Chapter – 1

Introduction
CHAPTER – 1
INTRODUCTION

1.1. BACKGROUND

“If I have seen further it is by standing on the shoulders of Giants”

- Issac Newton

Data Mining extracts pattern which contains itemsets and its support, i.e. the number of transactions that contain it. The huge size of transaction databases and the exponential increase in the number of frequent itemsets make the knowledge discovery into a challenging task.

Biological sequences consist of only four nucleotide characters namely Adenine (A), Cytosine (C), Guanine (G) and Thymine (T). DNA sequence is a big linear chain made of only four characters A, T, C and G. So it is long list of only 4 items. Similarly protein is also of 27 characters. Thus bio-sequences are long in size and consist of limited item list (4 for DNA, 27 for Protein). Two biological sequences are similar is no different from saying that two strings are similar in computer terminology. Biological sequence analysis is therefore rooted into computer science.

DNA sequences data is having huge amounts of data and is growing rapidly to petabytes size due to advancements in genomics and proteomics sequencing techniques. The identification of Bio-sequence patterns that play roles in various biological functions, genetic diseases and evolution, is challenging. This requires a great deal of research in computational algorithms, statistics, mathematical programming, data mining and machine learning. Efficient and faster information retrieval helps to develop effective genomic and proteomic data analysis tools.
1.2. DATA MINING

Data Mining refers to the process called knowledge discovery in databases. It actually extracts knowledge from data. The knowledge discovery process includes several pre-processing (or data preparation) and post-processing (or knowledge refinement) steps. The data preparation method is used to transform the data to facilitate the application of a given data mining algorithm(s). The knowledge refinement method is used to validate and refine discovered knowledge.

The knowledge discovery process is both iterative and interactive. It is iterative because the output of each step is often feedback to previous steps, as shown in Figure 1.1. This knowledge discovery process requires much iteration to extract high-quality knowledge from data. It is interactive because the user or an expert in the application domain, should be involved in this loop, to help in data preparation, discovered-knowledge validation and refinement, etc. (Freitas, 2002).

![Figure 1.1 Iterative nature of the knowledge discovery process](image)

1.2.1. Data Mining Tasks

The common functions in data mining practice include:

1. **Association rule discovery**: Describes association relationship among different attributes. The origin of association rules is in market basket analysis. A market basket is a collection of items purchased by a customer in an individual customer
transaction. One common analysis task in a transaction database is to extract pattern which contains itemsets and its support, i.e. the number of transactions that contain it. Knowledge of these patterns helps businesses to improve their stores for ordering items and for mail-order marketing. The huge size of transaction databases and the exponential increase in the number of frequent itemsets make the knowledge discovery into a challenging task.

2. **Clustering:** A process which maps a data item into one of several clusters. Clusters are formed from natural grouping of data items based on their similarity or probability density models. Clustering is used in number of data analysis tasks such as customer retention & management and web mining.

3. **Classification:** A process which classifies a data item into one of several predefined categorical classes. It is used for predictive data mining in several fields, like scientific discovery, fraud detection, atmospheric data mining and financial engineering.

4. **Sequence analysis:** A process to design models for sequential patterns like time-series data. This process helps to generate the sequence or to extract report deviation and trends over time. This framework is gaining importance recently because of its application in bioinformatics and streaming data analysis.

5. **Regression:** A process to map a data item into a real-valued prediction variable. It is used in different prediction and modelling applications.

6. **Summarization:** A process to provide a compact description for a subset of data such as mean and standard deviation for all fields. Summarization functions are often used in interactive data analysis, automated report generation and text mining.

7. **Dependency modelling:** A process to describe significant dependencies among variables.
1.2.2. Association Rules

An association rule is an expression \( A \Rightarrow B \), where \( A \) and \( B \) represent two different sets of items (Pal & Mitra, 2008). Given a database \( D \) of transactions where each transaction \( T \in D \) is a set of items. The rule \( A \Rightarrow B \) shows that if transaction \( T \) contains \( A \) then \( T \) also contains \( B \). The confidence \( c \) is defined as the percentage of transactions containing \( B \) and \( A \) with respect to the number of transactions containing \( A \). The support of a rule is defined as the number of transactions containing \( A \) (irrespective of presence of \( B \)) with respect to the total number of transactions in a database.

