International Journal of Emerging Technology and Advanced Engineering

cSPADE -UE: Algorithm for Sequence Mining for Unstructured Elements Using Time Gap Constraints

Pinkal Shah¹, Prof. (Dr.) A.K.Dua²
¹M.Tech, ²HOD, Computer Science, ASET, Amity University Rajasthan, Jaipur, India

Abstract--We present a new state machine that combines two techniques for complex data sequences: Data modeling and frequent sequence mining. This algorithm relies on unstructured variable gap sequence miner, to mine frequent patterns with different gap between elements. Here we will have two variations: Sequence pruning technique for other primary frequent sequences to reduce space complexity and allow creating same sequence to form even if they do not have matching on all the positions. We apply algorithm to task of protein sequence classification on real data from protein families. A state of the art method for protein classification, by decreasing the state space complexity and improving the accuracy.

Keywords – Data Mining, Data modeling, Frequent sequence for unstructured elements.

I. INTRODUCTION

Many real world applications, such as those in bioinformatics, web accesses, and text mining, finding sequential or temporary data with long range or short range dependencies. Techniques for analyzing such data that can be grouped in different categories like Pattern mining or data modeling. Few methods have been proposed for sequence pattern mining in both data mining [Srikant & Agrawal, 1996; Zaki 2001:]. For sequence data modeling have been widely used in a large scale of application as speech recognition, web surfing site or usage analysis [Pitkow and Pirolli 1999; Durbin et al 1998].

A simple, Markov model is stochastic process where each state depends on its only previous state. In regular models, each states produces only one symbol, and only the transition probabilities have to be estimated. A first order Hidden- Markov-Model, is actually has “Hidden” states. i.e. it may produce one or more symbols, according to the state-dependent output probabilities, which become additional parameters that have to be estimated. Due to their very basic nature: Hidden State: model may not capture longer range sequences. For this kind of problems many models [Schwardt 2000; Galassi 2007] were proposed. However, building an m-order model requires estimation of the joint probabilities of the previous m states. Further, not all earlier states may be predictive on based on current state. Hence.

The training process is high state-space complexity, reduction coverage and moreover its wrong or less range prediction accuracy.

To address this limitation, we introduce a new technique for temporal and sequential data analysis that combines temporary pattern mining and data modeling. The main intension behind this approach is many real world applications are not able to model via Markov model due to complex embedded dependencies. So we introduced model (for Un-structured Elements) to discover and interpret king and short range data [1]. First step of our method uses a new sequence mining algorithm, called Variable-Gap Sequence miner (VGS), to mine variable-length frequent patterns that may contain different gaps between the elements. The second step of our technique uses the mined variable-gap sequences to automatically HMM for Unstructured Elements, i.e. the topology of the model is learned directly from the mined sequence multiple higher order HMMs via a single variable-order model with explicit duration.

II. RELATED WORK

A. Sequence Mining

Sequence mining finds or generates frequent sequential patterns across time or positions in a given data set. Most of the models are designed to generate continuous subsequences, such as those that depend on suffix. Within data mining, the problem of mining sequential patterns was introduced in [Agrawal and Srikant 1995]. then extended the approach to include constraints like max/min gaps in [Srikant and Agrawal 1996]. Other sequence mining methods have also been proposed, such as SPADE [Zaki 2001], which was later extended to cSPADE [Zaki 2000] to consider constraints like max/min gaps and sliding windows. PrefixSpan [Pei et al. 2001] that considers gap constraints. Recent approach is called Recursive Data Mining (RDM) approach. approaches, we explicitly mine the variable gap sequences while keeping track of all the intermediate symbols (in the gaps), their frequencies, and the gap frequency distributions. This analysis of the gap is used, later on, for building the state model.
B. Pattern Extraction: Variable-Gap Sequence (VGS) miner

Variable-Gap Sequence miner (VGS) is based on cSPADE [4]. While cSPADE essentially avoids the length and symbol distributions of gaps, VGS is specially designed to extract such patterns within one or more sequences. The key difference is that during mining VGS clearly keeps track of all the inter-mediate followed symbols, their frequency, and the gap frequency distributions, which are used to build our Model. VGS takes as input the maximum gap allowed (\(\text{max\_gap}\)), the maximum sequence length (\(k\)), and the minimum frequency threshold (\(\text{minsup}\)). VGS mines all sequences having up to \(k\) elements, with no more than \(\text{max\_gap}\) gaps between least \(\text{minsup}\) times in the data.

