Design of DNA Sequencing Chain-Termination method using Supervised Machine Learning

[1]S.Satyanarayana, [2] T.Anuradha, [3]G.Sridevi, [4]V.Tata Rao
[1]Professor, [2,3,4] Assistant Professor
Department of Computer Science and Engineering,
Raghu Engineering College, Visakhapatnam, INDIA

Abstract:
The term DNA sequencing refers to methods for determining the order of the nucleotides bases adenine,guanine,cytosine and thymine in a molecule of DNA. The first DNA sequence were obtained by academic researchers, using laboratories methods based on 2- dimensional chromatography in the early 1970s. There are various DNA sequencing methods. In this paper we proposed the implemenation of Chain Termination Method using Supervised Machine Learning Algorithm. We design and implement DNA template, DNA primer, a DNA polymerase.

Index Terms
Categories and Subject Descriptors: **[Machine Learning]: Supervised Machine Learning, [DNA]: Chain Termination Method, [Python]: Implementation of Algorithm**

General Terms: **Algorithms, nucleotides, sequence, complementary sequence, file operations**

Keywords: **Python, tkinter and canopy**

I. INTRODUCTION

DNA sequencing enables us to perform a thorough analysis of DNA because it provides us with the most basic information of all: the sequence of nucleotides. With this knowledge, for example, we can locate regulatory and gene sequences, make comparisons between homologous genes across species and identify mutations. Scientists recognized that this could potentially be a very powerful tool, and so there was competition to create a method that would sequence DNA. Then in 1974, two methods were independently developed by an American team and an English team to do exactly this. The Americans, lead by Maxam and Gilbert, used a "chemical cleavage protocol", while the English, lead by Sanger, designed a procedure similar to the natural process of DNA replication. Even though both teams shared the 1980 Nobel Prize, Sanger's method became the standard because of its practicality (Speed, 1992).

Sanger's method, which is also referred to as dideoxy sequencing or chain termination, is based on the use of dideoxynucleotides (ddNTP's) in addition to the normal nucleotides (NTP's) found in DNA. Dideoxynucleotides are essentially the same as nucleotides except they contain a hydrogen group on the 3' carbon instead of a hydroxyl group (OH). These modified nucleotides, when integrated into a sequence, prevent the addition of further nucleotides. (Speed, 1992).This occurs because a phosphodiester bond cannot form between the dideoxynucleotide and the next incoming nucleotide, and thus the DNA chain is terminated.

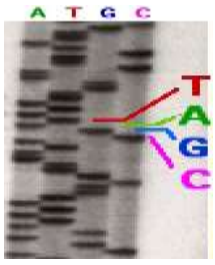

Fig. 1: A Radioactively Labeled Sequencing Gel

II. THE METHOD:

Before the DNA can be sequenced, it has to be denatured into single strands using heat. Next a primer is annealed to one of the template strands. This primer is specifically constructed so that its 3' end is located next to the DNA sequence of interest. Either this primer or one of the nucleotides should be radioactively or fluorescently labeled so that the final product can be detected on a gel (Russell, 2002).

Once the primer is attached to the DNA, the solution is divided into four tubes labeled "G", "A", "T" and "C". Then reagents are added to these samples as follows:

guanine "G" tube: all four dNTP's, ddGTP and DNA polymerase

adenine "A" tube: all four dNTP's, ddATP and DNA polymerase

thymine "T" tube: all four dNTP's, ddTTP and DNA polymerase

cytosine "C" tube: all four dNTP's, ddCTP and DNA polymerase

III. THE BASIC COMPONENTS:

Any of standard programming language components(API). We are proposing Python as standard language for Machine Learning components. It is proposed to use predefined and user defined components below:

Python built-in component(s):

- tkinter.filedialog
- Canopy as IDE

User defined component(s):

- Analysis 1 module – a1.py
- Analysis 2 module – a2.py
- Driver module – a2_driver.py

Fig. 2: Canopy IDE

IV. THE ALGORITHMS:

For DNA sequence processing the following data science algorithms are used:

Analysis module#1:

Step 1: Start
2: Get length of dna
3: Compare lengths of two dna's(1 and 2)
4: Count the nucleotides
5: Check two dna's contains sequence
6: Check whether valid dna or not
7: Inserting new dna sequence
8: Get the complement
9: Get complementary sequence
10: Stop

Analysis module#2:

Step 1: Start

2: Check whether word is valid DNA or not
3: Make a DNA sequence string from rows
4: Make a DNA sequence string from columns
5: Check whether data board contains DNA sequence
6: Calculate score
7: Count DNA words on the board
8: Read from file
9: Read board
10: Stop

Each of the above two modules contains various sub-algorithms listed below:

a) get_length() b) is_longer() c) count_nucleotides() d) contains_sequence() e) is_valid_sequence()
f) insert_sequence() g) get_complement() h) get_complementary_sequence()

