CSIT (March 2013) 1(1):9–28
DOI 10.1007/s40012-012-0001-1

ORIGINAL RESEARCH

Genome data classification based on fuzzy matching

Nagamma Patil · Durga Toshniwal · Kumkum Garg

Received: 18 June 2012 / Accepted: 13 August 2012 / Published online: 13 October 2012

Abstract Genomic data mining and knowledge extraction is an important problem in bioinformatics. Some research work has been done on unknown genome identification and is based on exact pattern matching of *n*-grams. In most of the real world biological problems exact matching may not give desired results and the problem in using *n*-grams is exponential explosion. In this paper we propose a method for genome data classification based on approximate matching. The algorithm works by selecting random samples from the genome database. Tolerance is allowed by generating candidates of varied length to query from these sample sequences. The Levenshtein distance is then checked for each candidate and whether they are *k*-fuzzily equal. The total number of fuzzy matches for each sequence is then calculated. This is then classified using the data mining techniques namely, naive Bayes, support vector machine, back propagation and also by nearest neighbor. Experiment results are provided for different tolerance levels and they show that accuracy increases as tolerance does. We also show the effect of sampling size on the classification accuracy and it was observed that classification accuracy increases with sampling size. Genome data of two species namely Yeast and *E. coli* are used to verify proposed method.

N. Patil (✉) · D. Toshniwal
Department of Electronics and Computer Engineering,
Indian Institute of Technology, Roorkee, India
e-mail: nagammapatil@gmail.com; parildec@iitr.ernet.in

D. Toshniwal
e-mail: durgafec@iitr.ernet.in

K. Garg
Department of Computer Science & Engineering,
Manipal University, Jaipur, India
e-mail: kumkum.garg@jaipur.manipal.edu

Keywords Bioinformatics · Soft computing · Genome data · Data mining · Approximate pattern matching · Exact matching

1 Introduction

Bioinformatics [1–3], has emerged as a forefront research area in the recent past since biological data is accumulating at an accelerated rate. In particular, the number and sizes of genome databases have grown rapidly over the last few years. One of the most important problems is automatically determining the group to which a previously unseen genome sequence belongs [4].

Classifying organisms from its genomic database into groups within a taxonomical hierarchy has several applications which include specific identification of any unknown organism, study of evolutionary characteristics, and study of mutual relationship existing between organisms [5]. Currently more than a million organisms have been discovered, but a large number are yet to be discovered. Any systematic study on an organism can be done only when it is identified to be in a particular group. Thus genome identification finds wide application in evolutionary studies of organisms. Classification and species identification have also been associated with practical applications such as bio-diversity studies [6], forensic investigations [7] and food and meat authentication [8], to name a few.

Pattern matching can be considered as either exact matching or approximate matching for sequential data [9]. In exact sequence pattern matching problems, we aim to find a substring in text T that is exactly the same as the searching pattern P. In real world biological applications, most of the sequences are "similar" instead of exactly the

same. Most fundamental operations like repeat pattern mining, similarity between two sequences, sequence alignment, etc., can be modeled as searching for given "patterns" in a "text." However, exact searching is of little use for this application, since the patterns rarely match the text exactly. Thus searching in sequence repositories often requires going beyond exact matching to determine the sequences which are similar [10, 11]. This gave a motivation to "search allowing errors" or approximate match. Approximate matching/fuzzy matching is the finding of the most similar match of a particular pattern within a sequence.

The existing approaches extract dinucleotide composition or trinucleotide composition from the sequences using exact pattern matching method and are used as features for classifier. In most of the real world biological problems exact matching may not give desired results because sequences are similar and the patterns rarely match the sequences. The problem which can appear in using *n*-grams is exponential explosion. It is clear that many of algorithms with *n*-grams are computationally too expensive [4].

In this paper identification of organism from its genomic database is considered. In the present work, we propose an approach for genome identification based on approximate pattern matching. Since genome data is very huge, we sample the data into different sizes. Given a database of randomly selected samples of genomic sequences, our proposed work includes two algorithms viz. algorithm for finding fuzzy occurrences based on Levenshtein distance and algorithm for finding total number of fuzzy matching patterns by varying candidate length so as to allow both positive and negative tolerance from the genome data sequences. These fuzzy matching patterns are used as features for a classifier. Since, candidate length is varied so as to allow both positive and negative tolerance, the length of subsequences and number of subsequences that are used as feature for classifier also changes. Classification has been done using data mining techniques namely, Naïve Bayes, support vector machine, backpropagation and also by nearest neighbor. Experimental results are reported for 100 randomly selected samples (size varying from 2,000 to 10,000 bp) from each of complete genome data of Yeast and *E. coli*. To extract fuzzy matching patterns from genome sequences, we used query of length 10 and allowed tolerance from 10 to 70 %. The proposed model is tested separately for fuzzy matching patterns extracted with each of the fault tolerance and the classification accuracies are monitored. The experimental results vary according to the tolerance allowed as well as according to sampling/sequence size.

The article is arranged as follows. In Sect. 2 we give some background information. Related work is explained in Sect. 3. In Sect. 4, proposed approach is explained. The experimental results obtained by using genome data of two species namely Yeast and *E. coli* are explained in Sect. 5. Conclusion section summarizes the results.

2 Background

2.1 Relevant terms

2.1.1 Genome

A genome is the complete genetic material of an organism. Its size is generally given as its total number of base pairs [12].

2.1.2 Base pair

A base pair consists of two nitrogenous bases (adenine and thymine or guanine and cytosine) held together by weak bonds. Two strands of DNA are held together in the shape of a double helix by the bonds between base pairs [13].

2.1.3 Base sequence

Base sequence is the order of nucleotide bases in a DNA molecule [13].

2.1.4 Nucleotide

Nucleotide is a subunit of DNA or RNA consisting of a nitrogenous base (adenine, guanine, thymine, or cytosine in DNA; adenine, guanine, uracil, or cytosine in RNA), a phosphate molecule, and a sugar molecule (deoxyribose in DNA and ribose in RNA). Thousands of nucleotides are linked to form a DNA or RNA molecule [13].

2.1.5 n-Gram

The *n*-gram is a subsequence composed of *n* characters, extracted from a larger sequence. For a given sequence, the set of the *n*-grams which can be generated is obtained by sliding a window of *n* characters on the whole sequence. This movement is carried out character by character. With each movement, a subsequence of *n* characters is extracted. This process is repeated for all the analysed sequences. The *n*-gram can be represented in binary form [14, 15] or either in dinucleotide [16] or trinucleotide [17, 18] frequencies.

Example: consider the generation of 3-g and representing them in binary form, from the following two sequences

Seq1: AVADEK
Seq2: QAVALGYVS

Table 1 *n*-Gram based sequence encoding in binary form

	AVA	VAD	ADE	DEK	QAV	VAL	ALG	LGY	GYV	YVS
Seq1	1	1	1	1	0	0	0	0	0	0
Seq2	1	0	0	0	1	1	1	1	1	1

Table 2 *n*-Gram based sequence encoding in dinucleotide frequencies

	AA	AC	AT	AG	CC	CA	CG	CT	TT	TA	TC	TG	GG	GA	GC	GT
Seq1	2	0	2	0	0	0	0	0	0	0	1	1	0	1	0	0

For $n = 3$, a total of 11 motifs are extracted from seq1 and seq2 by the sliding window procedure. Sequence1 results in 5 motifs, i.e., AVA, VAD, ADE and DEK while sequence 2 gives 7 motifs: QAV, AVA, VAL, LGY, GYV and YVS. Out of these 11 motifs, 10 are distinct (the motif AVA is repeated in both the sequences). These 10 distinct motifs are used as attributes/features to construct a binary table where each row corresponds to a sequence. The presence or absence of an attribute in a sequence is denoted by 1 or 0 as shown in Table 1.

