Identification of Exon-intron Boundaries by Integration of Base-oriented Genetic Programming and Statistical Heuristics

Kunihito YAMAMORI
Univ. Miyazaki
Miyazaki 889-2192

Yuji FUJITA
Mitsubishi Space Software
Ibaraki 305-0032

Masaru AIKAWA
Univ. Miyazaki
Miyazaki 889-2192

Ikuo YOSHIHARA
Univ. Miyazaki
Miyazaki 889-2192

Abstract

Genetic Programming (GP) is one of the promising methods to detect exon-intron boundaries from DNA sequences. In conventional method, two- or four-bit binary codes represent four bases, and each bit of codes are used as input to the identification model. These bit-oriented GP is simple, but it is difficult to know which bases are key to identify the boundary. Here we develop a novel base-oriented GP which can directly use bases as inputs. Moreover, we integrate the model generated by base-oriented GP with heuristics that is defined as bias of appearance frequencies on particular loci around "GT" and "AG" extracted from about 8,000,000 DNA sequences. Simulation results show that our method can improve the accuracy about 10% on the " AG " boundary identification.

1 Introduction

A gene in DNA consists two parts; one is exon and the other is intron. Exon encodes information how to synthesize proteins. Intron exists between exons, and function of intron is not clear still yet. Now it is a very important subject to automatically extract exon regions and intron regions from DNA to diagnose some kinds of diseases or to create new medicine. However, it is very difficult to analyze gene data because it is too many and still increasing day after day.

To automatically extract exons and introns from DNA sequences by computers, Some methods have been investigated until now. Kamimai[1] had proposed multi-modal neural network and succeeded to improve accuracy of identification. Ohta[2] had developed a combination of GA (Genetic Algorithm) and GMDH (Group Method of Data Handling), and showed good performance of identification. In these method all bases in a region including focusing bases are encoded to two- or four-bits binary codes, and each bit is used as inputs to the identification model. These binary encoding lead indistinctness which bases are the keys to identify boundaries.

Our aim is to develop a method that can directly use the bases as inputs to the model, and it also achieves accurate exon extraction from DNA sequences. So we modify Genetic Programming (GP) to be able to handle the base characters directly. And we combine the GP and statistical heuristics extracted from 8 million DNA sequences.

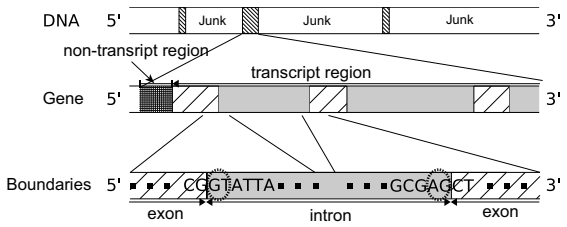

Figure 1: DNA and gene.

2 DNA sequences and heuristics

2.1 Decision of input region

Figure 1 illustrates that most 5'-end and 3'-end of introns has the bases "GT" and "AG"[3]. Since the sequences "GT" and "AG" also appear in exons and introns, it is impossible to identify boundaries by this rule. On the other hand some bases often appear in specific locus around the boundary. We utilize this bias of appearance to identify boundaries between exons and introns.

Human DNA sequences was obtained from NCBI[1] database, then ±200 bases around "GT" or "AG" are extracted. The number of extracted sequences is as follows;

- **Sequences with "GT"**
  - Boundary data: 26,046
  - Non-boundary data: 7,610,718
- **Sequences with "AG"**
  - Boundary data: 22,633
  - Non-boundary data: 8,481,368

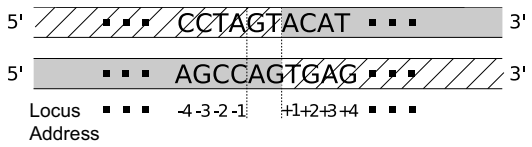

Figure 2: Addressing for loci.