Many generalized mining association rules for frequent itemset have been proposed. Apriori (Agrawal & Srikant, 1994) is most influential algorithm for mining Boolean association rules for frequent itemsets. There are two main problems while mining these association rules namely algorithmic complexity and the interesting rules must be picked up from the set of generated rules.

1.2.3. Frequent Pattern

Frequent patterns are patterns of itemsets or subsequences or substructures that appear frequently in a data set. For example, a set of items like milk and bread that appear frequently together in a transaction data set is a frequent itemset. A subsequence, such as buying first a PC, then a digital camera, then a memory card in a sequence, if it occurs frequently in a shopping history database, is a frequent sequential Pattern.

Frequent pattern mining searches for recurring relationships in a given data set. Mining frequent itemsets leads to the discovery of associations and correlations among items in large transactional or relational data sets.
After mining the frequent items from the transactions in a database the strong association rules can be generated which satisfy both minimum support and minimum confidence. The following equation defines the confidence.

\[
\text{Confidence } (A \Rightarrow B) = \frac{\text{support_count}(A \cup B)}{\text{Support_count}(A)}
\]

Where \(\text{support_count}(A \cup B)\) is the number of transactions containing the itemsets \((A \cup B)\) and \(\text{support_count}(A)\) is the number of transactions containing itemsets \(A\).

Based on the confidence equation association rules can be generated as follows:

- For each frequent itemset \(l\), generate all nonempty subsets of \(l\)
- For every nonempty subsets \(s\) of \(l\), output the rule “\(s \parallel (l-s)\)”, if \(\frac{\text{support_count}(l)}{\text{support_count}(s)} \geq \text{min}_\text{conf}\), where \(\text{min}_\text{conf}\) is the minimum pre-assigned confidence threshold (Han & Kamber, 2006).

Since the rules are generated from frequent itemsets, each one automatically satisfies the minimum support criterion.

### 1.2.4. Sequential Pattern Mining

A sequence database consists of ordered events, recorded with or without concrete notation of time. Sequential pattern mining is the mining of frequently occurred events or sub sequences as patterns. Sequential pattern mining is focussed on categorical or symbolic patterns.

Sequential pattern mining is computationally challenging because such mining may generate a large of number of intermediate subsequences. Many efficient and scalable algorithms have been developed for frequent Itemset mining from which association and correlation rules can be derived. These algorithms can be classified into 3 categories namely 1. Apriori-like algorithms, 2. Frequent pattern growth based
algorithm such as FP-Growth and 3. Algorithms that use the vertical data format (Han & Kamber, 2006).

The Apriori algorithm is the most influential algorithm for mining frequent itemsets for Boolean association rules. It applies the level-wise search approach. Apriori property states that “All nonempty subsets of a frequent itemset must also be frequent”. At the $k^{th}$ iteration, it forms frequent k-itemset candidates based on the frequent (k-1)-itemsets and scans the database once to find the complete set of frequent k-itemsets.

Frequent pattern growth (FP-growth) is a method of mining frequent itemsets without candidate generation. It constructs a highly compact data structure called FP-tree to compress the original transaction database. This method focuses on frequent pattern growth which avoids costly candidate generation resulting in greater efficiency.

Mining frequent itemsets using vertical data format is a method that transforms a given data set of transactions in the horizontal data format into the vertical data format. It mines the transformed data set based on the Apriori property and few additional optimization techniques.

Methods for mining frequent itemsets can be extended for the mining of closed frequent itemsets which help to get frequent itemsets easily. Closed frequent itemset is defined as “Let $X$ is a closed frequent itemset in a data set $S$ if there exists no proper super-itemset $Y$ such that $Y$ has the same support count as $X$ in $S$, and $X$ satisfies minimum support”. Closed frequent itemsets can substantially reduce the number of patterns generated in frequent itemset mining. The set of closed frequent itemsets can easily derive the set of frequent itemsets and their support. Thus in
practice, it is more desirable to mine the set of closed frequent itemsets rather than the set of all frequent itemsets in Apriori like algorithms.

The two directions to develop efficient and scalable methods for sequential pattern Mining are (i) Mining the full set of sequential patterns and (ii) Mining only the set of closed sequential patterns.

