Learning of Temporal Sequences of Behaviour for Artificial Creature.
Declaration

I declare that I worked on the submitted Bachelor Thesis by myself and that I quoted all the used information sources according to the Methodological instruction about compliance of ethical principals during the preparation of the theses.

In Prague 24.5.2013

[Signature]
Acknowledgement

I want to thank Doc. Pavel Nahodil for the possibility of working on this interesting project. He has been extremely supportive during the entire time and helped me to improve the quality of my thesis by many advices. I also thank to Ing. Jaroslav Vítků who was willing to share a professional advice no matter what time I needed it and helped me with finding of the materials for studying.
Abstrakt

Moje bakalářská práce se zabývá návrhem a implementací algoritmů pro on-line učení a generování sekvenčních vzorů primárně získávaných z okolitého prostředí vnímaného umělou bytostí. Následně tyto algoritmy zkoumá z hlediska použitelnosti na spracování rozsáhlých množin dat a predikci událostí pozorovaných autonomním agentem ve virtuálním prostředí. Implementované algoritmy jsou porovnány se současnými algoritmy, které jsou aplikovatelné na podobnou problematiku. Tato práce je součástí dlouholetého výzkumu v oblasti umělého života, vedeného na katedře kybernetiky ČVUT v Praze od roku 2000 docentem Nahodilem a nyní i Ing. Jaroslavem Vítku, jeho doktorandem.

Hlavní úlohou mojí práce je vytvoření nezávislého subsystému schopného zpracování příchozích informací a rozpoznávání sekvenčních vzorů v nich. Tyto vzory získané z minulosti jsou následně používané na predikci událostí. Výsledný program napsaný v programovacím jazyce Java je součástí knihovny pro on-line vytěžování a predikci sekvencí, kterou jsem vyvinul. Kromě této knihovny jsem naprogramoval i modul kompatibilní s robotickým operačním systémem s použitím ROS Java knihovny. Tento modul je spustitelný jako proces zvaný ROS uzel se schopností komunikovat s jinými procesy, které implementují API ROS uzlu. Tímto způsobem ho lze připojit do hybridní umělé neuronové sítě.
Abstract

My bachelor thesis studies, develops and implements algorithms for on-line learning and generating of sequential patterns primary gathered from the surrounding environment perceived by the artificial creature. Consequently it analyzes the usability of these algorithms on processing of the large data sets and on the prediction of events observed by the autonomous agent in the virtual environment. Implemented algorithms are compared to the existing algorithms, which are applicable for the similar problem. This thesis deepens a longtime research in the domain of the Artificial Life, conducted at the department of cybernetics of CTU in Prague since the year 2000 by associate professor Pavel Nahodil and now also Ing. Jarslova Vítků, his PhD student.

The main objective of my work is to develop independent subsystem capable of processing incoming information and recognizing the sequential patterns behind it. These patterns are consequently used to predict upcoming events by depending on information gathered in the past. The resulting program coded in Java programming language is part of the library for on-line sequential mining and prediction developed by me. In addition I programed module compatible with the robotic operating system using ROS Java library. It can be hosted as a process called ROS node with the ability of exchanging information with other processes implementing ROS node API. In this way it can be connected into the hybrid artificial neural network.
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Pavol Svec
Study programme: Ocean Information
Specialisation: Computer and Information Science

Title of Bachelor Project: Learning of Temporal Sequences of Behaviour for Artificial Creatures

Guidelines:

1. Learn basic types of learning and generating sequential patterns.
2. The most appropriate type of learning/sequencing of temporal sequences implement.
   The resulting subsystem will be compatible with Robotic Operating System (ROS).
3. Compare suitability of implemented methods for a given problem, and also the generality
   of the method.
4. Test the benefits of sequential learning in a simple architecture of an autonomous agent
   in the selected virtual environment simulator.

Bibliography/Sources: Will be provided by the supervisor

Bachelor Project Supervisor: doc. Ing. Pavel Nekoda, CSc.

Valid until: the end of the winter semester of academic year 2013/2014

signature

In Prague 24.5.2013
ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Pavol Štěrbaš

Studijní program: Odpovídá informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Učení časových sekvenčních chování pro umělé tvory

Příkazy pro vypracování:

1. Zabyde se základními typy učení a generování sekvenčních vzorů.
2. Nejvýhodnější typy učení/generování sekvencí implementujte tak, aby vysledné subsystémy byly kompatibilní s robotickým operačním systémem ROS
3. Porovnejte výkon implementovaných metod proti zadaný problém a současně obecnost jejich použití
4. Otestujte výkony sekvenčního učení v jednoduché architektuře autonorního agenta (umělé bytosti) ve vybraném simulátoru virtuálního prostředí.

Seznam odborné literatury: Dodá vedoucí práce:


Platnost zadání: do konce zimního semestru 2013/2014

[Signatures]

In Prague 24.5.2013

signature
# Contents

List of Abbreviations ........................................... ix
List of Figures .................................................. x
List of Tables ................................................... xi
List of Algorithms ............................................... xii

1 Introduction ................................................... 1
   1.1 Thesis Outline ............................................ 2

2 Theoretical Foundation and State-of-the-Art ................. 3
   2.1 Pattern Recognition ....................................... 3
       2.1.1 Introduction to Pattern Recognition ............... 3
       2.1.2 Pattern Recognition in Machine Learning ....... 3
   2.2 Unsupervised learning ................................... 5
       2.2.1 Hierarchical Clustering ............................ 6
       2.2.2 On-line Hierarchical Clustering ................. 7
   2.3 Sequence and Association Rule Mining .................. 8
       2.3.1 Association Rule Mining ............................. 8
       2.3.2 Sequential Pattern Mining ......................... 10
       2.3.3 Apriori Algorithm .................................. 10
       2.3.4 Generalized Sequential Pattern Algorithm .... 11
       2.3.5 PrefixSpan an FreeSpan ............................ 11
       2.3.6 Top-K sequential patterns ......................... 12
   2.4 Sequential pattern mining from continuous data stream .. 13
       2.4.1 Sequential Patterns Efficient Extraction in Data streams 14
       2.4.2 Interactive Mining for Weighted Frequent Patterns 16
2.4.3 A Frequent Sequential Data Stream Patterns Miner 17
2.4.4 Top-K Sequences in Data Stream 18

3 Algorithms Proposed and Tested 21
3.1 Requirements for Implemented Algorithms 21
3.1.1 Pattern recognition Algorithm 21
3.1.2 Sequence Learning Algorithm 23

4 Experiments 26
4.0.3 Pattern recognition Experiments on Processing Datasets 26
4.0.4 Sequence Learning Experiments on Processing Datasets 27
4.0.5 Sequence Learning Experiments in the Virtual Environment 30