Here $P(b, X)$ denotes the appearance probability of the base 'X' at the locus b, and the address of locus b is defined as shown in Figure 2. The $P(b, X)$ only from sequences including boundary is described as $P_B(b, X)$, and that without boundary is also described as $P_N(b, X)$. Then the difference $P_d(b, X)$ between $P_B(b, X)$ and $P_N(b, X)$ is calculated as Equation (1).

$$P_d(b, X) = P_B(b, X) - P_N(b, X), \tag{1}$$

here $b = \{-200, -199, \cdots, +200\}$ and $X \in \{A, G, C, T\}$.

From the bias of $P_d(b, X)$, we decide to use 20 bases to identify "GT" boundary (10 bases for 5'-side and 3'-side), and 37 bases to identify "AG" boundary (30 bases for 5'-side, 7 bases for 3'-side)[4].

Fujita[4] also said that the appearance frequencies of specific bases at some loci are different between boundaries and non-boundaries. So we define heuristics as the combination of the base and the locus with high bias in appearance frequency. These combination of specific loci and bases is integrated with GP by the following two method.

2.2 Majority method

In this way $P_d(b, X)$ is used as heuristics and combined through the majority operator into the model generated by GP. Following steps shows how to combine the model and the heuristics.

Step.1 Calculation of $P_d(b, X)$.

Step.2 Four combinations of b and X with the highest $|P_d(b, X)|$ are selected as heuristics.

Step.3 Four sub-trees consisting of a terminal node and a comparing node explained in Section 3 are constructed. The comparing node has $Eq(X, I)$ if the bias of appearance frequency at the locus b is positive. Otherwise it has $\overline{Eq(X, I)}$.

Step.4 Four sub-trees and an individual generated by GP is combined through the tree-fifth majority node $M(3/5)$.

All individuals combine with the same heuristics, and the fitness is evaluated by the output of the final $M(3/5)$. The same sub-trees as the heuristics never appear in individuals.

[1]National Center for Biotechnology Information

2.3 Embedding method

In this method a couple of bases are used as heuristics. These heuristics are called as 2-tuple heuristics. The 2-tuple heuristics are found from the ±200 bases around the focusing "GT" and "AG", then 30 2-tuple heuristics will be combined with the model by the following steps.

Step.1 The appearance frequency $P_B^2(b_1, b_2, X_1, X_2)$ and $P_N^2(b_1, b_2, X_1, X_2)$ are calculated from boundary data and non-boundary data, respectively. Here b_1 and b_2 is address of locus, and X_1 and X_2 denote bases.

Step.2 Calculation of $P_d^2(b_1, b_2, X_1, X_2) = |P_B^2(b_1, b_2, X_1, X_2) - P_N^2(b_1, b_2, X_1, X_2)|$.

Step.3 30 combinations of (b_1, X_1) and (b_2, X_2) with the largest $P_d^2(b_1, b_2, X_1, X_2)$ are selected as the 2-tuple heuristics.

Step.4 Two sub-trees are constructed; a terminal node b_1 and a comparing node $Eq(b_1, X_1)$, and a terminal node b_2 and a comparing node $Eq(b_2, X_2)$.

Step.5 The comparing node with "AND" operator integrates two sub-trees if $P_d^2(b_1, b_2, X_1, X_2) > 0$. Otherwise they are integrated by a node with "NAND" operator.

These 2-tuple heuristics are inserted into individuals at the following processes.

Insertion into initial individuals: 30 individuals are randomly selected and a part of individual is exchanged to a 2-tuple heuristics at the generation of initial population.

Insertion at mutation process: A part of individual is exchanged to 2-tuple heuristics at the mutation process.