1.2.5. Mining Full Set of sequential patterns

The three major approaches for mining the full set of sequential patterns are, represented by the algorithms GSP-Generalized Sequential Patterns (Ramakrishnan & Agrawal, 1996), SPADE-Sequential Pattern Discovery using Equivalent classes (Zaki, 2001) and PrefixSpan-(Prefix-projected Sequential Pattern mining (Pei et al., 2001) respectively. GSP adopts a candidate generate-and-test approach using horizontal data format (where the data are represented as (sequence ID: sequence of itemsets), as usual, where each itemset is an event). SPADE adopts a candidate generate and-test approach using vertical data format (where the data are represented as (itemset: (sequence ID, event ID)). The vertical data format can be obtained by transforming from a horizontally formatted sequence database in just one scan. PrefixSpan is a pattern growth method, which does not require candidate generation.

All three approaches either directly or indirectly explore the Apriori property “Every nonempty subsequence of a sequential pattern is a sequential pattern”. The Apriori property is anti-monotonic (or downward-closed) i.e. “If a sequence cannot pass a test, all of its super sequences will also fail the test”. This property helps to prune the search space and make the discovery of sequential patterns more efficient.
1.2.6. Mining Closed Sequential Patterns

Closed subsequences contain no super sequence with the same support. Mining closed sequential patterns can produce a significantly less number of sequences than the full set of sequential patterns.

It is a good approach is to search for closed frequent itemsets directly during the mining process itself. This requires us to prune the search space as soon as the identification of closed itemsets during the mining process. Pruning strategies include the following:

**Item merging**: “If every transaction containing a frequent itemset X also contains an itemset Y but not any proper superset of Y, then X ∪ Y forms a frequent closed itemset and there is no need to search for any itemset containing X but no Y”.

**Sub-itemset pruning**: “If a frequent itemset X is a proper subset of an already found frequent closed itemset Y and support_count(X) = support_count(Y), then X and all of X’s descendants in the set enumeration tree cannot be frequent closed itemsets and thus can be pruned”.

**Item skipping**: “In the depth-first mining of closed itemsets, at each level, there will be a prefix itemset X associated with a header table and a projected database. If a local frequent item p has the same support in several header tables at different levels help to prune p safely from the header tables at higher levels”.

Besides pruning the search space in the closed itemset mining process, another important optimization is to perform efficient checking of a newly derived frequent itemset to see whether it is closed, because the mining process cannot ensure that every generated frequent itemset is closed.
CloSpan-Closed Sequential Pattern Mining (Yan, et al., 2003) is an efficient closed sequential pattern mining method. This method is based on a property of sequence databases, called Equivalence of projected databases, stated as follows: “Two projected sequence databases, \( S \mid \alpha = S \mid \beta, \alpha \subseteq \beta \) (i.e. \( \alpha \) is a subsequence of \( \beta \)), are equivalent if and only if the total number of items in \( S \mid \alpha \) is equal to the total number of items in \( S \mid \beta \)”, where \( S \mid \alpha \) and \( S \mid \beta \) are the project databases of sequence database \( S \) with respect to prefix \( \alpha \) and \( \beta \) respectively. CloSpan stops growing one prefix-based projected database (called non closed sequential patterns), whenever it finds two prefix-based projected databases that are exactly the same. After such pruning and mining, a post processing step is still required in order to delete non-closed sequential patterns that may exist in the derived set. The algorithm called BIDE-BI-Directional Extension (Wang & J, 2004) which performs a bidirectional search and thus it avoids the process of additional checking for existence of non-closed sequential patterns.

Mining that is performed without user- or expert-specified constraints may generate numerous patterns that are of no interest. Such unfocused mining can reduce both the efficiency and usability of frequent-pattern mining. Thus, constraint-based mining incorporates user-specified constraints to reduce the search space and derives only patterns that are of interest to the user. Constraints based rule mining allows users to focus the search for rules by providing Meta rules (i.e. pattern templates) and additional mining constraints. Such mining is facilitated with the use of a declarative data mining query language and the user into force and poses great challenges for mining query optimization. Rule constraints can be classified into five categories: anti-monotonic, monotonic, succinct, convertible and inconvertible. The first four of
these categories can be used during frequent itemset mining and leading to more efficient and effective mining.

1.2.7. Biological Data Mining

Biological data mining has become an essential part of genomics, proteomics, functional genomics and bio-medical research.

DNA is well-suited for biological information storage. The DNA backbone is resistant to cleavage, and both strands of the double-stranded structure store the same biological information. Biological information is replicated as the two strands are separated. A significant portion of DNA (more than 98% for humans) is non-coding, meaning that these sections do not serve as patterns for protein sequences.