For example, let \(S = PRQSPWRQPSUVXTQ\) be an input sequence over the alphabet \(\Sigma = \{P, \cdots, X\}\), and let \(\text{max\_gap} = 2\), \(\text{minsup} = 2\) and \(k = 2\). VGS first mines the frequent subsequences of length 1, as shown in Table : VGS : 1 – Subsequences.

<table>
<thead>
<tr>
<th>SID</th>
<th>Time (EID)</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>PQ</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>PQ</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>PR</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>PQR</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>Q</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>Q</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>PQ</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>Q</td>
</tr>
</tbody>
</table>

Before we start describing VGS, we will provide definitions of some terms :

- \(k\)-seq: Sequence of length \(k\), i.e. \(k\) elements. For example, \(P\rightarrow Q\) is a \(2\)-seq where \(Q\) occurs after \(P\).
- \(\text{minsup}\): Minimum support is the minimum threshold for the frequency count of sequences.
- \(\text{max\_gap}\): Maximum gap allowed between any two elements of a \(k\)-seq.
- \(F_1\): The set of frequent \(1\)-seq (single items).
- \(F_k\): The set of all \(k\)-seq which frequency is higher than the minimum threshold \(\text{minsup}\) and the gap between their elements is at most of length \(\text{max\_gap}\).
- \(C_k\): The set of candidate \(k\)-seq.

Those symbols that are frequent are extended to consider sequences of length 2, as shown in Table A. For example, \(P\rightarrow Q\) is a frequent sequence with frequency \(\text{freq} = 3\), since it occurs once with gap of length 1 (PRQ) and twice with a gap of length 2 (PWRQ and PXTQ). Thus the gap length distribution of \(P\rightarrow Q\) is 0, 1, 2 as shown, in Table A, under columns \(g = 0\), \(g = 1\), and \(g = 2\), respectively. VGS also records the symbol distribution in the gaps for each frequent sequence. For \(P\rightarrow Q\), VGS will record symbol frequencies as \(R(2)\), \(T(1)\), \(W(1)\), \(X(1)\), based on the three occurrences. Since \(k = 2\), VGS would stop after mining sequences of length 2. Otherwise, VGS would continue mining sequences of length \(k \geq 3\), until all sequences with \(k\) elements are mined.
The key features of our approach are:

1. We use Vertical id-list database format of SPADE, where sequence is associated with list of objects in which it occurs and its time stamp.
2. We use lattice approach separate original process into sub smaller process so it can be running independently, this will reduce IO cost, since we require very less database scan.

Main reason behind cost efficient is we reduce database scans by using depth first search [2]. a vertical database format, where id-list of each item in dataset. Each entry in id-list is a (sid, eid) pair. Eid deals with the item exist in sequence which id is sid. Sid is the sequence id in the dataset and eid is event id where item exists.

In earlier Example we have 9 different items {P,Q,R,S,T,U,V,W,X}. The id-list is shown as in Table B. By joining items it will help to check frequency of items. By this example, as we join Q and P then it will be Q ∨ P = { Q → P ,P → Q }; this will give maximum subsequences formed in data set by P and Q (with maximum gap length).

These are main steps:

\( F_1 \): we compute all unique items in data set and frequencies compare to \( \text{minsup} \). This can be done by scanning it-list incrementing the support for each \( \text{sid} \). It’s different from SPADE where only \( \text{sid} \) will be considered to take patterns around them.

\( F_2 \): we compute 2-seq with gap of \( g \) between elements. \( g \in \{0, \ldots, \text{max\_gap}\} \), here \( g = 0 \) relates to no elements between two main elements of our sequence. \( g = 1 \) relates to 1 element between sequence and so on. It can be done by scanning id-list order wise. We for each pair we have input sequence whose id is \( \text{sid} \). we raise the frequency if difference between two \( \text{eid} \) events less than \( \text{max\_gap} \) allowed.

Enumeration of all frequent sequences, with freq-uncy at least \( \text{minsup} \), with variable gap as depth first search with each class. For Example, the 3-seq \( P \rightarrow Q \rightarrow R \) has gap \( g_1 \in \{0, \ldots, \text{max\_gap}\} \), between \( P \) and \( Q \) and gap \( g_2 \in \{0, \ldots, \text{max\_gap}\} \) between \( Q \) and \( R \). where \( "P \rightarrow Q" \) means \( P \) followed by \( Q \) after \( g_1 \) elements between them and \( "Q \rightarrow R" \) means \( Q \) followed by \( R \) after \( g_2 \) elements between them as described in algorithm.