3 Genetic operations

3.1 Operators

In this research we directly use the base characters themselves as inputs to GP. Figure 3 shows an example of individual to identify boundaries generated by GP. Fig.3 shows that terminal nodes keep the address of locus, and a non-terminal node is either a logic operator or a comparing operator. A node with a logic operator calls as a logic node, and a node with a comparing operator also calls a comparing node. Equation (2) says that the comparing operator

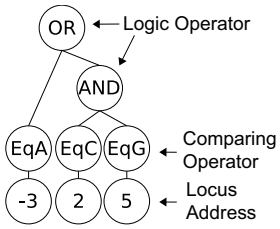

Figure 3: An example of individual.

Table 1: Logic operators in the model.

#s of arguments	logic operator
1	NOT
2	AND OR XOR NAND NOR XNOR
3	M(2/3) Two-third majority NM(2/3) Negative of M(2/3)

returns 1 if the input I from the terminal node is as same as the target base character X.

$$Eq(X, I) = \begin{cases} 1 & X = I, \\ 0 & X \neq I,\ X, I \in \{A, G, C, T\} \end{cases} \quad (2)$$

Table 1 shows available logic operators in logic nodes. Since an output of a terminal node is a character and inputs to a logic operator have to be boolean, a comparing node is coupled with a terminal node. These nodes are arranges as a tree for a boundary identification model.

3.2 Genetic operations

To make models, GP performs the following genetic operations.

Initialization: It randomly generates terminal nodes, logic nodes and comparing nodes, then arranges them as individuals.

Crossover: It randomly selects two individuals then exchange a part of individual (sub-tree) each other. So two children are generated by this operation. This operation does not allow some kinds of sub-tree exchanging because a terminal node must combine with a comparing node. The combinations of exchangeable nodes are completed in Table 2. In Table 2, L. node, C. node and I. node denote a logic node, a comparing node and an input node, respectively.

Fitness evaluation: Equation (3) defines the fitness.

$$fitness = \sum_{k=1}^{T} Z_k, \quad (3)$$

$$Z_k = \begin{cases} 1, & Y'_k = Y_k, \\ 0, & Y'_k \neq Y_k, \end{cases}$$

Table 2: Available combination at crossover operation.

	L. node	C. node	I. node
L. node	OK	OK	NG
C. node	OK	OK	NG
I. node	NG	NG	OK

where Y_k is 1 when the identification result by model says true (boundary) and Y'_k is also 1 when the k-th sequence is actually boundary. So $\sum_{k=1}^{T} Z_k$ means the sum of the correctly identified sequences.

Mutation: This operation randomly exchange the operator in a node to the other one with the same number of arguments if the selected node is a logic node. If a terminal node is selected, the other locus is employed. When a comparing node is selected, the other comparing operation take place of original one.

Selection: roulette strategy and elite preserving are employed.

4 Experiments and discussions

4.1 Parameters

We make eleven data sets from the sequences mentioned in Section 2, then a data set is used for model generation, and the others are used for model evaluation. No duplication exists among the data sets. The parameters of GP for simulations are as follows;

- Numbers of individuals: 100
- Maximum nodes in a model: 200
- Crossover ratio: 100%
- Mutation ratio: 1%
- Maximum generations: 4,000

The number of simulations are 10 on each data set for model evaluation.

4.2 Measures for evaluation

Sensitivity S_n and *Specificity* S_p defined by Equation (4) and Equation (5) evaluate accuracy.

$$S_n = \frac{N_c}{B} \times 100 \ \ (\%), \quad (4)$$

$$S_p = \frac{N_c}{N_c + N_m} \times 100 \ \ (\%), \quad (5)$$

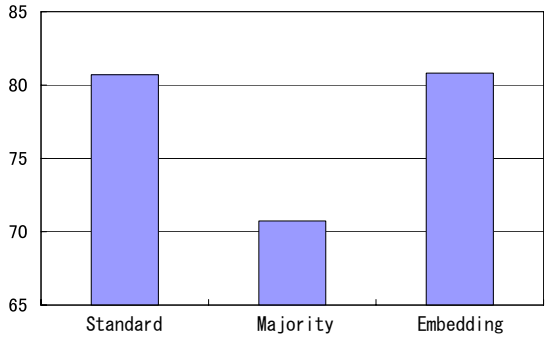

Figure 4: S_n by each method on "AG" boundaries.