2.1.6 Dinucleotide composition

DNA sequences are usually long sequences consisting of only four characters: A, T, C and G. The dinucleotide composition is the frequencies of "AC", "TG", "AA", "TT"… set of 16 subsequences of length two [16].

Example: consider the seq1 represented as AATGAATC, the 2-g that can be generated from the sequence are AA, AT, TG, GA, AA, AT, TC. Hence the dinucleotide frequencies for the example sequence can be represented as in Table 2.

2.1.7 Data mining

Data mining, or knowledge discovery from data refers to the process of extracting interesting, non-trivial, implicit, previously unknown and potentially useful information or patterns from data [19]. Mining of sequence data has many real world applications [19]. Transaction history of a bank customer, product order history of a company, performance of the stock market [20] and biological DNA data [21] are all sequence data, where sequence data mining techniques are applied.

2.1.8 Soft computing

Soft computing is a collection of techniques in artificial intelligence, which can be used to handle uncertainty, imprecision and partial truth. The guiding principle is to provide the computation method that leads to an approximate solution at low cost, thereby speeding up the process. Fuzzy sets, which constitute the oldest component of soft computing, are suitable for handling issues related to understandability of patterns, incomplete/noisy data and can provide approximate solutions faster [22].

3 Related work

Patil et al. [14] proposed a method for species identification based on approximate pattern matching. The novelty in the work was feature extraction technique for genome data. The existing *n*-gram based methods extract frequencies of 4^n features from genome data. In [14] authors extracted all candidate/subsequences that satisfy: length grater or equal to given minimum length, given number of mismatches and support grater or equal to user threshold. These frequent subsequences are used as features to construct a binary table where the presence or absence of an attribute/feature in a sequence is represented by 1 or 0 respectively. Classification of genome sequences has been done using data mining techniques namely, naive Bayes, support vector machine and *k*-nearest neighbor. Based on experimentation, data mining techniques with approximate patterns showed better results. In this work very short sequences were analyzed and all the frequent subsequences represented in binary form are used as features for classifier.

In [16], a set of 16 kinds of dinucleotide compositions was used to analyze the protein-encoding nucleotide sequences in nine complete genomes: *Escherichia coli*, *Haemophilus influenzae*, *Helicobacter pylori*, *Mycoplasma genitalium*, *Mycoplasma pneumoniae*, *Synechocystis* sp., *Methanococcus jannaschii*, *Archaeoglobus fulgidus*, and *Saccharomyces cerevisiae*. The dinucleotide composition was significantly different between the organisms. The distribution of genes from an organism was clustered around its center in the dinucleotide composition space. The genes from closely related organisms such as Gram-negative bacteria, mycoplasma species and eukaryotes

showed some overlap in the space. The genes from nine complete genomes together with those from humans were discriminated into respective clusters with 80 % accuracy using the dinucleotide composition alone.

Classification of organisms into 2 classes—Bacteria and Archea, based on their di-nucleotide frequencies in DNA sequence using naive Bayesian approach was discussed by [23]. The methodology is based on scanning all genomes for the occurrences of all possible overlapping motifs with a length of n nucleotides. Then a genomic sequence is chosen at random from anywhere inside a genome. From this genomic sequence, all overlapping motifs are extracted. The naive Bayesian classifier uses the extracted motifs to predict their most probable genomic origin by comparing the frequencies of the extracted motifs with the motif frequencies of the different genomes. The average accuracy obtained was 85 %.

Protein classification into domains of life was attempted and the test protein was predicted to be of bacterial or eukaryotic origin with 85 % accuracy using a Markov model for compositional bias analysis [24].

Norashikin et al. [25], used 2-g encoding, which is the frequency of occurrence of two consecutive amino acids from a protein sequence to investigate the effect of the spread factor value towards cluster separation in the growing self-organizing map GSOM. They used simple k-means algorithm as a method to identify clusters in the GSOM. By using Davies–Bouldin index, clusters formed by different values of spread factor were obtained and the resulting clusters of protein sequences were analyzed.

Andrija et al. [4] have addressed the problem of automated classification of isolates, i.e., the problem of determining the family of genomes to which a given genome belongs and also the problem of automated unsupervised hierarchical clustering of isolates, according only to their statistical substring properties. For both of these problems, they presented novel algorithms based on nucleotide n-grams, with no required preprocessing steps such as sequence alignment.

In [26] an approach for genome data clustering based on approximate matching is proposed. The proposed work includes finding total number of approximate matches to a query with specified fault tolerance from the genome data sequences. The number of matches is used as a feature value for clustering. Clustering has been done using soft computing technique namely, Fuzzy C-means (FCM), Possibilistic C-means (PCM) and results are compared to hard clustering technique, i.e., K-means. Experimental results are reported for 100 randomly selected samples (size 10,000 bp) from two different complete genome data sets namely, Yeast, *E. coli* and *Drosophila*, Mouse. It is also verified that proposed method outperforms existing n-gram frequency based method for both the data sets. Overall performance comparison of different clustering techniques shows that FCM has performed better compared to K-means and PCM for both the data sets. PCM is also a good clustering technique to genome data clustering except the undesirable effect of coincident centroids formed at lower tolerances.

Narasimhan et al. [17] designed a scheme for automatic identification of a species from its genome sequence. A set of 64 3-tuple keywords was first generated using the 4 types of bases A (for Adenine), T (for Thymine), C (for Cytosine) and G (for Guanine). These keywords were searched on randomly sampled genome sequence of a given length (10,000 elements) and frequency count for each of the $4^3 = 64$ keywords was determined to obtain a DNA-descriptor. The process was repeated for N such sampled genome sequences and then Principal Component Analysis was employed on the frequency counts for N sampled instances to obtain a unique feature descriptor which identifies the species from its genome sequence. It was shown that the feature descriptors were effective representatives of the structural signature of the species. No quantitative measures of accuracy are seen reported.

Narasimhan and coworkers [18] also proposed an alternative approach to automatic classification and identification of species using self-organizing feature map. The computational map was trained by using the DNA-descriptors (frequency count for each of the $4^3 = 64$ keywords from N number of sampled genome sequences of given length 10,000) of different species as the training inputs. The maps for different dimensions were constructed and analyzed for optimum performance. The scheme presented a novel method for identifying a species from its genome sequence with the help of a two dimensional map of neuron clusters, where each cluster represents a particular species. The map has been shown to provide an easier technique for recognition and classification of a species based on its genomic data. But once again no quantitative measures of accuracy are seen reported.

Wei et al. [27] attempted classification of several DNA sequences of bacteria using the artificial neural network model. The "dinucleotides compositions" method was used to characterize the DNA sequences which transform every DNA sequence to a 16-dimension vector. A back-propagation artificial neural network was developed and trained using "leave-one-out" method. Results showed that the accuracy of classification was 84.3 %, which proved that the model was satisfactory in summary. However, the author stated that the applicability of the characterization strategy needs to be improved to reflect the features of the DNA sequences.

