Recognizing DNA splice sites with the frequent pattern mining technique

Nittaya Kerdprasop and Kittisak Kerdprasop

Abstract—The completion of Human Genome Project in 2001 yields the entirety of human genetic information, or genome. A genome is organized in chromosomes and composed of thousands of genes, which are the heredity units of traits such as hair color and blood type. Genes in complex organisms such as primates and humans are composed of regions that code for protein, called exons, and non-coding regions, called introns. During the transcription from the DNA template for later translating into amino acid chain of protein structure, introns are to be removed and exons are then joined to form a continuous messenger-RNA strand. Splice sites are the junctions or borders between introns and exons. Accurate detection of splice sites from the fragments of DNA sequence is important to the success of gene prediction. Due to huge amount of genetic information in most genomes, computational techniques are essential for the interpretation and recognition of specific genetic sequences. In this paper, we propose a splice site prediction technique based on frequent pattern analysis. We apply association mining to each splice junction types, that is, exon/intron, intron/exon, and none of the two types. The frequent DNA patterns are then combined and prioritized with respect to their annotated confidence and support values. The final result of our method is a set of cascaded rules to be used for gene prediction. From the experimental results, our method can make a high recall prediction comparative to other classification-based methods. We also demonstrate computational improvement via a concurrency technique. Running time reduction is considerably observable.

Keywords—Gene expression, splice site prediction, DNA sequence, frequent pattern analysis.

I. INTRODUCTION

Proteins perform essential functions in most living organisms. The diverse functions of proteins include providing rigidity and mass in bones and tissues, forming enzymes for biological catalysts, and acting as special substances such as hemoglobin and insulin. Proteins are macromolecules comprised of the polymers of 20 different types of amino acids [18]. The amino acid composition of proteins is in turn determined by the DNA sequence, or a gene. Deoxyribonucleic acid (DNA) is the carrier of information to build proteins, and the ribonucleic acid (RNA) is involved in the biosynthesis of proteins. Relationship between DNA, RNA, and protein is called the central dogma of molecular biology [3] and can be illustrated in Fig. 1.

The basic constituents of DNA and RNA are nucleotides that are made of three chemical components: phosphate group, ribose sugar, and nitrogenous bases. DNA normally forms a single strand of nucleotides with a nitrogenous base as either adenine (A), guanine (G), cytosine (C), or uracil (U). RNA, on the contrary, is composed of the deoxy form of ribose and the four nitrogenous bases adenine, guanine, cytosine, and thymine (T). A single DNA strand is not stable; therefore, it normally appears in antiparallel double helix structure with the complementary pair A-T and C-G.

The base sequence, or gene, encodes genetic information to generate proteins. This information in DNA is transferred to the messenger RNA (or mRNA) during the process called transcription. The genetic flow followed by the translation process to build proteins from mRNA.

In prokaryotes (one-cell organisms with no nucleus, hence genetic materials disperse in the cell), genetic information is encoded continuously on a DNA strand. But in eukaryotes (organisms with one or more cells that have nucleus to encompass genetic materials), regions that code for proteins are interrupted by the non-coding regions. The non-coding regions in DNA are called introns (intervening sequences), whereas the coding regions are called exons (expresses sequences) [11]. During the transcription process of most eukaryotic genes, the primary RNA transcript needs additional modification step called splicing.

The splicing process involves the removal of introns by spliceosome and joining exons together to make one long continuous mRNA strand (Fig. 2). The spliced mRNA is then exported out of the cell nucleus and passed to the ribosome for translation to different kinds of amino acids. Ribosome translates code for protein synthesis by reading three consecutive bases, called codon. A specific codon encoded for the starting point of protein coding region is AUG (or ATG in DNA strand). During the translation process, tRNA associates
each codon with a unique amino acid to form a chain of polypeptide. The translation process stops when ribosome encounters one of the three stop codons: UAA, UAG, or UGA.

The borders of introns are called splice sites. The GT (or GU) end is called the donor splice site, and the AG end is called the acceptor splice site. In some rare cases, the GU-AG end is called the donor splice site and the AG end is called the acceptor splice site. DNA template is transcribed to the RNA strand, the beginning with AG at the 3' end. Since splicing occurs after a donor site is encountered, the donor site is GT. There are different kinds of introns, but the one involved in the splicing mechanism is known as a spliceosomal intron.