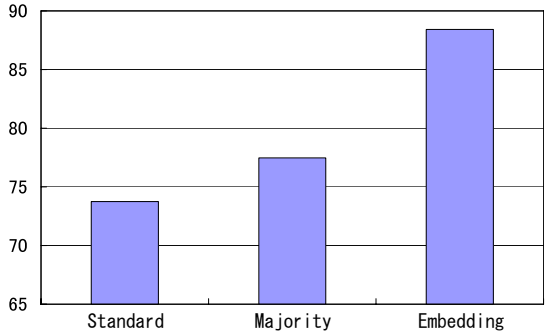

Figure 5: S_p by each method on "AG" boundaries.

where B denotes the number of boundary data, N_c and N_m denote the number of correctly identified boundaries and miss identified non-boundaries, respectively.

4.3 Results and discussions

Figure 4 and Figure 5 show the average S_n and S_p for verification sequences. Here we only show the result on "AG" boundaries because relatively poor results was reported in previous works[2]. In Fig.4 and Fig.5, "Standard" means the result by GP without heuristics.

In Figure 4, S_n by majority method falls about 10% than that of standard GP, it means the model combined with the majority operator often ignore the boundaries. It is because the answer from the model generated by GP and answers from the heuristics sometimes compete and cancel each other. Figure 5 illustrates that the GP with 2-tuple heuristics improves S_p about 10% than that by standard GP. It means that the embedding method can detect more boundaries than those by the other methods, and less non-boundaries are detected as boundaries.

Experimental results show that embedding method with 2-tuple heuristics improves accuracy of boundary identification with keeping boundary detection.

5 Conclusions

Automatic extraction of exon regions from DNA sequences is very difficult because the sequences have too much variety, and the number of sequences is also too huge. In particular, the identification of "AG" boundaries was not accurate by previous works because features to identify "AG" boundaries dispersedly exist in sequences.

In this research we develop a novel method to identify boundaries. At first, we decide the appropriate region for boundary identification from the bias of appearance frequencies on bases at each locus. The analysis reveals that ±10 bases are enough for "GT" boundaries, and 37 bases for "AG" boundaries. Then we think out a novel base-oriented identification model for GP that can directly handle the character of bases. Finally we compare the two method to integrate the model by GP and statistical heuristics. The embedding method achieved about 10% accuracy improvement to detect "AG" boundaries.

To achieve more accurate identification, the model considering solid structure of protein will be remaining as future works. Furthermore faster processing is also needed.

Acknowledgments

A part of this research is supported by Grant-in-Aid for Scientific Research #17700239.

References

[1] Y. Kamimai, I. Yoshihara, K. Yamamori and M. Yasunaga, "Methods to Extract Exon Regions from DNA Sequence – Development of Multi-modal Neural Network –", *Fuzzy, Artificial Intelligence, Neural Networks and Computational Intelligence (FAN'02)*, pp. 385–390 (2002).

[2] T. Ohta, I. Yoshihara, K. Yamamori and M. Yasunaga, "GP-based Method for Identifying Exon Region in DNA Sequences", *Proc. 4th Asia-pacific Conference on Simulated Evolution and Learning (SEAL'02)*, pp. cr1333(CD–ROM) (2002).

[3] M. B.Shapiro and P. Senapathy, "RNA splice junctions of different classes of eularyotes: sequence statistics and functional implications in gene expressions", *Neucleic Acids Research*, Vol. 15, pp. 7155–7175 (1987).

[4] Y. Fujita, K. Yamamori, I. Yoshihara and M. Aikawa, "Generation of Character-base Model by Genetic Programming to Identify Exon-Intron Boundaries", *Tech.Report IEICE (CAS2005-75)*, Vol. 105, No. 503, pp. 1–6 (2006) (In Japanese).