5 Thesis Conclusion and Future Work 34

Bibliography 37
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROS</td>
<td>Robotic Operation System</td>
</tr>
<tr>
<td>ČVUT</td>
<td>České vysokéučení technické v Praze</td>
</tr>
<tr>
<td>CTU</td>
<td>Czech Technical University in Prague</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>GSP</td>
<td>Generalized Sequential Pattern Algorithm</td>
</tr>
<tr>
<td>SPEED</td>
<td>Sequential Patterns Efficient Extraction in Data streams</td>
</tr>
<tr>
<td>TTW</td>
<td>Titled-Time Windows</td>
</tr>
<tr>
<td>SSM</td>
<td>A Frequent Sequential Data Stream Patterns Miner</td>
</tr>
<tr>
<td>IWFPTWA</td>
<td>Incremental Weighted Frequent Pattern Tree based on the Weight Ascending Order</td>
</tr>
<tr>
<td>IWFPTFD</td>
<td>Incremental weighted frequent pattern tree based on the frequency descending order</td>
</tr>
<tr>
<td>PLWAP</td>
<td>Preorder-like Web Access Pattern Tree</td>
</tr>
<tr>
<td>WAP</td>
<td>Web Access Pattern Tree</td>
</tr>
<tr>
<td>FP</td>
<td>Frequent Pattern</td>
</tr>
<tr>
<td>DS</td>
<td>Data Stream</td>
</tr>
<tr>
<td>SPMF</td>
<td>Sequential Pattern Mining Framework</td>
</tr>
<tr>
<td>UPGMA</td>
<td>Unweightened Pair Group Method</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Pattern recognition process</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>A simple caption</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Sequence mining from data stream</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>Logarithmic Titled-Time Windows</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>The region Lattice after batches processing</td>
<td>15</td>
</tr>
<tr>
<td>2.6</td>
<td>PLWAP tree</td>
<td>18</td>
</tr>
<tr>
<td>2.7</td>
<td>Top-K Sequences in Data Stream algorithm diagram</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>On-line hierarchical clustering algorithm dependence on the initial tree</td>
<td>26</td>
</tr>
<tr>
<td>4.2</td>
<td>Obtained clusters of on-line hierarchical clustering algorithm</td>
<td>27</td>
</tr>
<tr>
<td>4.3</td>
<td>Sequential mining algorithms small data set comparison</td>
<td>28</td>
</tr>
<tr>
<td>4.4</td>
<td>Sequential mining algorithms bigger data set comparison</td>
<td>28</td>
</tr>
<tr>
<td>4.5</td>
<td>Dependence of FP tree size on Success and Memory usage</td>
<td>29</td>
</tr>
<tr>
<td>4.6</td>
<td>Top-K Datastream threshold prediction</td>
<td>30</td>
</tr>
<tr>
<td>4.7</td>
<td>Screen-shot from the Vivae simulator</td>
<td>31</td>
</tr>
<tr>
<td>4.8</td>
<td>Performance of curves prediction</td>
<td>32</td>
</tr>
<tr>
<td>4.9</td>
<td>Dependence of error rate on the batch size</td>
<td>32</td>
</tr>
<tr>
<td>4.10</td>
<td>Comparison to the Top-K Sequential rules</td>
<td>33</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The linkage criteria of agglomerative clustering algorithms</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Example of transaction database</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Example of sequence database</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Database projections of Table 2.3</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>Incoming Batches</td>
<td>15</td>
</tr>
<tr>
<td>2.6</td>
<td>Sequence table</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Comparison of sequential pattern mining algorithms</td>
<td>23</td>
</tr>
</tbody>
</table>
# List of Algorithms

1. Agglomerative clustering algorithm ........................................ 7
2. Sequential hierarchical clustering algorithm ............................. 8
3. Apriori algorithm ............................................................ 11
4. Top-K Sequences algorithm (Top-K support version) .................. 13
5. WAP Tree update algorithm ................................................ 17
6. Algorithm for estimating best sequence size ............................ 24
7. Proposed algorithm for predicting most probable item ................. 25
Chapter 1

Introduction

One of the main goals of many researchers in the present world is to design artificial creature. The autonomous being capable of continuous learning and adapting to the complexity of its surrounding environment. Humans achieve the ability of making intelligent decisions in the complex world using the part of their brain called neocortex. Many scientists believe, that the structure of the neocortex consist of huge amount (over 300 million) of pattern recognizers connected together in the large hierarchy (Kurzweil, 2012) . To evaluate this possibility we could construct similar hierarchy by inter-connecting pattern recognizing nodes.

The aim of this thesis is to design and implement sequential pattern and hierarchical clustering algorithms with emphasis on their future interaction. These subsystems are oriented on on-line processing of continuous data stream with the need of utilizing the results in the real time. They are a part of framework of hybrid neural network systems proposed by Jaroslav Vitku. Implemented algorithms represented as ”neural modules” present in an agent architecture will communicate with other modules connected into a hybrid artificial neural network. They will receive input data of n-dimensional real variables and publish their results to other nodes situated in the higher level of the hierarchy of ANN. This setup of implemented systems allows us to simulate similar processes to those taking place inside human brain. This processes can be simulated and evaluated at different levels of abstraction.
1.1 Thesis Outline

Here is the brief description of this thesis outline.

**Chapter 1** The first chapter provides introduction to the thesis goals and aims of the Bachelor Thesis.

**Chapter 2** The second chapter describes theoretical foundation for the implementation of the algorithms used for the hierarchical clustering and sequence recognition. It also familiarizes reader with the present state-of-the-art for these algorithms.

**Chapter 3** The third chapter introduces the algorithms, which were chosen and implemented.

**Chapter 4** In the fourth chapter, the conducted experiments for evaluating and comparing the implemented algorithms are described.

**Chapter 5** Fifth chapter contains conclusion and compares the predicted and actual goal.
Chapter 2

Theoretical Foundation and State-of-the-Art of Pattern and Sequential Recognition Algorithms

2.1 Pattern Recognition

2.1.1 Introduction to Pattern Recognition

Pattern recognition involves identification of faces, objects, words, melodies, etc. It is one of the most basic ability of every human being. It enables us, to distinguish things from each other and recognize world around us. Furthermore it allows us to learn, how things and events are inter-connected (For example: If the skin is exposed to direct sunlight, it gets burned). Because pattern recognition is such essential ability of any intelligent living creature, it is also important when designing artificial creatures, which act autonomously in any environment, like intelligent creatures. In following part, there will be described, how pattern recognition tasks are represented in machine learning, and how we can solve some of them.

2.1.2 Pattern Recognition in Machine Learning

In machine learning, pattern recognition is the assignment of a label to a given input value. An example of pattern recognition is classification, which attempts to assign each input value to one of a given set of classes (for example, determine whether a given email is "spam" or "non-spam"). Pattern recognition algorithms generally aim to provide a
reasonable answer for all possible inputs and to perform “most likely” matching of the inputs, taking into account their statistical variation. Pattern recognition is generally categorized according to the type of learning procedure used to generate output value (Wikipedia, 2013).

**Supervised learning** is field of the pattern recognition, which assumes that training data (or training set) has been provided. This training set consists of instances, which have been properly labeled with the correct output value. This learning procedure then generates a model that tries to fulfill the objective of performing well on the training set and reducing incorrect decisions on the testing data-set. These two objectives can be conflicting.

**Unsupervised learning**, on the other hand processes unlabeled data and attempts to find inherent patterns to determine the correct output value of individual data instances.