All the work mentioned above take DNA sequence for classification. The possibility of species classification using CGR images of DNA sequences, by using different

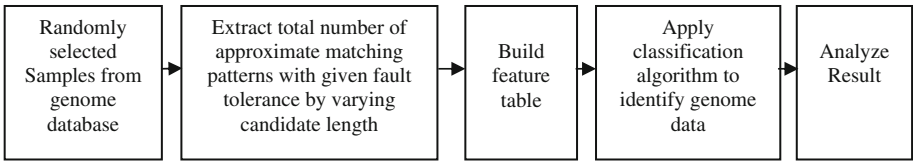

Fig. 1 Proposed approach for genome data classification based on approximate matching

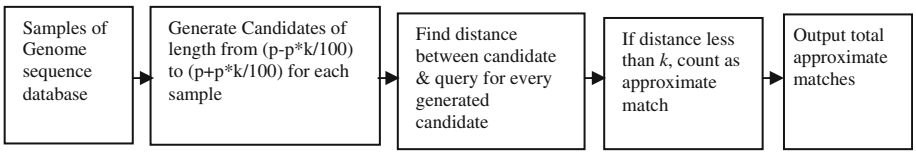

Fig. 2 Extraction of total number of approximate matches from genome samples

distance metrics [28] and by using neural networks [29] has been investigated. In both these works, only species identification is done taking a few different species. A detailed classification problem, addressing 6 categories in the taxonomical hierarchy of eukaryotic organisms, using a combination of FCGR and naive Bayesian approach is also attempted [30]. The average classification accuracy is reported as 85.63 %.

Thus, the drawbacks of existing approaches for genome identification are as follows:

1. Existing methods extract dinucleotide composition or trinucleotide composition from the sequences using exact pattern matching. In most real world biological problems, exact matching may not give desired results because the sequences are similar. Thus searching in sequence repositories often requires going beyond exact matching [11].
2. In the *n*-gram frequency based classification, the number of features (4^n for DNA sequence and 20^n for protein sequence) for classifiers, increases with increase in *n* and sometimes the classifier performance gets hampered by a lot of redundant features in the dataset [31]. Further, algorithms with *n*-grams are computationally too expensive [4].
3. Sequence alignment algorithms and techniques for estimating homologies and mismatches among DNA sequences that are used for comparing sequences of relatively small sizes, are not applicable to sequences with sizes varying between a few thousand base pairs to a few hundred thousand base pairs. Even for comparison of small sequences, the standard alignment and matching algorithms are known to be time consuming. There is a need for procedures that may be somewhat approximate in nature, yet useful in producing quick and significant results. To fill this gap [17, 18] has clustered complete genome data of Yeast and *E. coli*, by sampling it into size of 10,000 elements and choosing *N* such random samples. But no quantitative measures of accuracy are reported.

In the present work, we propose a novel approach for genome identification based on approximate pattern matching. Given a database of randomly selected samples of genomic sequences, our proposed work includes extraction of total number of fuzzy matching patterns with given fault tolerance. These total number of fuzzy matches are used as features for classifier. Classification of genome sequences has been done by data mining techniques and the number of subsequences that are used as features for classifier depends on the tolerance allowed. Experimental results are reported for randomly selected samples from complete genome data of Yeast and *E. coli*. The proposed approach is compared with *n*-gram sequence encoding method in binary form which resulted in highest accuracy of 53 %. But the proposed approach based on approximate matching resulted in highest accuracy of 99 % with 70 % tolerance and sampling size of 10,000 bp.

4 Methods

Our proposed approach used for species identification from genome data is as shown in Fig. 1. First, we sample the complete genome data of species into 2,000–10,000 bp size and then we choose randomly 200 such samples. In the next step we extract approximate matches for a given query with specified fault tolerance by varying candidate length so as to allow both positive and negative tolerance from the given sequences. These total approximate matches are used as features for classification algorithms and then the results are analyzed. Figure 2 shows extraction of total number of approximate matching patterns from a genome sequence database with query pattern of length *p* and percentage of fault tolerance allowed *k*.

In the next section, two algorithms namely, algorithm for finding approximate occurrences based on Levenshtein distance and algorithm for finding total number of approximate matches by varying candidate length so as to allow both positive and negative tolerance from the genome data sequences are explained. Finally, we explain classification techniques used in our method.

4.1 Algorithm for finding approximate occurrences

Given a pattern P, a text string T ($|P| = m$ and $|T| = n$) and fuzziness factor k, the task is to find all positions j in T such that there exists a suffix of $T[1\ldots,j]$ that has edit distance of less than k with P. We first define:

$$D(i,j) = \min_{1 \le i \le j} \text{edit distance between } T[1\ldots,j] \text{ and } P[1\ldots,i].$$

$D(i, j)$ will hold the Levenshtein distance between the first i characters of P and the first j characters of T for all i, j.

We then use the following recursion to compute the table $D(ij)$ for all i and j:

$$D(i,0) = i$$
$$D(0,j) = j$$
$$D(i,j) = \min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1) + \left(\text{if } P_i = T_j, \quad \text{then } 0 \text{ else } 1\right) \end{cases} \quad (1)$$

The positions of interest are those j's for which $D(m,j) < k$.

4.2 Algorithm for finding total number of approximate matches for a specified fault tolerance from sequence database

Given complete genome data of length l, sample size s. Sample the complete genome data into N number of sequences of size s. Now, given a sequence database with N number of sequences, query pattern of length p and percentage of fault tolerance allowed k, we want to find, for every sequence in the database, the total number of candidate patterns that will approximately match the query pattern by varying candidate length, so as to allow both positive and negative tolerance (at most k). The algorithm is given below.

1 **for** all sequences in database **do** step 2.
  2 **for** all candidate length from ($p - p*k/100$) to ($p + p*k/100$) **do** steps 3–8
    3 Generate candidates of length in step 2.
    4 Count = 0.
    5 Find the Levenshtein distance between candidate pattern and query pattern.
    6 If distance $\le k$, then increment count.
    7 Repeat steps 3 and 4 for all generated candidates.
    8 Output count as total number of approximate matches.
  End **for**
End **for**

In step 2, we vary candidate length so as to allow both positive and negative tolerance, i.e., if query pattern length is p and percentage of tolerance allowed is k, then candidate length is varied from ($p - p*k/100$) to ($p + p*k/100$). For example, if query length is 10 and tolerance allowed is 50 % then candidate length is varied from (10 − 50/100) to (10 + 50/100), i.e., from 5 to 15.

In the next step, candidates of length as in step 2 are generated. A candidate is a subsequence extracted from a larger sequence. For a given sequence, the set of the candidates with length n [in our approach n is from ($p - p*k/100$) to ($p + p*k/100$)] can be generated by sliding a window of n characters on the whole sequence. This movement is carried out character by character. With each movement a subsequence of n characters is extracted.

Once candidates are generated, we match each candidate to the query by using Levenshtein distance. If the distance is less than or equal to the specified tolerance, then the candidate pattern is counted as a approximate match. This process is repeated for the entire database. We select a query pattern that occurs frequently in the given database of sequences.

Once all the approximate matching patterns are extracted, we build a feature table where each row corresponds to a sequence and each column is a subsequence/candidate of length from ($p - p*k/100$) to ($p + p*k/100$). Therefore, the number of columns/features depends on the fault tolerance allowed. The value in each column for a particular sequence corresponds to the number of approximate matches. This feature table is called a learning context. It represents the result of the preprocessing step and the new sequence encoding format. In the mining step, clustering algorithms are applied to the learning context to identify genome data into different groups.

Example 1 Consider the sequence Seq1: AGCTTGCAAT

Let the query be AGCG of length 4 and tolerance allowed is 50 %. Therefore, candidate length varies from 2 to 6. To encode the given sequences, we first generate candidates and then find the Levenshtein distance between each candidate and query as shown in Table 3. Finally the number of approximate matches within the given tolerance is counted.

Approximate matches within the given tolerance are marked as bold in Table 3. Total approximate matches for candidates generated with length 2 (L2) for the given query and for the given tolerance are 3. Similarly total approximate matches for candidates with L3, L4, L5 and L6 are 4, 2, 1 and 1, respectively. Table 4 shows the sequence encoding/feature table in which a row indicates sequence and every column value for the sequence indicates total approximate matches.