Nearly all spliceosomal introns conform to the GT-AG rule in which the first two nucleotides at the 5' end begin with GT and end with AG at the 3' end. Since splicing occurs after a DNA template is transcribed to the RNA strand, the beginning of the spliceosomal introns is GU, not GT.

The rest of this paper is organized as follows. Section 2 states the splice site recognition problem and reviews related work regarding this problem. Section 3 explains our prediction method that employs association mining technique. Section 4 shows experimental design and results. Section 5 concludes the paper.

II. DNA SPlice SITE RECOGNITION PROBLEM AND RELATED WORK

The splice site recognition problem can be formulated as a subproblem of gene prediction that aims at correctly recognizing gene from the given fragment of DNA sequence. The task of splice site prediction is to recognize the actual boundaries of the protein-coding regions in the DNA sequence. There are many computational techniques applied to tackle this problem. The direct method [6], [7], [10], [21] is to analyze the sequence to capture gene profile and identify specific features that can accurately predict the splice junctions. Researchers from the machine learning community prefer to attack this problem via a single or multiple classification learning algorithms [12], [13], [16].

In the early years of computational molecular biology, common learning methods used to predict gene position and structure are neural network and hidden Markov model [2], [17], [19]. These techniques are still in use by current researchers, for instance, Mubark and colleagues [15] applied the two techniques to predict the structure of hemoglobin (which is a protein component of red blood cells) from the given DNA sequence. Cristea and his team [4] also applied neural network on the reduced feature dataset to detect mutations in the genomic sequences.

Different classification schemes from the machine learning area are soon realized to be capable of producing a high accurate predictor. These classification techniques include the support vector machines [5], [12], [22], C4.5 [20], and Bayesian method [8].

Our approach to solve the splice site recognition problem is different from those appeared in the literature in that our predictor is built from the association analysis technique [1], not the classification ones. The advantage of the proposed technique is that the prediction model can contain nucleotides at arbitrary position, not necessarily be the contiguous base sequences.
III. PROPOSED METHOD TO SPICE SITE RECOGNITION AND PREDICTION

At the initial stage of our proposed method (named assoDNA), the training dataset with a mixture of exon/intron, intron/exon, and none of the two DNA sequence splice sites is separated into three subsets according to splice junction types. Each data subset is then processed through the same steps of frequent patterns and association analysis. The conceptual model and flow diagram of our method are depicted in Figs. 3 and 4, respectively.

Once the three data subsets are processed through the frequent pattern analysis method, the three sets of learning results (displayed as prediction rules) are finally combined and prioritized according to the confidence and support values. The proposed assoDNA method can be explained as follows.

Step 1: Initialization phase
Split the training dataset into three subsets according to the class value. Thus, we will get data of class exon/intron, data of class intron/exon, and data of class none.

Step 2: Generation of frequent patterns
Each data subset is processed through the following steps:
2.1 Set the given minimum support as minSup
2.2 Initialize R (a set of frequent patterns) to be empty, \( R = \emptyset \)
2.3 Build a candidate pattern \( P \) of length \( K \)
   \[ P = \land_K (L_i = B_j) \]
   where \( K \) starts from 1, \( i \in \{-30, \ldots, +30\} \), and \( B_j \in \{A,C,T,G,D,N,S,R\} \)
2.4 Select a pattern \( P \) with support \( \geq \text{minSup} \) to contain in a set \( S \)
2.5 Set \( R = R \cup S \)
2.6 If \( S = \emptyset \), then continue to step 3 else increment \( K \) and go back to step 2.3

Step 3: Confidence computation
3.1 Compute confidence value of every pattern \( P \) in \( R \), and annotate confidence value to every pattern
3.2 Sort \( P \) in descending order according to confidence value, for a tie then descending sort with respect to a support value

Step 4: Rule generation
4.1 Set the given minimum confidence as minConf
4.2 Generate association rules from every pattern \( P \) in \( R \) that has confidence \( \geq \text{minConf} \)

Step 5: Building predictor model
Combine rules from the process of every data subset and sort according their confidence and support values