Architecture of autonomous agent designed by Doc. Nahodil and Ing. Vitku, which creates background for this thesis, is built on idea of agent, which is highly adaptable to the surrounding environment. Therefore following sections of the article will be more focused on unsupervised rather than supervised learning.
2.2 Unsupervised learning

Unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns (Dayan, 2001) The aim of the unsupervised learning algorithms is to find inherent patterns and determine the correct relations among received data.
2.2.1 Hierarchical Clustering

Cluster analysis is the specific field which belongs to the unsupervised learning algorithms section. It deals with the task of grouping the objects into the groups called clusters by measuring and evaluating objects similarity. Clustering algorithms without prior knowledge of the number of the clusters seeking to build a hierarchy of clusters are referred to as hierarchical clustering algorithms. The aim of my research is to implement subsystem capable of recognizing patterns among priorly unknown objects with high level of generality. Because hierarchical clustering creates patterns on the top of the raw received data it is suitable for the implementation of the autonomous pattern recognizing node.

In general, hierarchical clustering is divided into the algorithms building the hierarchy in the bottom-up approach. We refer to them as the agglomerative clustering algorithms. In contrast, the algorithms which seek to build the hierarchy among clusters in the top-down way are called divisive clustering algorithms. Agglomerative clustering algorithms use metrics to determine the distance among the data. In addition they determine the distance between clustering by the linkage criteria.

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-linkage</td>
<td>( \max{d(a, b); a \subseteq A, b \subseteq B} )</td>
</tr>
<tr>
<td>Complete-linkage</td>
<td>( \min{d(a, b); a \subseteq A, b \subseteq B} )</td>
</tr>
<tr>
<td>Unweighted Pair Group Method(UPGMA)</td>
<td>( \frac{1}{</td>
</tr>
<tr>
<td>Centroid clustering</td>
<td>( \left(\frac{\sum_{a \subseteq A}</td>
</tr>
</tbody>
</table>

Table 2.1: The linkage criteria of agglomerative clustering algorithms.

(Institute, 2009)

Depending on the linkage criteria different computational complexities arise when processing the data. For the best-merge persistent methods like UPGMA or Single-linkage clustering, the Local Closest Pair method has been introduced (Gronau and Moran, 2007). It is capable of determining the cluster distribution within the \( O(n^2) \) complexity. Other agglomerative clustering algorithms require at least \( O(n^2 \log(n)) \) of computational time.
in order to retrieve the hierarchy of the clusters.

\[
\begin{align*}
C_k & \text{ - candidate item set of size } k; \\
L_k & \text{ - frequent item set of size } k;
\end{align*}
\]

**Input:** \( S \) = dataset

**Output:** \( C \) = set of clusters

\[
C = \text{Create Separate Cluster for each data};
\]

\[
\text{while } \text{getSize}(C_{\text{last}}) < \text{getSize}(S) \text{ do}
\]

\[
\begin{align*}
[C_1, C_2] & \text{ = find clusters with minimal distance;} \\
C_{\text{last}} & \text{ = merge}(C_1, C_2); \\
C & = C \cup C_{\text{last}};
\end{align*}
\]

end

**Algorithm 1:** Agglomerative clustering algorithm.

### 2.2.2 On-line Hierarchical Clustering

Running the hierarchical clustering algorithms to cluster the data arriving in the on-line fashion is computationally too expensive. If we want to meet the time constraint of the real time data processing another approach needs to be proposed. Sequential Hierarchical Pattern Clustering (B. Farran, 2009) starts the process of clustering by building initial dendrogram using Memory-Constrained UPGMA approach.

MC-UPGMA is based on UPGMA but it deals with the memory requirement by firstly sorting the edges and saving them into the file on the hard drive. It then processes the files and creates the bottom-up hierarchy or the tree.

In the second phase, tree is being updated using the incoming data.
**Input**: Root of the initial tree (CurNode), the new pattern (NewNode), and the novelty threshold ($\theta$)

**Output**: Updated hierarchical tree

\[
\text{simdist}_{\text{CN}} \leftarrow \text{similarity} \_\text{distance}(\text{CurNode}, \text{NewNode})
\]

\[
\text{if } \text{simdist}_{\text{CN}} \leq \theta \text{ then}
\]

\[
(\star) \text{ Children } \leftarrow \text{getChildrenOf(CurNode)}
\]

\[
\text{if } \text{Children} == \text{NULL then}
\]

\[
\text{Make NewNode as a sibling of CurNode and update ancestors.}
\]

\[
\text{else}
\]

\[
\{ \text{CurNode has children} \}
\]

\[
\text{nearestNode } \leftarrow \text{min} (\text{similarity} \_\text{distance} (\text{Children}, \text{NewNode}))
\]

\[
\text{if } \text{nearestNode} \leq \text{thresh then}
\]

\[
\text{CurNode } \leq \text{nearestNode}
\]

\[
\text{Go to } (\star)
\]

\[
\text{else}
\]

\[
\text{Make NewNode as sibling of CurNode and update ancestors.}
\]

\[
\text{end}
\]

\[
\text{else}
\]

\[
\text{Make NewNode as sibling of CurNode by creating new root.}
\]

\[
\text{end}
\]

Algorithm 2: Sequential hierarchical clustering algorithm.

*(B. Farran, 2009)*

This tree updating strategy can be modified for other types of linkage criterion and distance measurements. Centroid clustering might also be feasible since in the algorithm we are using mean values determine the cluster of the new data.

### 2.3 Sequence and Association Rule Mining

#### 2.3.1 Association Rule Mining

Association rule learning is method of data mining, used for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness (Piatetsky-Shapiro, 1991).
Given a transaction database, it searches for items, which are interconnected and depend on each other and finds rules that predict the occurrence of an item based on the occurrences of other items in the transaction. Association rules are employed today in many application areas including Web usage mining, intrusion detection, Continuous production and bioinformatics (Wikipedia, 2013).

\[
\text{TID} \quad \text{Items} \\
1 \quad \text{Meat, Butter} \\
2 \quad \text{Meat, Spice, Water, Carrot} \\
3 \quad \text{Butter, Spice, Water, Juice} \\
4 \quad \text{Meat, Butter, Spice, Water} \\
5 \quad \text{Meat, Butter, Spice, Juice}
\]

Table 2.2: Example of transaction database

\[
\begin{align*}
\{\text{Spice}\} & \rightarrow \{\text{Water}\} \\
\{\text{Butter, Meat}\} & \rightarrow \{\text{Carrot, Juice}\} \\
\{\text{Water, Meat}\} & \rightarrow \{\text{Butter}\}
\end{align*}
\]

Association Rules Example:

Let Itemset \( I = \{i_1, i_2, i_3, \ldots, i_n\} \) be set of \( n \) binary attributes called items and Transaction \( T = \{I_1, I_2, I_3, \ldots, I_n\} \) set of \( n \) itemsets. Database \( D = \{t_1, t_2, t_3, \ldots, t_n\} \) is set of transactions called in the database. Then we define Rule as an implication of the form \( X \rightarrow Y; X, Y \subseteq I \land X \cap Y = \emptyset \). The sets of items (for short itemsets) and are called antecedent (left-hand-side or LHS) and consequent (right-hand-side or RHS) of the rule respectively. We define Support as the proportion of transactions in the data set which contain the itemset. In table 2.2 \( \text{supp}(\text{meat}) = 4/5 = 0.8 \).