The two strands of DNA run in opposite directions to each other and are therefore anti-parallel. Attached to each sugar is one of four types of nucleotide (informally, bases). It is the sequence of these four nucleotides along the backbone that encodes biological information. Under the genetic code, RNA strands are translated to specify the sequence of amino acids within proteins. These RNA strands are initially created using DNA strands as a template in a process called transcription.

In molecular biology, the term double helix (Anon., 2015) refers to the structure formed by double-stranded molecules of nucleic acids such as DNA. The double helical structure of a nucleic acid complex arises as a consequence of its secondary structure, and is a fundamental component in determining its tertiary structure. The term entered popular culture with the publication in 1968 of The Double Helix: A Personal Account of the Discovery of the Structure of DNA, by James Watson.
The DNA double helix polymer of nucleic acids, held together by nucleotides which base pair together. In B-DNA, the most common double helical structure, the double helix is right-handed with about 10–10.5 nucleotides per turn. The double helix structure of DNA contains a major groove and minor groove, the major groove being wider than the minor groove. Given the difference in widths of the major groove and minor groove, many proteins which bind to DNA do so through the wider major groove.

Figure 1.2 Double Helix Structure of DNA
Protein is the functional molecule of life determined by the sequences of amino acid. These proteins are formed by the triplets of Codon parts of DNA sequences. The following table 1.1 lists out DNA triplets for all twenty proteins.

<table>
<thead>
<tr>
<th>Amino acid</th>
<th>Codons</th>
<th>Compressed</th>
<th>Amino acid</th>
<th>Codons</th>
<th>Compressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ala/A</td>
<td>GCT, GCC, GCA, GCG</td>
<td>GCN</td>
<td>Leu/L</td>
<td>TTA, TTG, CTT, CTC, CTA, CTG</td>
<td>YTR, CTN</td>
</tr>
<tr>
<td>Arg/R</td>
<td>CGT, CGC, CGA, CGG, AGA, AGG</td>
<td>CGN, MGR</td>
<td>Lys/K</td>
<td>AAA, AAG</td>
<td>AAR</td>
</tr>
<tr>
<td>Asn/N</td>
<td>AAT, AAC</td>
<td>AAY</td>
<td>Met/M</td>
<td>ATG</td>
<td></td>
</tr>
<tr>
<td>Asp/D</td>
<td>GAT, GAC</td>
<td>GAY</td>
<td>Phe/F</td>
<td>TTT, TTC</td>
<td>TTY</td>
</tr>
<tr>
<td>Cys/C</td>
<td>TGT, TGC</td>
<td>TGY</td>
<td>Pro/P</td>
<td>CCT, CCC, CCA, CCG</td>
<td>CCN</td>
</tr>
<tr>
<td>Gln/Q</td>
<td>CAA, CAG</td>
<td>CAR</td>
<td>Ser/S</td>
<td>TCT, TCC, TCA, TCG, AGT, AGC</td>
<td>TCN, AGY</td>
</tr>
<tr>
<td>Glu/E</td>
<td>GAA, GAG</td>
<td>GAR</td>
<td>Thr/T</td>
<td>ACT, ACC, ACA, ACG</td>
<td>ACN</td>
</tr>
<tr>
<td>Gly/G</td>
<td>GGT, GGC, GGA, GGG</td>
<td>GGN</td>
<td>Trp/W</td>
<td>TGG</td>
<td></td>
</tr>
<tr>
<td>His/H</td>
<td>CAT, CAC</td>
<td>CAY</td>
<td>Tyr/Y</td>
<td>TAT, TAC</td>
<td>TAY</td>
</tr>
<tr>
<td>Ile/I</td>
<td>ATT, ATC, ATA</td>
<td>ATH</td>
<td>Val/V</td>
<td>GTT, GTC, GTA, GTG</td>
<td>GTN</td>
</tr>
<tr>
<td>START</td>
<td>ATG</td>
<td></td>
<td>STOP</td>
<td>TAA, TGA, TAG</td>
<td>TAR, TRA</td>
</tr>
</tbody>
</table>

For example, Codon part (Functional part - 48..92) and the respective Protein part for the protein namely “mitochondrial 28S ribosomal protein S14; MRPS14” as shown below.
1 ggcagtgtca ataaagtttc agcggtttgt agtttgtagc ggacaac **atg** **gcg** **gcc** **ttc** **a**