The input to the procedure is a set of items of sub lattice \( S \), along with their id-lists and the \( \text{minsup} \) and \( \text{max\_gap} \). The sequence suppose to be frequent from atoms proceed to next level.

### III. Algorithm

1) Procedure cSPADE (min_sup, maxgap)

2) \( P = \{ \text{Parent Class} P_i \} \);

3) For each parent class \( P_i \in P \) do

4) Enumerate-Frequent-Seq \( (P_i, \text{max\_gap}) \);

5) End for

6) End procedure

7) Function Enumerated-Frequent-Seq \( (S, \text{minsup}, \text{max\_gap}) \)

8) For all sequences \( v_i \in S \) do

9) \( T_i = \emptyset \);

10) For all items \( v_j \in S \) with \( j \geq i \) do

11) \( \text{R=} \text{new candidate sequence from } v_i \text{ and } v_j \);

12) \( L(R) = L(v_i) \cap L(v_j) \); with \( 0 \leq (v_i \text{ (eid)}) - v_i \text{ (eid)} \leq \text{maxgap} \), where \( v_i \text{ (eid)} \) is the event id for \( v \)

---

**TABLE B:** Is-list for item A

<table>
<thead>
<tr>
<th>SID</th>
<th>EID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
</tr>
</tbody>
</table>

But in our method we find patterns around sequences and with sequences.

**TABLE A:** VGS : with \( k = 2 \), \( \text{minsup} = 2 \) and \( \text{max\_gap} = 2 \)

<table>
<thead>
<tr>
<th>subsequence</th>
<th>freq</th>
<th>( g = 0 )</th>
<th>( g = 1 )</th>
<th>( g = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P → Q</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Q → P</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

---

594
If freq(R) ≥ minsup then

14 T_i = T_i U { R } ;

15 F_R = F_R U { R } ;

16 End if

17 End for

18 Enumerate – Frequent – Seq (T_i);

19 End for

20 Delete S;

21 End function

C. SUMMARY

In this paper we have introduced a new algorithm Variable Gap Sequence Miner to find frequent patterns to find different gaps and lengths between elements across and within sequences. It’s based on cSPADE [2], a method to find sequences on constraints based. While cSPADE ignores distribution of gaps and length. VGS is designed to extract more sequences from our generated sequences. It explicitly keeps track of all the intermediate symbols, their frequency and gap. Its main part for extracting patterns.

IV. EXPERIMENTAL EVALUATION

The Scope data set is divided into four hierarchical levels: Superfamily, Class, Fold and Family. For SCOP 1.61 (from 2002), the 44327 protein domains were classified into 701 folds, resulting in an average of 64 domains per fold. The number of domains per fold varies in SCOP, where some of the folds, such as T IM barrels, are highly populated, while some of the folds, such as the HSP 40/DnaJ peptide-binding fold that only contain one protein, contain a few examples. Therefore, the SCOP is an imbalanced data set. This imbalanced proportion of examples in each fold contributes to the poor performance of classical machine learning techniques such as support vector machines and neural networks [4]. When learning from such data sets, existing machine learning approaches tend to produce a strong discriminatory classifier or "high accuracy" with very low sensitivity or completeness. We used 10 superfamilies from the SCOP data set (ftp://ftp.rcsb.org/pub/pdb/derived_data/pdb_seqres.txt) namely, family 49417, 46458, 46626, 46689, 46997, 47095, 47113, 48508, 69118, and 81296. We will refer to them as SF1 to SF10 respectively. Each family has 10 sequences. We divided each family data set into 0.9 (9 sequences for each family) for training and 0.1 for testing (1 for each family to a total of 10 sequences).

Here, state space complexity of the two methods. The highest number of states in VOGUE is having, N=402, where as minsup = 18 and max_gap = 9. And where our algorithm is having highest state = 400, minsup = 30 and max_gap = 9. And it follow the fact that higher minsup leads to less but accurate frequent sequences.

We have built a machine for data and Fig B shows result of two models over dataset. With different max_gap.

V. CONCLUSION

Here we have used two types of technique one we have used variable-Gap sequence mining (VGS), to mine frequent patterns and second step to mine sequences to build accurate sequences using min gap between two important items. We have created extra step for mined sequences and to find elements between our sequences if their length is more than specified length and gap between items are more than max_gap then that sequence will be pruned.
We have reduced some of the states which will help to reduce space complexity and improve performance of the algorithm. As shown in Fig. A performance of the algorithms and result of improvement.

REFERENCES