Confidence of the rule is calculated as \( \text{conf}(X \rightarrow Y) = \text{supp}(X \cup Y) / \text{supp}(X) \) (Wikipedia, 2013).
2.3.2 Sequential Pattern Mining

Sequence mining is a topic of data mining concerned with finding statistically relevant patterns between data examples where the values are delivered in a sequence (Mabroukeh and Ezeife, 2010). Sequence mining unlike association rule mining aims to recognize sequences instead of transactions. Many areas of application of sequential pattern mining include BioInformatics or Web click log analysis in Information Retrieval systems.

<table>
<thead>
<tr>
<th>Sid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a,(b,c),d</td>
</tr>
<tr>
<td>2</td>
<td>a,(c,b,d),e</td>
</tr>
<tr>
<td>3</td>
<td>a,(a,c),b</td>
</tr>
<tr>
<td>4</td>
<td>c,(a,b),d</td>
</tr>
</tbody>
</table>

Mined sequences: (min support = 3)

\{a\}, \{b\}, \{c\}, \{d\} \{a, c\}, \{a, b\}
\{a, d\}, \{c, d\}

Table 2.3: Example of sequence database

Sequence is defined as an ordered list of itemsets (sets of items) \( S = I_1, I_2, I_3, \ldots, I_n \).

We define Database of sequences as \( D = \{s_1, s_2, s_3, \ldots, s_n\} \).

Support of itemset \( X \) is defined as count of sequences in database containing \( X \). Confidence of the rule \( \text{Conf}(X \rightarrow Y) \) is defined as count of sequences in database, in which item \( Y \) occurs after \( X \) divided by count of sequences containing item \( X \). A sequence \( \alpha = < \alpha_1, \alpha_2, \ldots \alpha_n > \) is called a subsequence of another sequence \( \beta = < \beta_1, \beta_2, \ldots \beta_n >, \) and \( \beta \) a supersequence of \( \alpha \), denoted as \( \alpha \subseteq \beta \) if there exist integers \( 1 \leq j_1 < j_2 \ldots < j_n \leq m \) such that \( a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \ldots, a_n \subseteq b_{j_n} \).

2.3.3 Apriori Algorithm

Apriori algorithm is association rule mining algorithm. It uses two parameters: minimum support and minimum confidence to get useful rules from transaction database. Algorithm starts with calculating support of all items in the database. Then it prunes the database, so that no transaction contains infrequent item (item, which occurrence in the database is below the minimum support). It then proceeds with recursively joining events and creating new rules and pruning rules which have small confidence or support.
\( C_k \) - candidate itemset of size \( k \);
\( L_k \) - frequent itemset of size \( k \);

**Input**: Transaction database, \( \text{min\_support} \), \( L_1 = \{\text{frequent items}\} \), \( k = 1 \)

**Output**: Frequent itemsets \( (F) \)

```
while \( L_k \) is not empty do
    for each transaction \( t \) in database do
        \( C_{k+1} \) = candidates generated from \( L_k \);
        increment the count of all candidates in \( C_{k+1} \) that are contained in \( t \);
        \( L_{k+1} \) = candidates in \( C_{k+1} \) with \( \text{min\_support} \);
    end
    \( F = F \cup L_k \);
    \( k++ \);
end
```

*Algorithm 3: Apriori algorithm (R. Agrawal, 1994)*

### 2.3.4 Generalized Sequential Pattern Algorithm

GSP algorithm utilizes the Apriori algorithm approach on the sequential data. It has also two main steps: joining of the sequences and pruning the infrequent ones. The main difference is in the joining of the sequences. Because we have to be careful about the order of the elements during joining step. The main difference between the GSP algorithm and the Apriori algorithm lies in the generation of the candidate sets.

Let us assume that: \((A,B)\) and \((A,C)\) are two frequent 2-sequences. The candidate generation in a usual Apriori-like style would produce following rules: \( \{A \rightarrow B, C \}; A, B \rightarrow C \}; A, C \rightarrow B \}; B, C \rightarrow A \}; B \rightarrow A, C \}; C \rightarrow A, B \}\). Because the GSP algorithm generates candidate generation of the sequences not associations, it will produce the candidate sequential rules \((A,B,C),(A,C,B)\).

### 2.3.5 PrefixSpan an FreeSpan

Apriori algorithm can potentially generate huge amount of candidate sequences. Moreover, it requires multiple scan of database and is ineffective, when generating long sequential patterns. FreeSpan algorithm was designed to overcome these drawbacks of
Apriori algorithm. It first takes all items, which have greater support than minimum support. It then sorts the list according to support in descending order and creates database-projections of the original sequential database. In first item projection—biggest subsequences of all sequences in original database containing only first item are stored. In second-projection subsequences containing first and second item are stored. In similar way it creates other projections. In second step it recursively mines sequential patterns in every projected database.

<table>
<thead>
<tr>
<th>a-projection</th>
<th>b-projection</th>
<th>c-projection</th>
<th>d-projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a,b</td>
<td>a,(b,c)</td>
<td>a,(b,c),d</td>
</tr>
<tr>
<td>a</td>
<td>a,b</td>
<td>a,(c,b)</td>
<td>a,(c,b,d)</td>
</tr>
<tr>
<td>a,a</td>
<td>a,a,b</td>
<td>a,(a,c),b</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>(a,b)</td>
<td>a,(a,b)</td>
<td>c,(a,b),d</td>
</tr>
</tbody>
</table>

Table 2.4: Database projections of Table 2.3

Descending list of frequent items: \{ (a:4),(b:4),(c:3),(d:3) \}

We denote, that sequence \( \beta = \langle s'_1, ..., s'_m \rangle \) is prefix of \( \alpha = \langle s_1, ..., s_n \rangle \) if:
1. \( m \leq n \)
2. \( \forall i \leq m - 1, s'_i = s_i \)
3. \( s'_m \subseteq s_m \)
4. \( s_m - s'_m > s_m \).

The major idea of PrefixSpan is that, instead of projecting sequence databases by considering all the possible occurrences of frequent subsequences, the projection is based only on frequent prefixes because any frequent subsequence can always be found by growing a frequent prefix (Pei and Han, 2001).

2.3.6 Top-K sequential patterns

Algorithm Top-K sequential patterns uses apriori-like approach, but it has different pruning phase. In general two Top-K sequential pattern approaches can be chosen:

1. We choose minimum support and integer K. Then we search for K sequential patterns \( (x_1, x_2, ..., x_n) \subseteq X \) such that:

\[
\forall y \subseteq Y, \forall x \subseteq X; Conf(x) \geq Conf(y); Supp(x) > min\_supp; Y = S - X \quad (2.1)
\]

Where S is set of all subsequences of sequences in sequence database.
2. We choose minimum confidence. Then, we search for K sequential patterns such, that:

\[
\forall y \subseteq Y, \forall x \subseteq X; \text{Supp}(x) \geq \text{Supp}(y); \text{Conf}(x) > \text{min\_confidence}; Y = S - X
\]  

(2.2)

| Seq_i - frequent sequences of size i;  
| Input: Sequence database, min\_confidence, Seq_1 = \{frequent items\}, i = 1, K = number of top sequences to be found  
| Output: Frequent sequences(S)  
| while Seq_i is not empty do  
| \( S = S \cup Seq_i; \)  
| Seq_{k+1} = Generate sequences from Seq_i, which meet the min\_confidence criterion.;  
| for each s in Seq_{i+1} do  
| if size(S) > K then  
| if s < min(S) then  
| remove s from Seq_{i+1};  
| end  
| end  
| i++;  
| end  

*Algorithm 4:* Top-K Sequences algorithm(Top-K support version)  
(Fournier and Tseng, 2011)

### 2.4 Sequential pattern mining from continuous data stream

In this part we will present known algorithms for finding frequent sequences in continuous data stream. In continuous data stream the size of database grows infinitely. After certain big amount of data is received, it would take too long for previously introduced algorithms to finish and retrieve sequential rules. Algorithms, which deal with this problem generally divide data stream into batches. After that they run algorithm on the batch and gather
the rules (sequences), which are then saved into some specific data structure which is used for retrieving the rules (sequences).