61 **tgctgggetc getgetgegg aegttcaagec aggtcaggcc** tcctactttta tccacaccgc

<table>
<thead>
<tr>
<th>1</th>
<th><strong>maafmlgsll rtfkq</strong> mvps ss asgqvrshyv dwrmrdrvkr rkmayeyade rlinslrkn</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td><strong>tilpkilqdvd adeciaalpr dscpvrirnr cvmtrsprgv krrwrlsriv frhladhgql</strong></td>
</tr>
<tr>
<td>121</td>
<td><strong>sgiqratw</strong></td>
</tr>
</tbody>
</table>

### 1.3. VARIOUS DNA SEQUENCE FORMATS

Generally, DNA Sequence formats are in ASCII TEXT. Sequence formats are the required arrangement of characters, symbols and keywords that specify what things such as the sequence, ID name, comments, etc. look like in the sequence entry and where in the entry the program should look to find them. There are generally no hidden, unprintable ‘control’ characters in any sequence format. All standard sequence formats can be printed out or viewed simply by displaying their file.

Most sequence formats include at least one form of ID name, usually placed somewhere at the top of the sequence format. An entry in a database must have some way of being uniquely identified in that database. Most sequence databases have two such identifiers for each sequence - an ID name and an Accession number. Most formats allow you to hold other description, annotation and comments, for example Fasta format holds comments in the title line.

Nucleotide (DNA or RNA) sequences are usually stored in the IUBMB standard codes. There are at least a couple of dozen sequence formats in existence at the moment. Some are much more common than others.

Here are the examples of various DNA sequence formats (Anon., 2015).
1. Plain sequence format

A sequence in plain format may contain only IUPAC (International Union of Pure and Applied Chemistry) characters and spaces (no numbers!). A file in plain sequence format may only contain one sequence, while most other formats accept several sequences in one file.

An example sequence in plain format is:

ACAGATGCATTTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCC
ACGGCCACCGCTGCCCTGCCCCCTGGAGGGTGGCCACCCGCCCGAGACA
GCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGA
CTTTCCCTCGCTTTGGTGTTTGAAGTGACCTCCCCAGGCCAGTGCCGGGCC
CTCATAGGAGAGGAAGCTCGGAGGTGCGCCAGCCGCGGAAAGGCAGCA
CCCCGCCAGCAATCCGCGCAGGGACAGAATGCGCTCAGGAGATTCTCT
TCTGGAAGACCTTCTCCTCCTGCAAATTTAATACCTCACCCATGAATGCTCA
CGCAAGTTTAATTACAGACCTGAA

2. EMBL format

A sequence file in EMBL (European Molecular Biology Laboratory) format can contain several sequences. One sequence entry starts with an identifier line ("ID"), followed by further annotation lines. The start of the sequence is marked by a line starting with "SQ" and the end of the sequence is marked by two slashes ("//").

An example sequence in EMBL format is:

| ID   | AB000263 standard; RNA; PRI; 368 BP. |
| XX   |                                   |
| AC   | AB000263;                          |
| XX   |                                   |
| DE   | Homo sapiens mRNA for prepro cortistatin like peptide, complete cds. |
| XX   |                                   |
| SQ   | Sequence 368 BP;                   |
3. FASTA format

A sequence file in FASTA format can contain several sequences. Each sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line must begin with a greater-than (">") symbol in the first column.

An example sequence in FASTA format is:

```
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA for prepro cortistatin like peptide, complete cds.|len=368
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCC
ACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGCGACAGCGAGCATATGCAGGAAGCG
GCAGGAATAAGGAAAAAGCAGC
CTCCTGACTTTCTCGCTTGGTGTTTGAGTGACCTCCAGGCCAGTGCC
CGGGCCCTCATAGGAGAGG
AAGCTCGGGAGGTCGCCAGACAGGAGGAGCAGCGACCCCACTCATATG
CGCACCCTTCAGAAGGACCCAGG
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTGCAAATAAAACCTCA
CCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA
```
4. GCG format

A sequence file in GCG format contains exactly one sequence, begins with annotation lines and the start of the sequence is marked by a line ending with two dot ("..") characters. This line also contains the sequence identifier, the sequence length and a checksum. This format should only be used if the file was created with the GCG package.