Table 3 Candidates generated and their respective distance with query AGCG for the example

Candidates generated with L 2	D	Candidates generated with L 3	D	Candidates generated with L 4	D	Candidates generated with L 5	D	Candidates generated with L 6	D
AG	**2**	AGC	**1**	AGCT	**1**	AGCTT	**2**	AGCTTG	**2**
GC	**2**	GCT	**2**	GCTT	3	GCTTG	3	GCTTGC	4
CT	3	CTT	4	CTTG	3	CTTGC	4	CTTGCA	4
TT	4	TTG	3	TTGC	3	TTGCA	3	TTGCAA	4
TG	3	TGC	**2**	TGCA	**2**	TGCAA	3	TGCAAT	4
GC	**2**	GCA	**2**	GCAA	3	GCAAT	4		
CA	3	CAA	4	CAAT	4				
AA	3	AAT	3						
AT	3								

L length, *D* distance

Table 4 Sequence encoding for the given example

Sequence number	Total fuzzy matches with L 2	Total fuzzy matches with L 3	Total fuzzy matches with L 4	Total fuzzy matches with L 5	Total fuzzy matches with L 6
Seq 1	3	4	2	1	1

Table 5 Confusion matrix

	Predicted class	
	Yes (*E. coli*)	No (Yeast)
Actual class		
Yes (*E. coli*)	True positive	False negative
No (Yeast)	False positive	True negative

4.3 Algorithms used for classification

We have used four classifiers namely back propagation (BP), naive Bayes (NB), support vector machine (SVM) and K-nearest neighbor (KNN). These classifiers are briefly explained below (Table 5).

4.3.1 Back propagation

Figure 3 shows the structure of back-propagation neural network model. The artificial neural network (ANN) model we designed includes three layers- one input layer, one hidden layer and one output layer. The input layer includes nodes equal to the number of candidates/subsequences for a specified tolerance; the number of nodes in the hidden layer needs to be determined in the training process; the output layer includes two nodes which represents the kind of genome data, i.e., *E. coli* and Yeast. The back-propagation algorithm with a momentum term was used in training the ANN model [32]. During training, the predicted output is compared with the desired output, and error is calculated. If the error is more than a prescribed

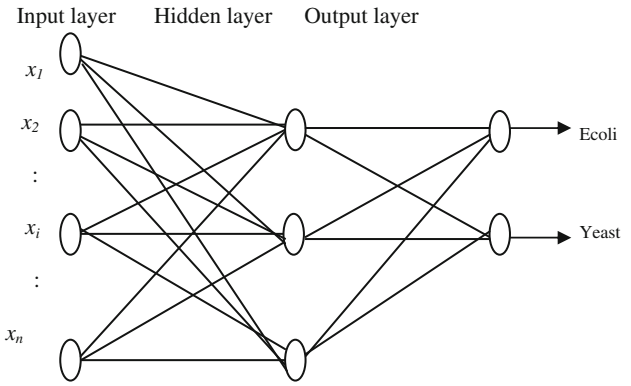

Fig. 3 A multilayer feed-forward neural network

limiting value, it is back propagated from output to input, and weights are further modified till the error or number of iterations is within a prescribed limit.

The general rule for updating weights is:

$$\Delta w_{ji} = \eta \delta_j o_i \tag{2}$$

η is a positive number (called learning rate), which determines the step size in the gradient descent search. A large value enables back propagation to move faster to the target weight configuration but it also increases the chance of its never reaching this target. o_i is the output computed by neuron i. $\delta_j = o_j(1 - o_j)(T_j - o_j)$ for the output neurons where T_j wanted output for the neuron j and $\delta_j = o_j(1 - o_j)\sum_k \delta_k w_{kj}$ for the internal neurons. In our experiment, learning rate of the model is 0.3, the coefficient of the momentum term is 0.2, and the number of iterations is 500.

4.3.2 Naive Bayes

The simple naive Bayes (NB) algorithm [33] is used in this study. The main advantage of Bayesian classifier is that they are probabilistic models, robust to deal with the real data noise and missing values [34]. In addition, it also has advantages in terms of simplicity, learning speed, classification speed and storage space [35]. Naïve Bayes is simplified version of Bayes theorem that is used to classify the unknown instances into relevant class. Posterior probability of each class is calculated based on given attribute value associated with each tuple.

$$p(C_i/v_1, v_2 \ldots, v_n) = \frac{p(C_i)\prod_{j-1}^{n} p(v_j/C_i)}{p(v_1, v_2 \ldots, v_n)} \quad (3)$$

The posterior probability of class C_i given the attribute $\langle v_1, v_2 \ldots, v_n \rangle$. Learning with the Naive Bayes classifier involves estimating the probabilities in the RHS Eq. 3 from the training tuples

4.3.3 Support vector machine (SVM)

The SVM is a supervised classification algorithm that learns by example to discriminate among two or more given classes of data. Given a training set in a vector space, SVMs find the best decision hyper plane that separates two classes. The quality of a decision hyper plane is determined by the distance between two hyper planes defined by support vectors. The best decision hyper plane is the one that maximizes this margin. SVM extends its applicability on the linearly non-separable data sets by either using soft margin hyper planes or by mapping the original data vectors into a higher dimensional space in which the data points are linearly separable [36]. The mapping to higher dimensional spaces is done using appropriate kernel functions.

For a binary classification problem, assume that we have a series of feature vectors x_i and class labels y_i ($i = 1, 2 \ldots, N$, where N is the number of samples), where $x_i \in R^n$ and $y_i \in \{-1, +1\}$ The SVM requires the solution of the following optimization problem [37]:

$$\text{Min}\frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i \quad (4)$$

subject to $\gamma_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0$.

Here, feature vectors x_i are mapped into a higher dimensional space by the function $\phi(x)$. Then SVM constructs an optimal separating hyper plane (OSH), which maximizes the margin in the higher dimensional space. $C > 0$ is the penalty factor of the error term. Furthermore, $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is called the kernel function.

There are several typical kernel functions. In this work, we have adopted SVM with polynomial kernel function. The polynomial kernel has strong generalization ability [38].

Polynomial kernel function: $K(x, y) = (x \cdot y + 1)^p$.

4.3.4 K-nearest neighbor (KNN)

KNN classification algorithm assumes that all instances correspond to points in an n-dimensional space. Nearest neighbors of an instance are described by a distance/similarity measure. When a new sample comes, a KNN classifier searches the training dataset for the k closest sample to the new sample using distance/similarity measure for determining the nature of new sample. These k samples are known as the k nearest neighbors of the new sample. The new sample is assigned the most common class of its k nearest neighbors. KNN is the best choice for making a good classifier, when simplicity and accuracy is important issues [39, 40]. In the nearest neighbor model, choice of a suitable distance function and the value of the members of nearest neighbors (k) are very crucial. The k represents the complexity of nearest neighbor model. The model is less adaptive with higher k values. We have used Euclidean distance measure with $k = 1$ in our experiment which are default parameters in WEKA and these parameters resulted in highest accuracy for the experimental data used.

5 Results and discussion

In our experiment, we used complete genome data of two different species namely the bacterium *Escherichia coli* (*E. coli*) [41], *Saccharomyces cerevisiae* (Yeast) [42, 43]. *E. coli* data is downloaded from NCBI [44] and the total length is 4639675 bp. Total length of complete genome data of Yeast is 12,136, and 020 bp (with mitochondrial genome). Since complete genome data of species is very huge, data is sampled into different sizes. We classified sequences of five different lengths: 2000, 4000, 6000, 8000, and 10,000 bp and monitored the classification accuracy. In each case, proposed model is tested with total 200 samples out of which 100 samples are from *E. coli* and 100 samples are from Yeast.