Continuous data stream

![Sequence mining from data stream](image)

**Figure 2.3:** Sequence mining from data stream

### 2.4.1 Sequential Patterns Efficient Extraction in Data streams

SPEED algorithm extracts sequential patterns from the data streams. It incrementally maintains the frequent sequential patterns using the specific data structures with the help of the tilted-time windows structures. Because it uses the Logarithmic TTW, it is suitable for the problems requiring the retrieval of the sequential rules depending on time, when sequences had been gathered. Logarithmic TTW store the gathered sequences in the atomic cells called windows. The most recently mined sequences are stored all in one window. As the time passes the sequences are first coupled by 2 batches into the windows, then by 4,8,16,...etc. in the logarithmic manor. If we received N batches from the stream, the logarithmic tilted-time windows table size will be bounded by $2 \times \log(N) + 2$ which makes this windows schema very space-efficient (Raissi C., 2006).

SPEED uses following data-structures. Lattice ($Lattice_{reg}$), each path in the $Lattice_{reg}$ is provided with a region and sequences in a path are ordered according to the inclusion property. So in every region we have sequences S such, that $\forall a, b \subseteq S : a \sqsubset b \lor b \sqsubset a$. Only maximal sequences are kept in $Lattice_{reg}$. Items and sequences are being stored in the two separate tables.
CHAPTER 2. THEORETICAL FOUNDATION AND STATE-OF-THE-ART

Figure 2.4: Logarithmic Titled-Time Windows
(Raissi C., 2006)

\begin{figure}
\centering
\includegraphics[width=\textwidth]{logarithmic_titled_time_windows}
\caption{Logarithmic Titled-Time Windows}
\end{figure}

Table 2.5: Incoming Batches

\begin{tabular}{|c|c|c|}
\hline
Batches & Seq & Items \\
\hline
$B_0^1$ & $S_a$ & 1,2,3,4,5 \\
 & $S_b$ & 8,9 \\
\hline
$B_2^1$ & $S_c$ & 1,2 \\
\hline
$B_2^3$ & $S_d$ & 1,2,3 \\
 & $S_e$ & 1,2,8,9 \\
 & $S_f$ & 2,1 \\
\hline
\end{tabular}

Table 2.6: Sequence table

\begin{tabular}{|c|c|c|}
\hline
Sequences & Size & TTW \\
\hline
$S_a$ & 5 & \{\{t_0 = 1\}\} \\
\hline
$S_b$ & 2 & \{\{t_0 = 1\}, \{t_2 = 1\}\} \\
\hline
$S_c$ & 2 & \{\{t_0 = 1\}, \{t_1 = 1\}, \{t_2 = 2\}\} \\
\hline
$S_d$ & 3 & \{\{t_0 = 1\}, \{t_2 = 1\}\} \\
\hline
$S_e$ & 4 & \{\{t_2 = 1\}\} \\
\hline
$S_f$ & 2 & \{\{t_2 = 1\}\} \\
\hline
\end{tabular}

Figure 2.5: The region Lattice after batches processing
(Raissi C., 2006)

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{lattice}
\caption{The region Lattice after batches processing}
\end{figure}
SPEED Algorithm runs in the two steps:
(1) Insertion of each sequence of the studied batch in the data structure \( Lattice_{reg} \) using the region principle.
(2) The extraction of the maximal subsequences.

In the first step, it iterates through all new sequences from the current batch and inserts the new sequences to the sequences table, items table and the lattice. For every processed sequence, it searches in the sequence table to find the sub-sequence or the super-sequences of the currently processed sequence. It updates TTW for each found sub(super)-sequence. It then inserts the sequence either in the new region in the lattice, if no sub-sequences and super-sequences were found, or it inserts the sequence in a region of found sub(super)-sequence. In the second step, we prune both sequences and the items which are considered infrequent. Frequency of the item is denoted by the support in TTW. When all TTW of the sequence are dropped the entire sequence is dropped from the \( Lattice_{reg} \), items are being dropped in the similar way.

2.4.2 Single-pass Incremental and Interactive Mining for Weighted Frequent Patterns

A weight of an item is a non-negative real number assigned to reflect the importance of the item in the transaction database. The weight of a pattern, \( P_{x_1, x_2, \ldots, x_k} \) and the weighted support are defined as follows:

\[
Weight(P) = \frac{\sum_{q=1}^{\text{length}(P)} \text{Weight}(x_q)}{\text{length}(P)} \quad (2.3)
\]

\[
W\text{support}(P) = Weight(P) \times Support(P) \quad (2.4)
\]

Single-pass incremental and interactive mining for weighted frequent patterns deals with problem of finding frequent weighted transaction patterns in large transaction databases, or continuous transaction streams. The main problem of the weighted transaction patterns is, that the weighted frequency of the pattern does not have the downward closure property. For example, it can happen, that:

\( W\text{support}(t = (a)) < W\text{support}(t = (a, d)) \). Two tree structures are proposed: \( IWFPT_{WA} \) (Incremental weighted frequent pattern tree based on the weight ascending order) and the \( IWFPT_{FD} \) (Incremental weighted frequent pattern tree based on the
frequency descending order). Both of these tree structures can be used for the maintaining and mining of the weighted frequent sequential patterns. $IWFP_{WT}$ and the $IWFP_{FD}$ maintain the downward closure property by multiplying each pattern’s support by the local maximum weight. The patterns are being mined while the sub-tree containing pattern with $W_{support} > \text{min}_W_{support}$ exists (Ch. F. Ahmed, 2012).

### 2.4.3 A Frequent Sequential Data Stream Patterns Miner

SSM algorithm uses three main data structures to mine the frequent sequential patterns. D-list is a list structure containing supports of all items, which have been processed from the batches. It is sorted in descending order (head element is element with highest support). Algorithm uses the PLWAP (Preorder-like Web access pattern) tree to mine frequent sequences. PLWAP is based on the WAP tree, which is Apriori-like algorithm. Every sequence from the batch is inserted into the WAP tree in the prefix way.

| Input: WAP Tree, Sequence S |
| Output: Updated WAP Tree |

CurrentNode = getRoot(Tree);

for ($i = 1; i < \text{Length}(S); i++)$ do

  Child = Find child of CurrentNode with label of($S[i]$);

  if (child not found) then

    Child = create child with label of($S[i]$);

  end

  CurrentNode = Child;

end

**Algorithm 5:** WAP Tree update algorithm

When mining from the WAP tree, we recursively construct post-fix trees from the original tree using the header-links and then mine frequencies in the constructed trees. Pre-Order linked WAP tree algorithm, is a version of the WAP tree algorithm that assigns unique binary position code to the each tree node and performs the header node linkages pre-order fashion (root, left, right). Both the pre-order linkage and the binary position codes enable the PLWAP to directly mine the sequential patterns from the one
initial WAP tree starting with prefix sequence, without re-constructing the intermediate WAP trees (C. I. Ezeife, 2005).