An example sequence in GCG format is:

| ID   | AB000263 standard; RNA; PRI; 368 BP. |
| XX   |                                       |
| AC   | AB000263;                              |
| XX   |                                       |
| DE   | Homo sapiens mRNA for prepro cortistatin like peptide, complete cds. |
| XX   |                                       |
| SQ   | Sequence 368 BP;                       |
|      | AB000263 Length: 368 Check: 4514 ..    |
| 1    | acaagatgcc attgtcccc gcgcctctgc tgctgtcgtctctcggg gcccaccgcggcaccgcgg |
| 61   | ctgcctgcc ctggagaget gccccaccc gcgcagacag cgacatatcgacgagaggg |
| 121  | caggaataag gaaaaacacgc ttcctgacttcctgcggtggtggttggacgacgtc |
| 181  | aggcaggcg ccggccccctc ataggagaggg aagctcggg aagctcgggg cggcaggag |
| 241  | gcgcaccccc accaattcc gcgcgcgggg acagattgcc tgcaggaacctttccttga |
| 301  | agctcctctgccaaa taaaaacctca ccatgaatgcctcagcaagttttaattaca |
| 361  | gacctgaa                                    |

5. GCG-RSF (rich sequence format)

The new GCG-RSF can contain several sequences in one file. This format should only be used if the file was created with the GCG package.
6. GenBank format

A sequence file in GenBank format can contain several sequences. One sequence in GenBank format starts with a line containing the word LOCUS and a number of annotation lines. The start of the sequence is marked by a line containing "ORIGIN" and the end of the sequence is marked by two slashes ("//").

An example sequence in GenBank format is

<table>
<thead>
<tr>
<th>LOCUS</th>
<th>AB000263</th>
<th>368 bp</th>
<th>mRNA</th>
<th>linear</th>
<th>PRI 05-FEB-1999</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINITION</td>
<td>Homo sapiens mRNA for prepro cortistatin like peptide, complete cds.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCESSION</td>
<td>AB000263</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORIGIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 acaagatgcc attgccccc ggcctcctgc tgcgtgtct ctcgcccccc acggccaccc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61 ctgcctgcc cctggaggt gggcccaccg gcgcagacag cgcgtatgc cggagaagccg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>121 caggataag gaaaaagcag ctcctgactt tctcctgttg gttgttgag tcggacatcc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>181 aggccagtgc cgggccccctc ataggaggg aggtcgggaa gttgcccagg cggcaggaag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>241 gcgcaccccc ccagcaatcc gcgcgcgggg acagaaagcc ctgcaggaac tttttcgg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>301 agacctcctc tctctgcaa taaaaacctca cccatgaatg ctacagccag gtttactaca</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>361 gcacag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>//</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. IG format

A sequence file in IG format can contain several sequences, each consisting of a number of comment lines that must begin with a semicolon (";"), a line with the sequence name (it may not contain spaces!) and the sequence itself terminated with the termination character '1' for linear or '2' for circular sequences.
An example sequence in IG format is

<table>
<thead>
<tr>
<th>; comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>; comment</td>
</tr>
<tr>
<td>AB000263</td>
</tr>
<tr>
<td>ACAAGATGCCATTGTCCCGGCTGCTGGCTCTCCCCGGGACC</td>
</tr>
<tr>
<td>ACGGCAACGCTGCTCCCTGCC</td>
</tr>
<tr>
<td>CCTGGAGGGTGGCCCCACCGGCGAGACAGCAGCATATGAGGAGAAGCG</td>
</tr>
<tr>
<td>GCAGGAATALAGGAAGGACAGC</td>
</tr>
<tr>
<td>CTCCTGACTTTCTGGATTGTTGGATGGAGGACCTCCAGCCAGCAGTGG</td>
</tr>
<tr>
<td>CGGCGGCTCTTATTAGGAGG</td>
</tr>
<tr>
<td>AAGCTCGGAGGGTGGCCAGCAGGAGACGAGCACCGGCAGCGAGCAGCAG</td>
</tr>
<tr>
<td>CCCATGAAATGGGCTCACGGAAG</td>
</tr>
<tr>
<td>TTTAAATTACAGACCTGAAA</td>
</tr>
</tbody>
</table>

1.4. DNA DATABASES IN PUBLIC DOMAIN

Biological databases have become an important tool in assisting life scientists to understand and explain a host of biological phenomena from the structure of biomolecules and their interactions, to the whole metabolism of organisms and to understand the evolution of species. This knowledge helps to facilitate the development of medications and in discovering basic relationships amongst species in the history of life.