The experiments were done on an Intel pentium-4 processor-based machine having a clock frequency of 2.66 GHz and 1 GB RAM. In the classification process we use the k-fold cross-validation in which, the data was randomly partitioned into k subset or k-fold each having approximately equal size. Training and testing is performed k times and each time one of the subset is held out in turn. The classifier is trained on the remaining $k - 1$ subsets to

Table 6 Accuracy (in %) of different classifiers with 10 % tolerance

Size of sample	NB	BP	SVM	KNN
2,000	47	**49**	46.5	44
4,000	51.5	**52**	50.5	**52**
6,000	**54**	52.5	50.5	53.5
8,000	54	**55.5**	52	55
10,000	57.5	**60**	54	57

Table 7 Accuracy (in %) of different classifiers with 20 % tolerance

Size of Sample	NB	BP	SVM	KNN
2,000	51.5	47	**52.5**	**52.5**
4,000	**60.5**	57.5	57	55.5
6,000	62	61	**63**	56
8,000	**65**	62	64	59
10,000	66	63.5	**66.5**	60.5

Table 8 Accuracy (in %) of different classifiers with 30 % tolerance

Size of sample	NB	BP	SVM	KNN
2,000	69.5	**70**	67	64
4,000	74	71	**75.5**	64.5
6,000	80.5	77.5	**82.5**	71.5
8,000	**83**	78	**83**	76
10,000	84.5	81	**86.5**	77

Table 9 Accuracy (in %) of different classifiers with 40 % tolerance

Size of Sample	NB	BP	SVM	KNN
2,000	77.5	75.5	**79**	72
4,000	**85.5**	84	**85.5**	76
6,000	89.5	**92**	90.5	85
8,000	92	**93**	**93**	86.5
10,000	95	94.5	**95.5**	93

build classification model and classification error of the iteration is calculated by testing the classification model on the holdout set. Finally, the k numbers of errors are summed up to yield an overall error estimate. Obviously, at the end of cross-validation, every sample has been used exactly once for testing.

We used the following classifiers: SVM, NB and KNN and multilayer neural network with BP of the workbench WEKA [45]. We generated and tested the classification models; then the classification accuracies (rate of correctly classified sequences) are reported.

To extract approximate matching patterns from genome sequences, we used query of length 10 and allowed tolerance of 10–70 %. The proposed model is tested separately

Table 10 Accuracy (in %) of different classifiers with 50 % tolerance

Size of sample	NB	BP	SVM	KNN
2,000	86	**88.5**	84	80.5
4,000	88.5	90	**91**	85
6,000	93	**93.5**	**93.5**	92
8,000	94.5	**96**	94.5	94.5
10,000	96.5	**98.5**	96.5	96.5

for fuzzy matching patterns extracted with each of the fault tolerance and the classification accuracies are monitored. The experimental results vary according to the tolerance allowed as well as according to sampling/sequence size.

Tables 6, 7, 8, 9 and 10 shows performance of different classifiers for different sample size (sequence length) with specified fault tolerance. Same query of fixed length 10 is used in all the experiment. It can be observed from the obtained results that, classification accuracy increases with increase in fault tolerance as well as increase in sampling size of the sequences. Highest accuracy obtained at each sample size is marked as bold in Tables 6, 7, 8, 9 and 10. Our results show that, the classification accuracy achieved is 98.5 % by BP, 96.5 % by other classifiers, i.e., by NB, SVM and KNN with sampling/sequence size of 10,000 bp and with allowed tolerance of 50 %.

Tables 11, 12, 13, 14 and 15 shows detailed performance comparison. For every classification technique shown in Tables 11, 12, 13, 14 and 15 confusion matrix column makes use of four values. Left upper side indicates true positive and right upper side indicates false negative. Similarly, lower left side indicates false positive and lower right side indicates true negative. The result of confusion matrix is used to calculate the accuracy, sensitivity and specificity of a classifier. Kappa value of the BP is 0.97 and 0.93 by other classifiers, i.e., by NB, SVM and KNN with sampling/sequence size of 10,000 bp and with allowed tolerance of 50 %. The area under the curve (AUC) for BP is 0.999 which is the largest as compared to the other three classifiers viz. NB, SVM and KNN with an area of 0.996, 0.965, and 0.965, respectively, with sampling/sequence size of 10,000 bp and with allowed tolerance of 50 %.

Tables 16 and 17 shows the performance comparison of different classification methods at 60 and 70 % tolerance, respectively, with sampling size of 10,000 bp. Our results show that, the classification accuracy achieved is 98.5 % by BP and NB, 98 % by SVM and 97 % by KNN with sampling/sequence size of 10,000 bp and with tolerance of 60 %. The highest classification accuracy achieved is 99 % by all the classification methods used in the experiment at 70 % tolerance and at sampling size of 10,000 bp. The AUC for NB and BP is 1 which indicates a model with perfect accuracy at 70 % tolerance.

Table 11 Confusion matrix, accuracy, sensitivity, specificity, AUC and Kappa of four classification methods at 10 % tolerance

Classifiers	Sample size	Confusion matrix		Accuracy (%)	Sensitivity (%)	Specificity (%)	AUC	Kappa value
NB	2,000	18	82	47	18	76	0.445	−0.06
		24	76					
	4,000	83	17	51.5	83	20	0.55	0.03
		80	20					
	6,000	89	11	54	89	19	0.506	0.08
		81	19					
	8,000	77	23	54	77	31	0.493	0.08
		69	31					
	10,000	78	22	57.5	78	37	0.53	0.15
		63	37					
SVM	2,000	37	63	46.5	37	56	0.465	−0.07
		44	56					
	4,000	88	12	50.5	88	13	0.505	0.01
		87	13					
	6,000	88	12	50.5	88	13	0.505	0.01
		87	13					
	8,000	87	13	52	87	17	0.52	0.04
		83	17					
	10,000	82	18	54	82	26	0.54	0.08
		74	26					
BP	2,000	59	41	49	59	39	0.475	−0.02
		61	39					
	4,000	77	23	52	77	27	0.537	0.04
		73	27					
	6,000	80	20	52.5	80	25	0.508	0.05
		75	25					
	8,000	75	25	55.5	75	36	0.564	0.11
		64	36					
	10,000	82	18	60	82	38	0.574	0.2
		62	38					
KNN	2,000	62	38	44	62	26	0.4	−0.12
		74	26					
	4,000	77	23	52	77	27	0.444	0.04
		73	27					
	6,000	85	15	53.5	85	22	0.478	0.07
		78	22					
	8,000	76	24	55	76	34	0.51	0.1
		66	34					
	10,000	77	23	57	77	37	0.536	0.14
		63	37					

5.1 Effect of tolerance on classification accuracy

The proposed model has been tested separately by varying tolerance from 10 to 70 %. When the allowed tolerance is only 10 %, since our query pattern is of length 10, we varied the candidate length (subsequences that are used as features for classifier) from 9 to 11. It indicates that, when candidates generated are of length 9, all 9 characters must match to query pattern. Similarly, for the candidates with length 10, 11 any 9 characters, any 10 characters, respectively, in the candidate must match to query pattern for the given tolerance of 10 %. Hence in this case, for a given sequence, the percentage of matching of candidate to query is very less, i.e., total fuzzy matches will be less. Number of subsequences that are used as features for classifier are only 3 (candidates of length from 9 to 11). Hence all the

Table 12 Confusion matrix, accuracy, sensitivity, specificity, AUC and Kappa of four classification methods at 20 % tolerance