The last data-structure called FP -Tree is used to store overall frequent sequential patterns. Inserting of frequent sequential patterns is done in similar way like in the PLWAP tree, except for header links which are not kept in the memory. The frequent patterns are then gathered from the FP -Tree (C. I. Ezeife, 2007).

![PLWAP tree](image)

**Figure 2.6: PLWAP tree**
PLWAP tree constructed from the sequences in the table 2.3

### 2.4.4 Top-K Sequences in Data Stream

Top-K Sequences in Data Stream utilizes the idea of the Top-K Sequences and applies it efficiently on mining sequences from continuous data stream. The algorithm receives sequences in the batches. It then finds acceptable support threshold for finding the frequent patterns. It mines data and stores them together with the real threshold.
Top-K Sequences in Data Stream builds on top of the SSM algorithm in the way of mining the sequential rules. Except for it takes as input parameter $K$ which denotes the number of the patterns to be found. To find $K$ best sequential patterns in the Apriori-like approach, we would have to explore too many unnecessary sequences. Top-K Sequences in Data Stream deals with this problem by estimating minimum threshold for expanding current sequence (Dai, 2010). Algorithm iteratively updates the threshold table to store minimum threshold value of each batch that can extract at least $K$ sequential patterns as the information which is used to estimate the threshold of the next batch. To prevent the size of the threshold table to increase infinitely, sliding window structure is adopted with a constant size to construct the threshold table. First, Hoeffding bound (Hoeffding, 1963) is used to set the initial value. Suppose we make $M$ independent observations of a random variable $r$ with range $R$. If we compute the mean of $r$, the Hoeffding bound states that the true mean of $r$ is at least the mean of $r$ minus $\varepsilon$, with probability $1-\delta$, where $\varepsilon$ is a user-specified parameter and $\delta$ is

$$\delta = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2N}}$$

(2.5)

Thus the threshold is estimated by the mean value of all value in the threshold table minus $\varepsilon$. Estimated threshold is used to extract the sequential patterns. If not
enough patterns are gathered, estimated threshold is considered to be an outlier. If the estimation is still less than the actual threshold for more than twice, the algorithm regards that the change of the threshold starts to transfer severely. Therefore actual threshold in the last three batches is taken to do the binomial linear regression. When the estimated threshold by the regression is correct, it has to be observed, whether the actual threshold matches the regression function or not. If the three threshold values of the last three batches from oldest to latest, named a, b, and c, satisfy the function $a \geq b \geq \text{cand}(a - b)/2 < (b - c)$, it is presumed that the curve trend of this function is downward. This is because the Hoeffding bound is not suitable in the condition that the threshold changes severely compared to previous batches. Thus regression function is adopted to estimate the threshold. Otherwise the original method is adopted to estimate the threshold of the next batch (Dai, 2010).
Chapter 3

Algorithms Proposed and Tested

3.1 Requirements for Implemented Algorithms

Designed architecture of autonomous agent requires both sequential and pattern learning capability. The main goal is to implement efficiently both of these learning approaches. Resulting algorithms will be inter-connected to produce hierarchy of nodes communicating among each other and enabling agent to make more complex decisions. Proposed agent will lack of a priori information about its surrounding environment. Potentially large amount of data will be processed by him in a reasonable time interval. Demands for the resulting subsystems:

- generality (the algorithms should be applicable on various agent-related problems)
- on-line learning and data processing (the data received in continuous data-streams and fast processing of incoming information)
- unsupervised (no training-set available, searching for inherent patterns without label-information)
- multi-dimensionality of incoming data (processing of n-dimensional vectors of real values)

3.1.1 Pattern recognition Algorithm

Implemented pattern recognizing node will receive un-labeled data in the on-line fashion with the aim of reducing their dimensionality to single value (labeling them). Because we
are unable to determine the number of resulting categories of the data we have to search among the on-line hierarchical clustering algorithms. The UPGMA creates the initial tree for the on-line hierarchical clustering procedure in the $O(n^2)$ time which is more than any other best-merge non persistent agglomerative clustering algorithm. Single-linkage clustering and Weighted Pair-Group Method with Arithmetic achieve equivalent time complexity. The drawback of the single-linkage clustering algorithm is that it is sensitive to outliers. It decides depending on the single value, which does not necessarily have to be corresponding to the actual cluster distribution. WPGMA is similar to UPGMA except that it assumes the weight of the cluster depending on its size. This assumption could be useful in certain cases but to preserve the generality of implemented solution we will rather restrict to the use of UPGMA.

So far only the solution for creating the hierarchy has been presented. Another goal of the algorithms is to acquire correct cluster distribution from the hierarchy. Inspired by existing implementation of the hierarchical clustering (Matlab, 2013) The inconsistency coefficient technique has been adopted to determine the correct number of the clusters from the dendrogram. We denote the distance $\delta$ of the node X as the distance of the clusters directly connected by the node X. For the leaf nodes and the parent nodes of the leaf nodes have inconsistency coefficient equal to 0. To calculate coefficients of other nodes we choose depth $D$. Then inconsistency coefficient $IC$ of the node $X$ is equal to:

$$IC = \frac{\delta_X - \mu(\delta_{Depth_X})}{\sigma(\delta_{Depth_X})}$$

(3.1)

Where $\delta_{Depth_X}$ stands for the distances of the nodes which are situated up to the depth of $D$ bellow the X. $\mu$ and $\sigma$ represent mean and standard deviation of these distances. To adopt on-line hierarchy seeking algorithm proposed by B. Farran for our problem, I modified the threshold($\theta$) constraint. Before adding the node to the exact place in the dendrogram we have to make sure that none of the direct or indirect child nodes of any nodes will have smaller value than its ancestor. Also we do not move down in the tree hierarchy unless the mean value of the child node is closer to the new node.

Moving down the tree hierarchy Iff:

$$\text{dist(newNode, parent)} > \text{dist(newNode, child)}$$

(3.2)

$$\delta \text{ of updated Parent} > \delta \text{ of updated Child}$$

(3.3)
3.1.2 Sequence Learning Algorithm

In previous chapter several sequence learning algorithms have been introduced. Because data will arrive continuously, we will focus on finding suitable data stream sequence mining algorithm. Order of the elements may also play important role in the event prediction, therefore sequential mining algorithms are preferred from association rules mining algorithms. To preserve the generality of implemented algorithm, assumption of different weights of the events is not feasible. In addition without prior information about events these assumptions would most likely lead to incorrect predictions. For the sake of both the fast processing of the on-line data and the generality the possibility of choosing the count of the patterns extracted in each batch seems more suitable than just simply dropping the patterns below minimum support. Concurrency of the implementation is crucial in order to extract the important patterns and predict upcoming events in the parallel way. Following table shows suitability of quoted algorithms for subsystem design.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SPEED</th>
<th>Top-K Seq in DS</th>
<th>$IW F P T_{WA}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction/Sequence</td>
<td>Sequence</td>
<td>Sequence</td>
<td>Transaction</td>
</tr>
<tr>
<td>(Un)Weighted</td>
<td>WeightedWeighted</td>
<td>Unweighted</td>
<td></td>
</tr>
<tr>
<td>Freq Patterns/Top-K</td>
<td>Frequent Patterns</td>
<td>Top-K</td>
<td>Frequent Patterns</td>
</tr>
<tr>
<td>Parallel processing</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of sequential pattern mining algorithms.