**Primary Sequence Databases**

The International Nucleotide Sequence Database (INSDB) consists of the following Primary Sequence Databases.
1. DDBJ (DNA Data Bank of Japan)

2. EMBL-EBI Nucleotide DB (European Molecular Biology Laboratory-European Bioinformatics Institute)

3. GenBank (NCBI - National Centre for Biotechnology Information)

The International Nucleotide Sequence Database Collaboration (INSDC) is a long-standing foundational initiative that operates between DDBJ, EMBL-EBI and NCBI. INSDC covers the spectrum of data raw reads, though alignments and assemblies to functional annotation, enriched with contextual information relating to samples and experimental configurations.

The National Centre for Biotechnology Information (NCBI) is part of the United States National Library of Medicine (NLM), a branch of the National Institutes of Health. The NCBI is located in Bethesda, Maryland and was founded in 1988 through legislation sponsored by Senator Claude Pepper.

The NCBI houses a series of databases relevant to biotechnology and biomedicine. Major databases include GenBank for DNA sequences and PubMed, a bibliographic database for the biomedical literature. Other databases include the NCBI Epigenomics database. All these databases are available online through the Entrez search engine.

The DNA Data Bank of Japan (DDBJ) is a biological database that collects DNA sequences. It is located at the National Institute of Genetics (NIG) in the Shizuoka prefecture of Japan. It is also a member of the International Nucleotide Sequence Database Collaboration or INSDC. It exchanges its data with European Molecular Biology Laboratory at the European Bioinformatics Institute and with
GenBank at the National Centre for Biotechnology Information on a daily basis. Thus these three databanks contain the same data at any given time. DDBJ (DNA Data Bank of Japan) Centre collects nucleotide sequence data as a member of INSDC (International Nucleotide Sequence Database Collaboration) and provides freely available nucleotide sequence data and supercomputer system, to support research activities in life science.

The European Bioinformatics Institute (EMBL-EBI) is a centre for research and services in bioinformatics, and is part of European Molecular Biology Laboratory (EMBL). EMBL-EBI (European Molecular Biology Laboratory – European Bioinformatics Institute) provides freely available data from life science experiments, performs basic research in computational biology and offers an extensive user training programme, supporting researchers in academia and industry.

There is a difference between market-basket data and biological sequence data. In market-basket sequence data, available items list in the stores are larger than the purchased item list. But DNA sequences consists of only four nucleotide characters namely Adenine (A), Cytosine (C), Guanine (G) and Thymine (T). DNA sequence is a big linear chain made of only four characters A, T, C and G. So it is long list of only 4 items. Similarly protein is also of 27 characters. Thus bio-sequences are long in size and consist of limited item list (4 for DNA, 27 for Protein) [3] [4]. So the minimum support for DNA sequences FCP will be in terms of thousands whereas minimum support for market-basket FCP is very limited. The DNA sequences contain high minimum support sequences. So there is a considerable difference in referring, identifying FCPs in biological sequences data than market-basket sequence data.
1.5. MOTIVATION FOR THE RESEARCH WORK

“Nature is a tinkerer and not an inventor” (Jacob, 1977).

The accurate (exact patterns) identification of FCPs in DNA Sequences with a minimal time, poses a great challenge for the FCP mining algorithms. DNA sequences have exponential number of possible hidden patterns (especially short length patterns) leads to huge searching space and increases the number of patterns comparison. Hence, time and space complexity for mining the complete-set of FCPs are high. The other issues in mining the complete-set of FCPs are repeated scanning of huge database and usage of large number of intermediate tables for patterns comparison and required patterns extraction.

These issues motivate to design and development of enhanced algorithms for mining the complete-set of FCPs in DNA sequences.

1.6. SCOPE OF THE RESEARCH WORK

During the past two decades, many algorithms were developed and used to find FCPs met with success, but with varying levels of limitations also.

Few of those limitations are as given below:

- Scans the huge database multiple times
- Generates all possible set of sub-sequences (huge in size) instead of considering only the existing patterns in the db
- Inefficient for mining long sequential patterns
- A long pattern grow up from short patterns
- An exponential number of short patterns
- Huge length spanning tree is used to mine concatenated frequent DNA sub-sequences
• Construction of recursive mining from short length pattern to long length pattern

• This is very time consuming, because in a practical DNA sequence database, a sub-sequence may occur multiple times in the same sequence

• Huge number of intermediate tables are used to store and process prefixes/suffixes patterns in the DNA sequences

• Multiple scans of the database is needed to construct Prefix/Suffix patterns

This research work focuses on reduction of the above limitations and to design and develop “Enhanced algorithms to mine the complete-set of FCPs in DNA sequences”.

