$ there does not exist any pair of segments (SEG_q, SEG_s) such that $sim\_seg(SEG_q, SEG_s) \geq \zeta$ for $1 \leq q \leq k, 1 \leq s \leq l$. Then,

$$\begin{aligned}
sim\_seq(Q, S) &= \frac{1}{\sum_{i=1}^T k_{I_i}} \cdot \sum_{i=1}^T (k_{I_i} \cdot sim\_seg(SEG_{q, I_i}, SEG_{s, I_i})) \\
&< \frac{1}{\sum_{i=1}^T k_{I_i}} \cdot (k_{I_1} \cdot \zeta + k_{I_2} \cdot \zeta + \dots + k_{I_t} \cdot \zeta) = \zeta
\end{aligned}$$

The assumption causes $sim\_seq(Q, S) < \zeta$ that is a contradiction. Thus, Lemma 7 holds. ■

Lemma 7 indicates that sim_seg is safe with respect to sim_seq , i.e., a set of candidate sequences obtained by sim_seg is a superset of that obtained by sim_seq . In this paper we present a similarity measure for equal-length sequences. The similarity for different-length sequences can be evaluated such that a shorter sequence is compared to a longer one by sliding the shorter one from the start to the end of the longer one.

Algorithm Pattern_Similarity_Match

```

SETcand ← ∅      /* a set of candidate sequences */
SETans ← ∅      /* a set of answer sequences */
/* i is for counting segments of a query sequence Q. */
/* j is for counting segments of a data sequence S. */
/* k is for counting data sequences in a database. */

Step 0: /* Pre processing */
    Partition data sequences S into one or more segments
    Extract a trend vector from each segment
    Index trend vectors and store them into a database

Step 1: /* Segmentation of a query sequence */
    Partition a query sequence Q into one or more SEGQ,i
    Extract a trend vector from each query segment SEGQ,i

Step 2: /* 1st pruning by d_hr (index search) and sim_seg */
    For each SEGQ,i of a query sequence Q
        Search an index based on the distance d_hr to find candidate segments SEGk,j
        For each candidate segment SEGk,j
            if (sim_seg(SEGQ,i, SEGk,j) ≥ ζ) then
                SETcand ← SETcand ∪ {Sk}

Step 3: /* 2nd pruning by the similarity measure sim_seq */
    For each selected sequence Sk in the set SETcand
        if (sim_seq(Q, Sk) ≥ ζ) then
            SETans ← SETans ∪ {Sk}

Step 4: Return SETans

```

Fig. 4. Pattern similarity match algorithm

Fig. 4 shows a pattern similarity match algorithm. First, a query sequence is partitioned into segments from which trend vectors are extracted. Next, for each segment of the query sequence, an index is searched to find candidate segments by d_hr , with respect to a threshold ε that is derived from a user-provided similarity threshold ζ . Those candidate segments are evaluated again by using sim_seg , to prune

irrelevant segments further. Then, the sequences in which final candidate segments are contained will be candidate sequences. These sequences are evaluated with respect to the similarity measure between sequences, sim_seq , to determine a final set of answer sequences.

4 Experimental Evaluation

For experiments we generate 2500 MDS's from video streams by extracting colour features from pixels of each frame and averaging them. Each frame is mapped to a 3-dimensional point in the $[0, 1]^3$ unit cube and each sequence contains 64 to 1024 points. For each test, we have issued randomly selected queries from data sequences and taken the average of query results. To evaluate our method we compared the following: SS(sequential scanning method), MB-Diff(mean-based method for varying-length segments [3]), and TV-Diff(our method for varying-length segments).

To evaluate the effectiveness, we used the precision and recall that are well known in similarity search applications. We set the ground truth as a set of retrieved sequences by SS. That is, all sequences retrieved by it are regarded as *relevant* sequences. The experiment has been done with similarity threshold values, ranging from 0.60 to 0.95. This range is considered to be appropriate since two sequences with values below 0.60 are turned out to be 'different' in their pattern shapes in our experiment. From the experiment we observed that our method provides fairly better precisions (up to 1.9 times) than mean-based methods while they show almost the same recall rate (0.98-1.00), which implies that our method prunes more irrelevant sequences than other methods.

Fig. 5. Efficiency for SS, MB-Diff, and TV-Diff

Fig. 6. Reconstruction error for MB-Diff and TV-Diff

Fig. 5 and Fig. 6 show the response time in second for various threshold values and the reconstruction error. As we can see TV-Diff performs better than MB-Diff in all threshold ranges (1.34-1.69 times faster). The high precision causes more irrelevant sequences to be pruned, resulting in the less processing time. Meanwhile, the reconstruction error of TV-Diff is lower (59-74%) than that of MB-Diff, which indicates that a trend vector provides better approximation.

5 Conclusions

We addressed the problem of searching similar patterns for multidimensional sequences such as video streams. To solve the problem we represented each segment of a sequence as a trend vector that encapsulates the moving trend of points in the segment. Based on it we have defined similarity measures between segments and between sequences. Using the measures, our method prunes irrelevant sequences from a database with respect to a given query. The trend vector has the competitive strength since it provides better approximation and faster processing time than mean-based methods as shown in the experiment. Potential applications that are emphasized in this paper are the similarity search on video streams, but we believe other application areas can also benefit. As the future work, we plan to study on applying the proposed method to specialized application domains considering their own characteristics, such as voice signal matching and region-based image search.

References

1. C. Faloutsos, H.V. Jagadish, A. Mendelzon, and T. Milo. A signature technique for similarity-based queries. *SEQUENCES*, Italy, (1997).
2. E. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, (2000).
3. E. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *Proc. of ACM SIGMOD*, pages 151-162, (2001).
4. E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. *Proc. of Int'l Conference of Knowledge Discovery and Data Mining*, pages 20-24, (1997).
5. S. L. Lee, S. J. Chun, D. H. Kim, J. H. Lee, and C. W. Chung. Similarity search for multidimensional data sequences. *Proc. of IEEE ICDE*, pages 599-608, (2000).
6. S. L. Lee and C. W. Chung. Hyper-rectangle based segmentation and clustering of large video data sets. *Information Science*, Vol.141, No.1-2, pages 139-168, (2002).
7. T. Pavlidis. Waveform segmentation through functional approximation. *IEEE Transactions on Computers*, Vol. C-22, No. 7, (1976).
8. C. Wang and S. Wang. Supporting content-based searches on time series via approximation. *Int'l Conference on Scientific and Statistical Database Management*, (2000).
9. B. K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary L_p norms. *Proc. of Int'l Conference on VLDB*, pages 385-394, (2000).