Fig. 3 A conceptual model of splice site recognition

Fig. 4 A flow diagram illustrating the assoDNA method
Confidence and support are basic metrics [1] used in association analysis to delimit the number of association rules and size of the search space, respectively. The two metrics can be defined through the following example. Given the association pattern that nucleotide base $B_1$ appears at location $L_1$ and base $B_2$ appears at location $L_2$, then the prediction result is that this DNA sequence splice site is in category $C_1$. This prediction based on association pattern can be written as the following classification rule:

$$\text{IF } (L_1 = B_1 \land L_2 = B_2) \text{ THEN } (\text{splice-site} = C_1)$$

(1)

Note that for this specific problem splice site category, $C$, can either be the exon/intron junction, intron/exon junction, or none. Nucleotide symbol, $N$, can be one of the eight symbols: A, C, T, G, D, N, S, R, where the last four symbols represent ambiguity. D means nucleotide base can be either A, or G, or T. N is ambiguity among A or C or G or T. S is ambiguity among C or G, and R is ambiguity among A or G. The base location, $L$, is in the range -30 to +30.

The confidence of rule (1) is computed as:

$$\frac{\text{support}(L_1 = B_1 \land L_2 = B_2 \land \text{splice-site} = C_1)}{\text{support}(L_1 = B_1 \land L_2 = B_2)}$$

(2)

The value of support($L_1 = B_1 \land L_2 = B_2 \land \text{splice-site} = C_1$) is the number of times that the three events ($B_1$ appears at location $L_1$, $B_2$ appears at location $L_2$, and splice-site junction is of type $C_1$) occur together within the same DNA sequence in the training dataset. The support value of ($L_1 = B_1 \land L_2 = B_2$) is thus the frequency that both events co-occur in the training dataset.

We implement the assoDNA method with the Erlang functional language. Some major functions are displayed in Fig. 5.

**Fig. 5** Some main functions of the assoDNA program

```
main1() ->
    {AllInput,FNo,ThisClass} = input(),
    DB = myToSet(AllInput),
    Total = length(AllInput),
    (_,Per) = io:read(" input percent> "),
    [FNo,ThisClass, DB, Per] .

apriori(DB, Items, Min) ->
    C1=[ [from_list([X]), findSup(from_list([X]), DB) ] || X <- Items ],
    L1=[ [FS,Sup] || [FS,Sup] <- C1,Sup>=Min ],
    LkPrint=[ [to_list(FS), Sup,Sup/length(DB)*100] || [FS,Sup] <- L1 ],
    K = 2,
    LS = [FS || [FS,_] <- L1],
    aprioriLoopPar(L1, DB, LS, K, Min) .
```

The screenshot in Fig. 6 shows the total of eleven frequent patterns found by the assoDNA program on an intron/exon group when setting minimum support to be 50%. Each pattern is annotated with its occurrence frequency and percentage of support value. The pattern format is implemented as “base(location)”, for example, A(-2) means the base nucleotide A appears at location -2.

**Fig. 6** Screenshot of the assoDNA program

IV. EXPERIMENTAL DESIGN AND RESULTS

A. Dataset Preparation

The dataset used in this work primate splice-junction gene sequences available at the UCI repository of machine learning databases [14]. This dataset are taken from GenBank 64.1 containing 3,190 DNA sequences. Each sequence is a window of 60 DNA base pairs starting at position -30 and ending at position +30 corresponding to the splice site location. The splice junction can be either a junction between intron and exon (labeled as intron/exon), a junction between exon and intron (labeled as exon/intron), or no junction at all (labeled as none).

From the total of 3,190 DNA sequences, we split the data into two sets: training set (with 2,000 DNA sequences) and test set (with 1,190 DNA sequences). We try to conserve class distribution of both data subsets close to the original data as much as possible. Class distribution of training and test sets are summarized in Table 1. From the original 767 DNA sequences of class exon/intron, we split them by taking the first 500 DNA sequences to be in the training set and the remaining 267 sequences to be in the test set. The intron/exon and none classes of training and tests sets are prepared in the same manner.
Table 1. Class distribution in the training and test sets