Classifiers	Sample size	Confusion matrix		Accuracy (%)	Sensitivity (%)	Specificity (%)	AUC	Kappa value
NB	2,000	69	31	51.5	69	34	0.527	0.03
		66	34					
	4,000	74	26	60.5	74	47	0.659	0.21
		53	47					
	6,000	70	30	62	70	54	0.703	0.24
		46	54					
	8,000	69	31	65	69	61	0.724	0.3
		39	61					
	10,000	74	26	66	74	58	0.734	0.32
		42	58					
SVM	2,000	77	23	52.5	77	28	0.525	0.05
		72	28					
	4,000	73	27	57	73	41	0.57	0.14
		59	41					
	6,000	70	30	63	70	56	0.63	0.26
		44	56					
	8,000	65	35	64	65	63	0.64	0.28
		37	63					
	10,000	72	28	66.5	72	61	0.665	0.33
		39	61					
BP	2,000	64	36	47	64	30	0.491	-0.06
		70	30					
	4,000	73	27	57.5	73	42	0.583	0.15
		58	42					
	6,000	64	36	61	64	58	0.674	0.22
		42	58					
	8,000	70	30	62	70	54	0.657	0.24
		46	54					
	10,000	60	40	63.5	60	67	0.674	0.27
		33	67					
KNN	2,000	62	38	52.5	62	43	0.551	0.05
		57	43					
	4,000	58	42	55.5	58	53	0.561	0.11
		47	53					
	6,000	60	40	56	60	52	0.545	0.12
		48	52					
	8,000	74	26	59	74	44	0.537	0.18
		56	44					
	10,000	62	38	60.5	62	59	0.619	0.21
		41	59					

classifiers that are used in our proposed work resulted in less accuracy.

When the allowed tolerance is 20 %, since our query is of length 10, we varied the candidate length (subsequences that are used as features for classification) from 8 to 12. Hence the subsequences that are used as features for classification are 5 (candidates of from length 8 to 12). A tolerance of 20 % indicates that the maximum allowed distance in matching a candidate to a query is 2. It indicates that when candidates generated are of length 8, all 8 characters must match the query pattern. Similarly, for the candidates of length 9, 10, 11 and 12, any 8 characters, any 8 characters, any 9 characters and any 10 characters, respectively, in the candidate must match the query pattern.

Table 13 Confusion matrix, accuracy, sensitivity, specificity, AUC and Kappa of four classification methods at 30 % tolerance

Classifiers	Sample size	Confusion matrix		Accuracy (%)	Sensitivity (%)	Specificity (%)	AUC	Kappa value
NB	2,000	75	25	69.5	75	64	0.731	0.39
		36	64					
	4,000	77	23	74	77	71	0.801	0.48
		29	71					
	6,000	83	17	80.5	83	78	0.893	0.61
		22	78					
	8,000	83	17	83	83	83	0.881	0.66
		17	83					
	10,000	84	16	84.5	84	85	0.922	0.69
		15	85					
SVM	2,000	67	33	67	67	67	0.67	0.34
		33	67					
	4,000	79	21	75.5	79	72	0.755	0.51
		28	72					
	6,000	84	16	82.5	84	81	0.825	0.65
		19	81					
	8,000	80	20	83	80	86	0.83	0.66
		14	86					
	10,000	86	14	86.5	86	87	0.865	0.73
		13	87					
BP	2,000	74	26	70	74	66	0.769	0.4
		34	66					
	4,000	78	22	71	78	64	0.745	0.42
		36	64					
	6,000	78	22	77.5	78	77	0.826	0.55
		23	77					
	8,000	81	19	78	81	75	0.82	0.56
		25	75					
	10,000	84	16	81	84	78	0.885	0.62
		22	78					
KNN	2,000	63	37	64	63	65	0.64	0.28
		35	65					
	4,000	69	31	64.5	69	60	0.645	0.29
		40	60					
	6,000	73	27	71.5	73	70	0.715	0.43
		30	70					
	8,000	78	22	76	78	74	0.76	0.52
		26	74					
	10,000	77	23	77	77	77	0.77	0.54
		23	77					

Hence, for this case, fuzzy matches between the candidates generated and query will increase as well as number of subsequence/features for classifier also increases. Therefore, our results show that there is increase in classification accuracy compared to 10 % tolerance. Maximum classification accuracy achieved for 20 % tolerance and with sample/sequence size of 10,000 bp is 66.5 % by SVM. Other classifiers NB, BP and KNN resulted with 66, 63.5 and 60.5 % accuracy, respectively, for sampling size of 10,000 bp.

When the tolerance allowed is 30 and 40 %, the number of features are 7 (from length 7 to 13), 9 (from length 6 to 12), respectively, and maximum allowed distance in matching is 3 and 4, respectively. Hence in these two cases,

Table 14 Confusion matrix, accuracy, sensitivity, specificity, AUC and Kappa of four classification methods at 40 % tolerance

Classifiers	Sample size	Confusion matrix		Accuracy (%)	Sensitivity (%)	Specificity (%)	AUC	Kappa value
NB	2,000	80	20	77.5	80	75	0.849	0.55
		25	75					
	4,000	88	12	85.5	88	83	0.895	0.71
		17	83					
	6,000	92	8	89.5	92	87	0.952	0.79
		13	87					
	8,000	95	5	92	95	89	0.963	0.84
		11	89					
	10,000	95	5	95	95	95	0.985	0.9
		5	95					
SVM	2,000	81	19	79	81	77	0.79	0.58
		23	77					
	4,000	88	12	85.5	88	83	0.855	0.72
		17	83					
	6,000	93	7	90.5	93	88	0.905	0.81
		12	88					
	8,000	97	3	93	97	89	0.93	0.86
		11	89					
	10,000	97	3	95.5	97	94	0.955	0.91
		6	94					
BP	2,000	79	21	75.5	79	72	0.782	0.51
		28	72					
	4,000	84	16	84	84	84	0.868	0.68
		16	84					
	6,000	94	6	92	94	90	0.925	0.84
		10	90					
	8,000	93	7	93	93	93	0.954	0.86
		7	93					
	10,000	96	4	94.5	96	93	0.978	0.89
		7	93					
KNN	2,000	75	25	72	75	69	0.72	0.44
		31	69					
	4,000	80	20	76	80	72	0.76	0.52
		28	72					
	6,000	86	14	85	86	84	0.85	0.7
		16	84					
	8,000	86	14	86.5	86	87	0.865	0.73
		13	87					
	10,000	93	7	93	93	93	0.93	0.86
		7	93					

for a given sequence, the percentage of matching of candidate to query is slightly increased compared to 20 % tolerance, i.e., total approximate matches will be more and feature values start being distinguishable for Yeast, *E. coli* genome data. Therefore, increase in classification accuracy can be observed from 30 % tolerance compared to 20 % tolerance. At 40 % tolerance, SVM resulted in maximum accuracy of 95.5 % for sample size of 10,000 bp compared to other classifiers.