V. THE SIMULATION RESULTS AND DISCUSSION:

RESULTS#1:

```
def get_length(dna):
    return len(dna)


def is_longer(dna1, dna2):
   return len(dna1)>len(dna2)


def count_nucleotides(dna, nucleotide):
   count=0
   for i in dna:
      if nucleotide==i: #when found
         count+=1
   return count


def contains_sequence(dna1, dna2):
   return dna2 in dna1

def is_valid_sequence(str):
            count=0
            for ch in str:
                        if ch!='A' and ch!='T' and ch!='C' and
ch!='G':
                                    return False
            return True

def insert_sequence(st1, st2, pos):
            st=st1[:pos] #new string st with slice of st1
            st+=st2
            st+=st1[pos:] #and second slice

            return st
```

```
def get_complement(str):
            if str=='A':
                        return 'T'
            elif str=='T':
                        return 'A'
            elif str=='C':
                        return 'G'
            elif str=='G':
                        return 'C'
def get_complementary_sequence(str):
            st=""
            for i in range(len(str)-1,-1,-1): #reverse loop
                        st+=str[i] #concatenate
            return st #return it
```

RESULTS#2:

```
def is_valid_word(wordlist, word):
   if word in wordlist:
      return True
```

```
def word_score(word):
   if len(word)>=10:
      return len(word)*3
```

```
    else:
        return False


def make_str_from_row(board, row_index):
    st=""
    for ch in board[row_index]:
        st=st+ch
    return st


def make_str_from_column(board, column_index):
    st=""
    for row in range(0,len(board)):
        st+=board[row][column_index]
    return st


def board_contains_word_in_row(board, word):
    for row_index in range(len(board)):
        if word in make_str_from_row(board, row_index):
            return True

    return False


def board_contains_word_in_column(board, word):
    for row_index in range(len(board)):
        for col_index in range(len(board[row_index])):
            if word in make_str_from_column(board, col_index):
                return True

    return False

def board_contains_word(board, word):
    if board_contains_word_in_row(board,word) and
board_contains_word_in_column(board,word):
        return True
    else:
        return False
```

```
    elif len(word)>=7:
        return len(word)*2
    elif len(word)>=3:
        return len(word)*1
    else:
        return 0

def update_score(player_info, word):
    player_info[1]+=word_score(word) #call above method and
add score


def num_words_on_board(board, words):
    count=0
    for word in ".join(words[count]): #get each word
        if board_contains_word(board,word)==True: #when
found
            count+=1 #increment count

    return count #finally return count

def read_words(words_file):
    words=list() #create empty list
    for word in words_file: #read each word
        word=word.strip()
        words.append(word) #add each word
    return words

def read_board(board_file):
    lst_words=list() #empty list
    for line in board_file: #read each line
        line=line.strip()
        lst_words.append(list(line)) #split line to sublist and add to
lst_words

    return lst_words #finally return list of list of words
```


VI. ACKNOWLEDGMENT

We are grateful for the sincere advice and care by Andhra University, Dept of Computer Science and Systems Engineering and other superiors in achieving this research paper. The authors would like to thank Raghu Engineering College, Visakhapatnam for their infrastructural support. Our gratitude goes to the people, who previously succeeded in implementation of DNA sequencing under supervised learning and making it available for further development.

REFERENCES

[1] Artem E. Men, Peter Wilson, Kirby Siemering, and Susan Forrest, Sanger DNA Sequencing: WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, ISBN: 978-3-527-32090-5

[2] Veronique G. LeBlanc 1,2 and Marco A. Marra, Next-Generation Sequencing Approaches in Cancer: Where Have They Brought Us and Where Will They Take Us? : ISSN 2072-6694, www.mdpi.com/journal/cancers

[3] Lilian T. C. Franc : a 1 , Emanuel Carrilho 2 and Tarso B. L. Kist, A review of DNA sequencing techniques

[4] R.D. Fleischmann et al., "Whole-Genome Random Sequencing and Assembly of H. Influenzae," Science, Vol. 269, No. 5,223, 1995, pp. 496–512

[5] 2. J. Weber and W. Myers, "Human Whole Genome Shotgun Sequencing," Genome Research, Vol. 7, No. 5, 1997, pp. 401–409

[6] L. Rowen, B.F. Koop, and L. Hood, "The Complete 685-Kilobase DNA Sequence of the Human Beta T cell Receptor Locus," Science, Vol. 272, No. 5,269, 1996, pp. 1755–1762.

[7] J.C. Venter, H.O. Smith, and L. Hood, "A New Strategy for Genome Sequencing," Nature, Vol. 381, No. 6,581, 1996, pp. 364–366.

[8] Identify Key Sequence Features to Improve CRISPR sgRNA Efficacy, LEI CHEN 1 , SHAOPENG WANG 2 , YU-HANG ZHANG 3 , JIARUI LI 2 , ZHI-HAO XING 3 , JIALIANG YANG 4 , TAO HUANG 3 , AND YU-DONG CAI, 2169-3536 2017 IEEE. Translations and content mining are permitted for academic research only.

[9] Mengye Lu, Shuai Liu, Arun Kumarsangaiah, Yunpeng Zhou, Zheng Pan, Yongchun Zuo : Nucleosome Positioning with Fractal Entropy Increment of Diversity in Telemedicine, DOI 10.1109/ACCESS.2017.2779850, IEEE Access

[10] Youxi Wu, Yao Tong, Xingquan Zhu, Senior Member, IEEE , and Xindong Wu,Fellow: NOSEP: Nonoverlapping Sequence Pattern Mining With Gap Constraints, 2168-2267 - 2017 IEEE