<table>
<thead>
<tr>
<th>Classes</th>
<th>Training data</th>
<th>Test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>exon/intron</td>
<td>500 (25%)</td>
<td>267 (22.4%)</td>
</tr>
<tr>
<td>intron/exon</td>
<td>500 (25%)</td>
<td>268 (22.5%)</td>
</tr>
<tr>
<td>none</td>
<td>1,000 (50%)</td>
<td>655 (55.1%)</td>
</tr>
<tr>
<td>Total</td>
<td>2,000</td>
<td>1,190</td>
</tr>
</tbody>
</table>

B. Experimentation

We prepare a separate test set in order to make a fair comparison between a prediction model obtained from our method (assoDNA) and models from other learning techniques. Other machine learning methods [9] used in this experimentation include the C4.5 algorithm, naïve Bayes technique, instance based method with 10 nearest neighbors (1 and 5 neighbors yields worse performance), and support vector machine (algorithm SMO). We also tested the neural network algorithm on the training dataset, but it ran out of memory space. Class prediction identified by the model is compared against the real class of each DNA sequence in the test set.

The outcome of the prediction can be either correct, or incorrect. For the incorrect cases, a more specific analysis of incorrectness is described in a form of confusion matrix as shown in Table 2.

Table 2. Confusion matrix of the three-class prediction model

<table>
<thead>
<tr>
<th>Real class</th>
<th>Class prediction made by the model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>exon/intron</td>
</tr>
<tr>
<td>exon/intron</td>
<td>a</td>
</tr>
<tr>
<td>intron/exon</td>
<td>d</td>
</tr>
<tr>
<td>none</td>
<td>g</td>
</tr>
</tbody>
</table>

The variable ‘a’ means the number of DNA sequences with exon/intron splice sites that are actually predicted by the model as exon/intron. Variable ‘b’ is the number real exon/intron splice sites that are incorrectly predicted as intron/exon, whereas ‘c’ is the number of real exon/intron junctions that are incorrectly predicted as none. Other variables (‘d’ through ‘i’) can be interpreted in the same manner.

To evaluate the performance of the prediction model, we use the four measurement metrics: accuracy, precision, recall, and F-measure. The four measures are defined as:

\[
\text{Accuracy} = \frac{(a + e + i)}{(a + b + c + d + e + f + g + h + i)} \quad (3)
\]

\[
\text{Precision (or specificity)} = \frac{a}{(a + d + g)} \quad (4)
\]

\[
\text{Recall (or sensitivity)} = \frac{a}{(a + b + c)} \quad (7)
\]

\[
\text{F-measure (by class)} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (10)
\]

Experimental results regarding the performance comparison of our method against other classification-based learning methods are given by class in Tables 3-5. It can be seen from the results that our method is superior in its recall capability, especially for the class exon/intron. The proposed method is as good as others in its precision power.

Table 3. Prediction performance on recognizing exon/intron splice site

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>assoDNA</td>
<td>Accuracy</td>
</tr>
<tr>
<td></td>
<td>0.961</td>
</tr>
<tr>
<td>C4.5</td>
<td>0.936</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.966</td>
</tr>
<tr>
<td>Instance based (IB10)</td>
<td>0.801</td>
</tr>
<tr>
<td>Support vector machine</td>
<td>0.920</td>
</tr>
</tbody>
</table>

Table 4. Prediction performance on recognizing intron/exon splice site

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>assoDNA</td>
<td>Accuracy</td>
</tr>
<tr>
<td></td>
<td>0.961</td>
</tr>
<tr>
<td>C4.5</td>
<td>0.936</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.966</td>
</tr>
<tr>
<td>Instance based (IB10)</td>
<td>0.801</td>
</tr>
<tr>
<td>Support vector machine</td>
<td>0.920</td>
</tr>
</tbody>
</table>
### Table 5. Prediction performance on recognizing none of the splice site

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>assoDNA</td>
<td>0.961</td>
</tr>
<tr>
<td>C4.5</td>
<td>0.936</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.966</td>
</tr>
<tr>
<td>Instance based (IB10)</td>
<td>0.801</td>
</tr>
<tr>
<td>Support vector machine</td>
<td>0.920</td>
</tr>
</tbody>
</table>

Comparative results of different gene finding schemes in terms of precision and recall rates on recognizing DNA splice sites can graphically displayed in Figs. 6 and 7, respectively. It can be noticed that our method (assoDNA) produce high recall results on recognizing exon/intron and intron/exon borders. For the class of none of the two borders, our method has lower recall rate than naïve Bayes.