Suitability of algorithms for our problem - green(suitable), red(not-suitable), Parallel processing(1 - most optimized for concurrency, 2 - less optimized for concurrency)

The most suitable algorithm for proposed autonomous agent design is Top-K Sequential patterns in Data Stream. The possibility of choosing Top-K most promising patterns decreases computational complexity by reducing the number of the searched patterns. SPEED algorithm relies on minimum support when pruning the sequences, which might be insufficient factor when dealing with vast amount of information. In addition Top-K predicts the maximal threshold required for mining these patterns. Yet certain challenges need to be met when utilizing Top-K patterns in required pattern recognizing node. Output value received from hierarchical clustering node is real-valued vector. Alphabet will therefore consist of real-valued vectors instead of the characters. Bigger problem arises by the fact that Top-K sequences operates on sequential databases. But input for the system consists of continuous stream of data. The length of the sequences is unknown and no database is available. Algorithm proposed by me allows user to choose initial
sequence size(\(\alpha\)), step size(\(\delta\)) and batch size(\(\beta\)). After enough data has been received to form a batch. Batch is divided into sequences with size of \(\alpha\). Sequences are then gathered using Top-K Sequential patterns in DS algorithm. We then define total gain from the batch as \(\varphi\). After that algorithm for mining sequences runs again but this time sequence size is modified by adding \(\delta\) and forming new sequences. \(\varphi\) is calculated again and in each iteration, the sequence with highest \(\varphi\) is saved. We proceed in this way until certain processing time bound(\(t\)) is reached, we then use result with the highest \(\varphi\) found during processing of the batch. We calculate \(\varphi\) as:

\[
\varphi = \sum_{i=1}^{K} \text{Confidence}(i) \times \text{Support}(i) \tag{3.4}
\]

\(\text{Seq}_i\) - frequent sequences of size \(i\);

**Input:** \(A\) - array of size \(\beta\), \(t\) - processing bound, \(\alpha\), \(\delta\)

**Output:** Top-K Sequences(\(S\))

\(\text{SeqSize} = \alpha\);
\(S = \text{null}\);
\(\varphi_* = 0\);

**while** \(\text{CurrentTime} < t\) **do**

\(\text{Batch} = \text{Form\_Batch\_Of\_Sequences}(A, \text{SeqSize})\);
\(\text{TopK} = \text{Get\_Top\_K\_patterns}(\text{Batch})\);
\(\varphi = \text{Calculate\_gain}(\text{TopK})\);

**if** \(\varphi > \varphi_*\) **then**

\(\varphi_* = \varphi\);
\(S = \text{TopK}\);

**end**

\(\text{SeqSize} += \delta\);

**end**

**Algorithm 6:** Proposed algorithm for estimating best sequence size for mining sequences with highest interestingness and mining these sequences
In addition to mining the Top-K patterns, our goal is to predict the event, with the highest mined occurrence count depending on the previously received elements. Proposed algorithm recursively takes last n received elements and finds most probable subsequent element in FP Tree. It starts with n equal to 0 and recursively increases n by 1. It terminates after reaching certain threshold for n. From all candidate items found it then chooses the one with the highest confidence as the predicted item.

```
Input: FP_Tree, LastItems - Array of last received elements
Output: predictedItem

BestFoundConfidence = 0;
n = 0;
while n < size(LastElements) do
    SearchedSequence = takeLastItemsOf(LastElements,n);
    [Confidence, Item] = searchCandidate(FP_Tree, SearchedSequence);
    if Confidence > BestFoundConfidence then
        BestFoundConfidence = Confidence;
        predictedItem = Item;
    end
    n = n + 1;
end
```

*Algorithm 7*: Proposed algorithm for predicting most probable item
Chapter 4

Experiments

In order to examine the implemented algorithms I have concluded several experiments both in virtual environment and by investigating the behavior of algorithms when processing datasets.

4.0.3 Pattern recognition Experiments on Processing Datasets

Implemented hierarchical clustering algorithm has been tested using the Iris dataset from the UCI Machine Learning Repository (Fisher, 1988). This dataset contains 150 samples of three different types of the plant.

Figure 4.1: Performance of the implemented on-line hierarchical clustering depending on the initial tree size. Inconsistency factor = 0.7 × Inconsistency factor of the root.
From the graphs can be concluded that running purely updating algorithm on the data produces higher classification error than using the tree initialization at first. It is essential for the purpose of algorithm to avoid the frequent use of the initialization of the three because of the higher time complexity. The most satisfying result was obtained when using 1/3 of the dataset for the tree initialization and the rest had been updated in the on-line fashion. By using more than 1/2 of the dataset for the initialization too many classes were obtained when compared to the amount of real classes.

(a): Synthetic dataset (500 samples) .

(b): IRIS dataset (150 samples) .

Figure 4.2: The visualization of the obtained clusters of the implemented on-line hierarchical clustering algorithm. In (a) all clusters where determined purely by tree update. In (b) 1/2 of the data was used to create initial tree.

4.0.4 Sequence Learning Experiments on Processing Datasets

This part investigates the features of the implemented sequential learning algorithm mentioned in the previous section. Sequential Pattern Mining Framework (Fournier-Viger, 2013) is an open-source data mining platform written in Java. It allows programmers to use and integrate source code of each algorithm in their Java software. SPMF has been used to compare my algorithm with other sequential pattern mining algorithms. Anonymous web data gathered by recording the web page msbbc.com (Heckerman, 1999) has been used as a testing dataset for evaluating and comparison of algorithms. Implemented recognizer has been compared to the algorithms PrefixSpan and Top-K Sequential rules developed by the authors of the SPMF. Algorithms have been tested both on the relatively large
and smaller data sets to capture all the possible differences among them.

Figure 4.3: Performance of applying the sequence mining algorithms on the smaller dataset. $K = 100$, minimum confidence = 0.2, BatchSize = 500, FP tree remembers all mined patterns. Average of 10 runs of the algorithms.

Figure 4.4: Performance of applying the sequence mining algorithms on the bigger dataset. $K = 100$, minimum confidence = 0.2, batchSize = 5000, and FP tree had been set again to remember all mined patterns. Color representation in figure (b) and (a) is consistent. Average of 10 runs of the algorithms.

From the evaluations of the graphs the following observation has been made. Implemented algorithm deals with the mining of the patterns from the the smaller database.
relatively well compared to the other algorithms, but it really starts to benefit and beats the other algorithms when it processes the large amount of the data. The memory requirement appears to grow linearly at the beginning and after a certain amount of the patterns have been saved into the FP tree, the growing of the complexity curve terminates. Processing time of the algorithm depending on the patterns processed grows in the linear way with relatively small gradient compared to the other algorithms.