When the allowed tolerance is 50 %, number of subsequences that are used as features for classifier are 11 and are of length from 5 to 15. In this case, maximum distance allowed in matching candidate to query is 5. Since, 50 % of the mismatch is allowed in matching a candidate to query,

Table 15 Confusion matrix, accuracy, sensitivity, specificity, AUC and Kappa of four classification methods at 50 % tolerance

Classifiers	Sample size	Confusion matrix		Accuracy (%)	Sensitivity (%)	Specificity (%)	AUC	Kappa value
NB	2,000	88	12	86	88	84	0.914	0.72
		16	84					
	4,000	89	11	88.5	89	88	0.942	0.77
		12	88					
	6,000	96	4	93	96	90	0.975	0.86
		10	90					
	8,000	97	3	94.5	97	92	0.983	0.89
		8	92					
	10,000	96	4	96.5	96	97	0.996	0.93
		3	97					
SVM	2,000	88	12	84	88	80	0.84	0.68
		20	80					
	4,000	94	6	91	94	88	0.91	0.82
		12	88					
	6,000	96	4	93.5	96	91	0.935	0.87
		9	91					
	8,000	94	6	94.5	94	95	0.945	0.89
		5	95					
	10,000	96	4	96.5	96	97	0.965	0.93
		3	97					
BP	2,000	91	9	88.5	91	86	0.89	0.77
		14	86					
	4,000	91	9	90	91	89	0.955	0.8
		11	89					
	6,000	95	5	93.5	95	92	0.974	0.87
		8	92					
	8,000	96	4	96	96	96	0.978	0.92
		4	96					
	10,000	99	1	98.5	99	98	0.999	0.97
		2	98					
KNN	2,000	82	18	80.5	82	79	0.805	0.61
		21	79					
	4,000	84	16	85	84	86	0.85	0.7
		14	86					
	6,000	90	10	92	90	94	0.92	0.84
		6	94					
	8,000	93	7	94.5	93	96	0.945	0.89
		4	96					
	10,000	97	3	96.5	97	96	0.965	0.93
		4	96					

number of fuzzy matches will increase. In this case, our experimental results show that the classification accuracy of 98.5 % by BP and 96.5 % by other classifiers, i.e., by NB, SVM and KNN with sampling/sequence size of 10,000 bp.

Similarly when the tolerance allowed is 60 and 70 %, number of subsequences that are used as features for classifier are 13 and 15 which are of length from 4 to 16 and 3 to 17, respectively. In this case, our experimental results show that the classification accuracy achieved is 98.5 % by BP and NB, 98 % by SVM and 97 % by KNN with sampling/sequence size of 10,000 bp and with tolerance of 60 %. The highest classification accuracy achieved is 99 % by all the classification methods used in the

Table 16 Confusion matrix, accuracy, sensitivity, specificity, AUC and Kappa of four classification methods at 60 % tolerance and sample size 10,000 bp

Classifier	Confusion matrix		Accuracy (%)	Sensitivity (%)	Specificity (%)	AUC	Kappa value
NB	98	2	98.5	98	99	0.999	0.97
	1	99					
SVM	97	3	98	97	99	0.98	0.96
	1	99					
BP	98	2	98.5	98	99	0.994	0.97
	1	99					
KNN	97	3	97	97	97	0.97	0.94
	3	97					

Table 17 Confusion matrix, accuracy, sensitivity, specificity, AUC and Kappa of four classification methods at 70 % tolerance and sample size 10,000 bp

Classifier	Confusion matrix		Accuracy (%)	Sensitivity (%)	Specificity (%)	AUC	Kappa value
NB	99	1	99	99	99	1	0.98
	1	99					
SVM	99	1	99	99	99	0.99	0.98
	1	99					
BP	99	1	99	99	99	1	0.98
	1	99					
KNN	99	1	99	99	99	0.99	0.98
	1	99					

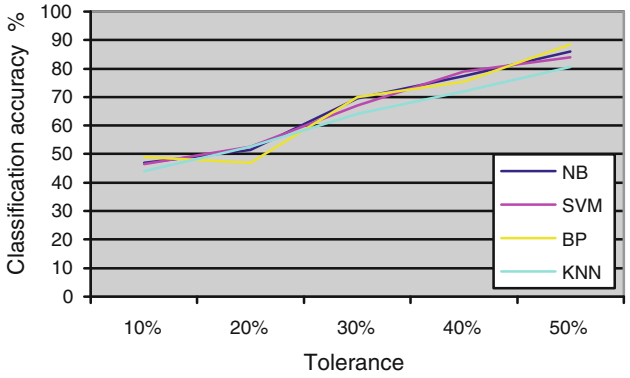

Fig. 4 Classification accuracy for sample size 2,000 at different tolerance

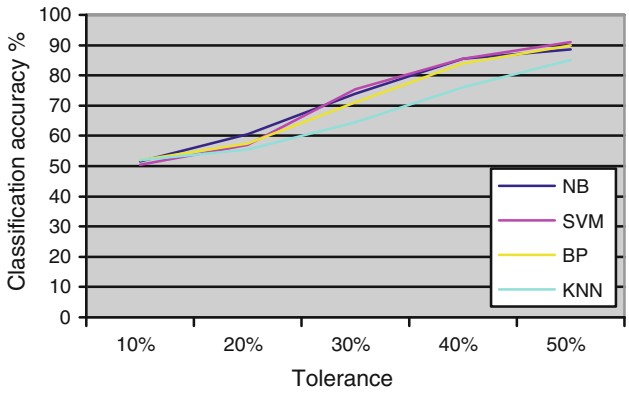

Fig. 5 Classification accuracy for sample size 4,000 at different tolerance

experiment at 70 % tolerance and at sampling size of 10,000 bp.

Figures 4, 5, 6, 7 and 8 shows change in accuracy over tolerance for classification methods. It can be observed that as tolerance increases, classification accuracy also increases.

5.2 Effect of sampling size on classification accuracy

We can observe from experimental results that, as sampling size increases, classification accuracy also increases. We have tested our model with 200 samples (100 from *E. coli* and 100 from Yeast) of size 2000, 4000, 6000, 8000 and 10,000 bp separately and monitored classification accuracy. As sample/sequence size increases, a huge number of candidates are generated. Every candidate in this huge candidate database is compared with query for approximate match. Numbers of candidates generated for each candidate length say c and sample size s by sliding window procedure is (s-c-1). Finally, out of (s-c-1) candidates we will find total candidates/features that approximately match to query within given tolerance. It has been observed that, as candidates generated increases, number of fuzzy matches also increases. Feature values are very much distinguishable (for each of genome sequence data of *E. coli* and Yeast), as

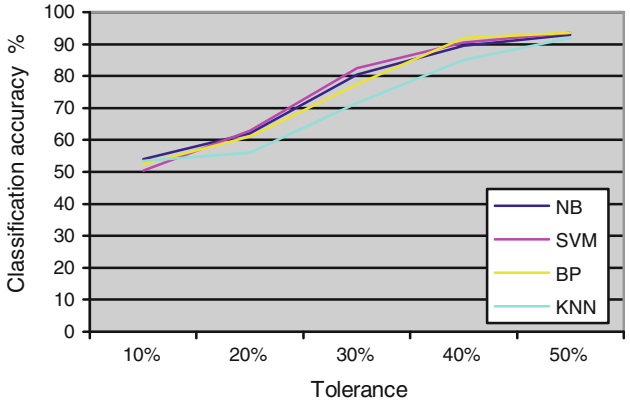

Fig. 6 Classification accuracy for sample size 6,000 at different tolerance

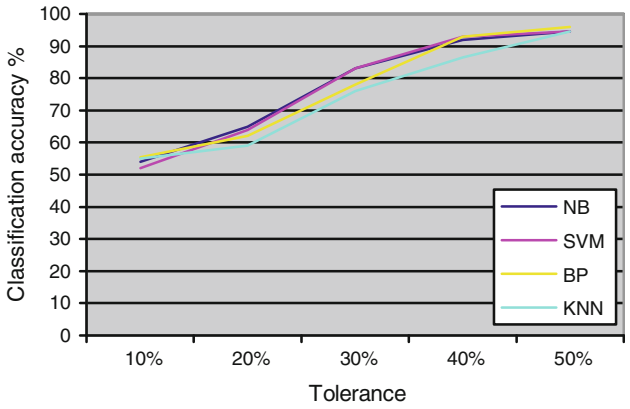

Fig. 7 Classification accuracy for sample size 8,000 at different tolerance

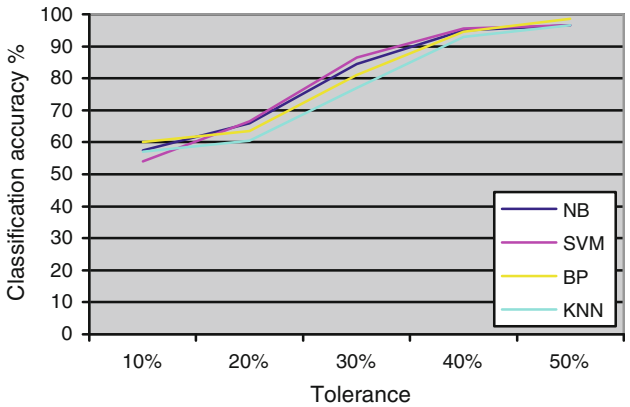

Fig. 8 Classification accuracy for sample size 10,000 at different tolerance

we start increasing sampling size. This results in increase in classification accuracy with sampling size.