### C. Performance Improvement via Concurrency

To improve the computational performance of the proposed assoDNA method, we employ the concept of concurrent programming. Erlang is a functional language that provides facilities for designing and implementing concurrent program. Those facilities include the spawn, send, and receive primitives to handle concurrent programming. Concurrent processes in Erlang communicate through asynchronous message passing with dedicated memory space for each process, rather than a thread concept and shared memory. The example of transforming sequential assDNA program to be a concurrent assoDNA is showed in Fig. 8. Reduction in running time can be compared through the screenshots in Fig. 9 in which the last line on a upper screen is sequential running time (in a unit of microseconds), whereas the last line on a lower screen is concurrent running time. Time reduction in this example is around 46.29%.

```erlang
-module(assoDNA_par).
concurrent(P1, P2, P3) ->
  spawn(assoDNA_par, run, [self(), P1]),
  spawn(assoDNA_par, run, [self(), P2]),
  spawn(assoDNA_par, run, [self(), P3]),
  receive
    my_end -> ok
  end.
run(MasterID, InputL) ->
  R = main2(any, 3, InputL),
  file:delete("out.txt") ,
  AD = lists:last(R),
  [ADD | _] = AD,
  Rules = lists:sublist(R, length(R)-1),
  PrintRules = map(fun( {D, S, Per, Class} ) ->
    {to_Col3(notLast(D)), S, Per, transformBack(Class) }
   end, Rules),
  ADP = lists:map(fun(Data) ->
    {Data, checkRules(Data, Rules) }
   end, AD),
  ADPprint = map(fun({Data, V}) ->
    Predict = transformBack(V),
    {Data, [last(Data), Predict, mark(last(Data), Predict) ] }
   end, ADP),
  Predict = map(fun({F, S}) ->
    {to_Col3(notLast(F)), S}
   end, ADPprint),
  writeToFile(Predict),
  [_, Stop | _] = InputL,
  if Stop == 2 -> MasterID ! my_end ;
     true -> MasterID ! not_end
  end.
```

**Fig. 6** Precision comparisons of different recognition methods

**Fig. 7** Recall comparisons of different recognition methods

**Fig. 8** Program coding for running concurrent assoDNA
V. CONCLUSION

Splice site prediction from the fragment of DNA sequence is an extensively studied problem in computational molecular biology and bioinformatics. Splicing is the modification process that occurs during the transcription of gene expression, which is the process of transcribing a DNA template into mRNA sequence and then translating the mRNA to protein structure.

In organisms with cell nucleus, called eukaryotes, the transcription is not straightforward from DNA template to mRNA. It instead involves the intermediate step from DNA to pre-mRNA, then to mRNA. Splicing occurs at the stage of changing from pre-mRNA to the mRNA transcript. This change involves removing introns and then joining the exon parts together to form a continuous genetic sequence ready for the translation to a functional polypeptide chain of protein structure. To accurately predict the splice site of the DNA sequence fragment is an important task of gene prediction.

In this paper, we present a new method to splice site prediction from genomic sequence by means of association analysis. Association mining is unsupervised learning task that has been successfully applied to the marketing and business applications. It is rarely used in the area of bioinformatics. We thus devise a method to apply association mining technique to induce prediction model from the DNA sequence dataset. From the experimental results comparative to other classification learning techniques, we found that the prediction model based on association analysis can produce a significantly high recall prediction, whereas the precision rate is as good as other learning techniques.

The proposed method is however unable to generate frequent patterns when the support metric is set lower than 10%. This is because high memory consumption during the candidate pattern generation steps. We are planning to solve this problem in our future work.

ACKNOWLEDGMENT

This research has been funded by grants from the National Research Council of Thailand (NRCT) and the Thailand Research Fund (TRF, grant number RMU5080026). The DEKD Research Unit has been supported by Suranaree University of Technology.

REFERENCES


Nittaya Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. Her current address is School of Computer Engineering, Suranaree University of Technology, 111 University Avenue, Nakhon Ratchasima 30000, Thailand, e-mail: nittaya@sut.ac.th.

Kittisak Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, USA, in 1999. She is a member of ACM and IEEE Computer Society. Her research of interest includes Knowledge Discovery in Databases, AI, Logic Programming, Deductive and Active Databases.