Another conducted experiment aimed to determine the effect of keeping frequent patterns in the FP Tree. The same dataset as in the previous experiment has been used. In the virtual environment of autonomous agent, where rules are constantly changing, the effect of keeping would most likely be different. This experiment only shows, the situation in environment which is changing slightly or not changing at all. At the same time the environment provides rules with high level of uncertainty.

![Graph](image)

**Figure 4.5:** Dependence of the number of the batches remembered by FP Tree on Maximal memory usage and success. BatchSize = 1000. K = 500.
4.0.5 Sequence Learning Experiments in the Virtual Environment.

Processing of large datasets by implemented algorithm provides the idea of how fast can algorithm process the data, but it is insufficient to determine the behavior of the algorithm when processing data received from the virtual environment. In order to evaluate the suitability of the implemented algorithm for the agents architecture proposed by my supervisor I conducted experiments simulator of the autonomous agents Vivae developed at the CTU. The agent is moving in the rectangular route and avoiding the curves. It is equipped both with sensors measuring the distances from the objects, movement and the friction sensors. Simulator enables extracting of the data from the sensors and actuator of the virtual agent. These data has been used to evaluate the implemented algorithm.

In the first set of virtual environmental experiments, the output of the robot movement sensor for detecting the curves has been used. This movement sensor produces output of 0 real value in the case of the curve occurrence and 1 real value, if the agent moves along the straight line. This sort of output is categorical and consistent to the output of

the hierarchical clustering algorithm. Data from 20 passes of the agent through the track and the precision of 0.01 second had been collected and utilized to perform evaluations. These 20 passes will be referred to as 40 rounds throughout the experiment descriptions. Obviously the path is symmetrical and we can perform this division. One round took approximately 1.00 second. In the following predictions of the curves the prediction is considered as incorrect if it is not within the $(-0.2, 0.2)$ time distance from the actual curve.

In the first conducted experiment in the virtual environment the performance of the first upcoming event prediction depending on the prediction start has been tested. In the second graph the prediction of the more distant events is examined.
CHAPTER 4. EXPERIMENTS

Figure 4.8: Algorithm performance on the curves prediction. Batch size = 1 round, \( K = 5000 \), FP tree set to contain all mined patterns.

(a): Error depending on the prediction start.

(b): Error depending on the time distance of the event.

By observing the graphs showing the batch dependence of the algorithm, it can be seen that the performance of the algorithm is relatively highly dependent on the chosen batch size for sequential mining algorithm. We also see that the bigger batch size generally provides less ‘daring’ predictions. The incorrect prediction counts remain smaller than those in case of the smaller batch size. To assure the generality of the algorithm in the virtual environment it is therefore better for the algorithm to use bigger batch size. By using this setup, the predictor does not necessarily need to produce bigger amount of exact predictions but the rate of the incorrect prediction remains lower.
In the following experiment I compared the implemented algorithm with the SPMF Top-K Sequential rules in the virtual environment. Because algorithm from SPMF processes integers and not arrays of real values the data from the single movement sensor have been extracted. The values have been multiplied and converted to integers. The plotted errors differ from the real data by no more than 0.005 of deviation. Algorithm implemented by SPMF does not provide direct algorithm for predicting the sequences but set of rules is generated. To evaluate the predictions intuitive assumption of choosing the pattern with the highest confidence has been made. On the background of this assumption algorithm of SPMF has been extended by prediction capability. The processing time of the prediction was not evaluated during the experiment. Data has been divided into sequences of the same length and the algorithm for choosing the best sequence size from the batch has been disabled during the experiment to provide the same conditions for both tested algorithms.

(a): Average distance of predicted events from the actual ones. (b): Average processing time during runtime.

Figure 4.10: Comparison of Top-K Sequences(SPMF) and Implemented algorithm.

Implemented algorithm when used in the virtual environment beats the Top-K Sequential Rules in both processing time and prediction capability. The FP Tree structure enables remembering and searching for the patterns efficiently. Top-K Sequences generates rules by extending RH side and LH side of the rule. Implemented algorithm stores frequent sequences and after that determines the rules. When predicting to the more distant future this can cause a small time overhead but it is negligible compared to the speed of the sequential pattern mining.
Chapter 5

Thesis Conclusion and Future Work

The implemented sequential pattern mining and predicting algorithms proved to be useful for the stated problem. By encapsulating them into the ROS node they can interact with other nodes and contribute to the evaluation of the architecture of the autonomous agent proposed by my supervisor. My design of the sequential mining algorithm proved that the capability of the database mining algorithms can be extended and reused in the architecture of the algorithm suitable for the data stream processing of the incoming items. The algorithm appears to be faster and more precise for solving the stated problem then some of the algorithms implemented by the colleges in this field. Furthermore algorithm is capable of predicting and processing the sequences in real time. The prediction of the event happening in the more distant future is possible without the severe loss of precision. The implemented on-line hierarchical clustering algorithm proved that the hierarchy of the clusters can be constructed in $\Omega(n)$ time and $\Omega(n)$ memory requirement. Although the resulting successfulness of the classification is less precise the implemented algorithm meets the real-time constraint which is essential for the use in the proposed agents architecture.

All the assigned tasks have been successfully fulfilled:

- Basic types of learning and generating sequential patterns together with sequential learning in the fluent data streams has been carefully studied and examined. This task has been fulfilled in the Chapter 2.

- The most suitable algorithms have been implemented and encapsulated in the ROS node and compatible with the ROS. The implemented algorithms are described in the Chapter 3 and the source code is burned on the appended CD.
• Implemented algorithms have been tested and compared to the existing implementation of sequential pattern mining algorithms. The results of the testing is described in the Chapter 4. Algorithm for learning sequential patterns proved to be useful both for the processing of the data captured from the agents environment and data from the large databases. It is capable of fast processing of the data and predicting the sequences with high rate of successfulness compared to the tested other algorithms. In addition it is able to predict sequences of the more distant events.

• The benefits of the sequential learning have been tested in the Chapter 4. The ability of the agent to predict the events taking place in the future provides him with more information about the surrounding environment. In the Chapter 4 it has been proved for example that agent can successfully predict the curves of the virtual trajectory.

Although both of the ROS nodes have been successfully implemented still great amount of work lies ahead. To assure the cooperation of the ROS nodes in the virtual environment, the performance of the hierarchical clustering algorithm in the virtual environment needs to be evaluated. Moreover both algorithms have to be evaluated when collaborating in the single agents instance.

My thesis brings benefits in the field of the sequential pattern learning. The implementation of the algorithms is suitable both for autonomous agents design and for extracting patterns from the databases. The sequential learning is very interesting topic and recently many publications have arisen. If we continue with the improvement of the existing algorithms and with creation of the new ones even greater performance can be achieved. The resulting algorithms could further improve the sequence prediction and as a consequence enhance the capabilities of the autonomous agent. To summarize many accomplishments have been reached but more improvements are possible in this promising area.
Bibliography


BIBLIOGRAPHY


Pei and Han (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth.