1.7. DEFINITION OF THE PROBLEM

Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified min_support threshold, mining the complete-set of Frequent Contiguous Pattern (FCP) is to find all of the frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min_support.

A sequence database (SDB) consists of ordered events/sequences recorded with or without concrete notation of time. A sequence \( \alpha = a_1, a_2, ..., a_n \) is called a contiguous sub-sequence of another sequence \( \beta = b_1, b_2, ..., b_m \), and \( \beta \) is a contiguous super-sequence of \( \alpha \), denoted as \( \alpha \subseteq \beta \) and the order of occurrence of items in sub-sequence \( \alpha \) must follow the order of occurrence of items in the super-sequence \( \beta \); It is said that \( \alpha \) is contained by \( \beta \). Repeatedly occurring consecutive subsequences / patterns in the SDB are called as Frequent Contiguous Patterns (FCP).
FCP is mathematically defined as below.

Let set $X = (i_1, i_2, ..., i_k)$ is called an item set of size $k$

$$D = \{(s_1, a_1), (s_2, a_2), ..., (s_n, a_n)\} \text{ is a SDB of size } n, \text{ where } s_i (1 \leq i \leq n) \text{ are sequence ids and } a_i (1 \leq i \leq n) \text{ are the sequences, consists of items in } X \text{ and length of each sequence is } m, \text{ then the set } F = \{b_1, b_2, b_3, ..., b_l\}, \text{ where } b_i, 1 \leq i \leq l \text{ is the contiguous sub-sequence of “D” and the occurrence of each } b_i \text{'s, is no less than min_support, an user-specified threshold, then the set } F \text{ is called FCPs.}

Mining the complete-set of FCP is the process to find all FCPs in a SDB “D”, where the occurrence frequency of each FCP is no less than min_support.

For eg, The Item set $X = (A, G, T, C)$ and SDB

$$D (s_i, a_i) = \{(1, GTG ATCG ACT ATTG),
                  (2, ATCG CTTC ATCG)
                  (3, CGTGAAGTATCG)
                  (4, ATTGTGTCGTG)
                  (5, GTGGCATTG)\}, \text{ 1 \leq i \leq 5}
$$

Length_4 FCPs in SDB “D” are \{ATCG, ATTG, CGT\} of min_support 2.

1.8. OBJECTIVES

The objectives of this research are as follow:

- Mining the accurate (Exact) FCPs of “length_n” with high min_support
- Mining the complete-set of FCPs by using efficient Pattern grouping with the help of heuristic approach and Permutation with Repetition (PR) concept
- Reduce the search space and intermediate tables by using derived mathematical formulae
- No repeated scanning of database (scanning the DB only once)
• Minimize Runtime and Cache/Buffer memory by reducing the number of patterns comparison

The research findings are validated by using Human Genome DNA sequence data of size 50 GB from NCBI (National Centre for Biological Information, USA) and with the following parameters.

• Effective usage of Cache memory
• Reduce the search space
• Reduce the repeated scanning of database
• Space complexity
• Time complexity

1.9. ORGANIZATION OF THESIS

Chapter 2 discusses detailed literature review related to this thesis on Sequential Pattern Mining and Frequent Contiguous Pattern mining algorithms and analyses their merits and demerits.

Chapter 3 discusses the proposed Heuristic and Hashing approach FCP algorithms. This chapter also analyses implementation results and findings with large DNA sequences.

Chapter 4 proposes a Permutations with Repetition (PR) based FCP algorithm called Sequential_PR_FCP and analyses the performance with large DNA sequences.

Chapter 5 proposes a Recursive Positions based FCP mining algorithm and discusses its merits and demerits.

Chapter 6 discusses the applications of proposed FCP algorithms in life sciences field like DNA-Drug Design and Genomic Functional Analysis Tools development.
Chapter 7 concludes with research contribution and the general mathematical models to mine Complete-Set of FCP for large DNA sequences.

### 1.10. CHAPTER SUMMARY

This chapter provides the basis of data mining concepts for Pattern Mining and introduction to DNA sequences. It discusses the importance of FCP mining in DNA sequences. This chapter also discusses the existing limitations of mining complete-set of DNA FCPs which motivates this research work. This chapter briefs the scope of the research work, problem definition and research objectives.