Figure 9 shows, when allowed fault tolerance is 10 % sample size of 2000, 4000, 6000, 8000, and 10,000 bp

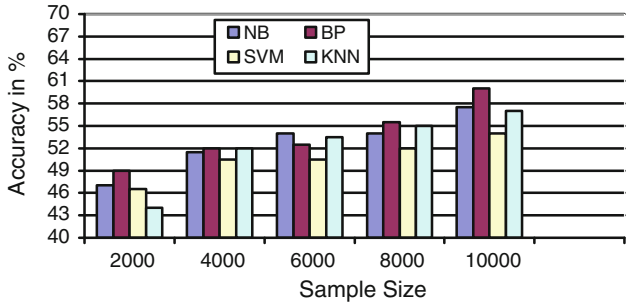

Fig. 9 Performance of different classifiers with 10 % tolerance

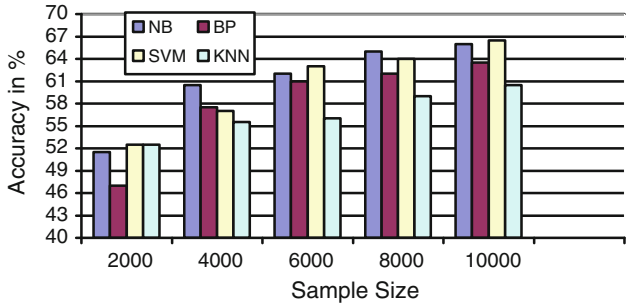

Fig. 10 Performance of different classifiers with 20 % tolerance

resulted in maximum accuracy of 49, 52, 54, 55.5, and 60 % by BP, BP and KNN, NB, BP, and BP, respectively. But, as tolerance is increased we can observe further increase in performance of classifiers with increase in sampling size as shown in Figs. 10, 11, 12 and 13. Figure 13 shows that, at 50 % tolerance, sample size of 2000, 4000, 6000, 8000, and 10,000 bp resulted in maximum accuracy of 88.5, 91, 93.5, 96, and 98.5 % by BP, SVM, BP and SVM, BP, BP, respectively.

5.3 Comparison with *n*-gram based method

The proposed approach is compared with *n*-gram sequence encoding method in binary form [14, 15]. In this method, Preprocessing consists of extracting motifs from a set of sequences. These motifs will be used as attributes/features to construct a binary table where each row corresponds to sequence. The presence or the absence of an attribute in a sequence is respectively denoted by 1 or 0. This binary table is called a *learning context*. It represents the result of the preprocessing step and the new sequence encoding format. In the mining step, a classifier is applied to the learning context to generate a classification model. The latter model is used to classify other sequences in the post-processing step.

Table 18 shows the performance of classifiers (classification accuracy in %) by using binary sequence encoding

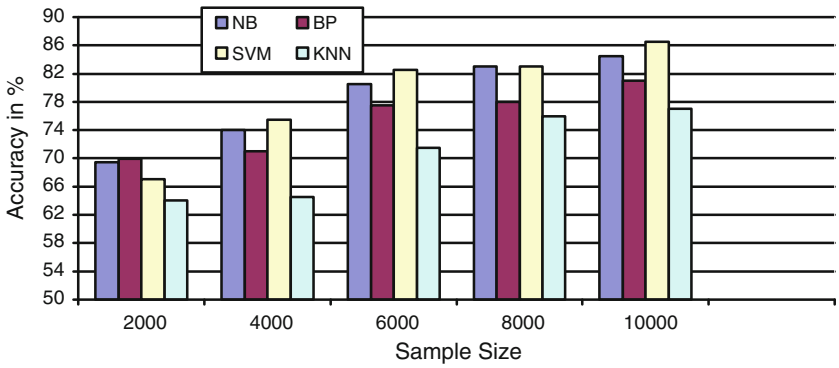

Fig. 11 Performance of different classifiers with 30 % tolerance

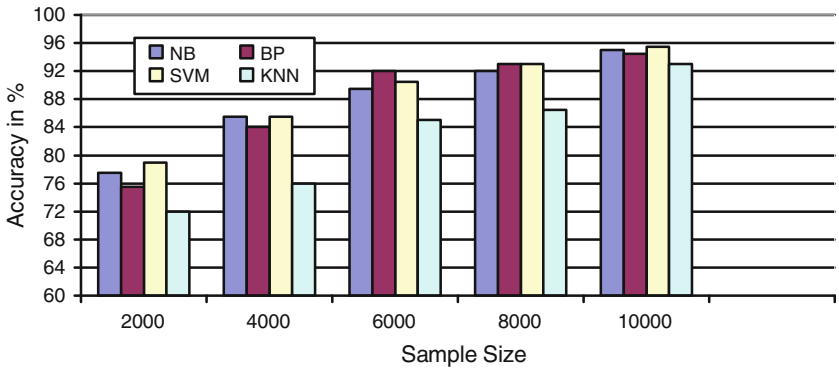

Fig. 12 Performance of different classifiers with 40 % tolerance

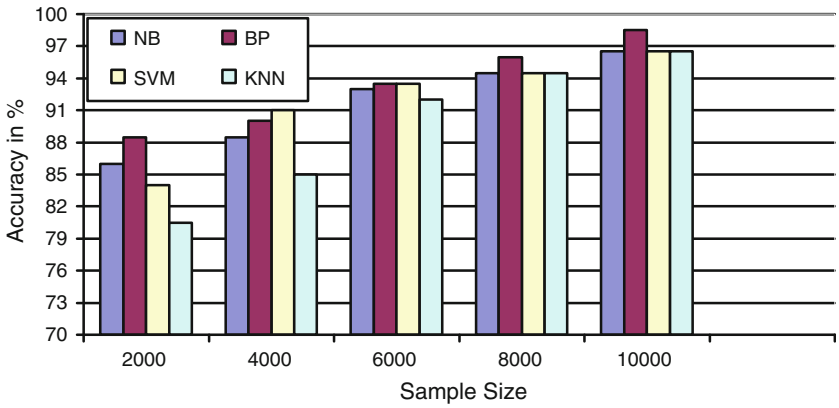

Fig. 13 Performance of different classifiers with 50 % tolerance

Table 18 Accuracy (in %) of different classifiers based on for 3-g represented in binary form

Size of sample	NB	BP	SVM	KNN
1,000	53	52.5	53	53
2,000	50	51	51	50
4,000	51	50.5	50	50
6,000	50.5	51	51	50
8,000	50	51	50	50
10,000	50	50	50	50

method for 3-g. A set of 64 3-tuple keywords is first generated using the 4 types of bases A (for Adenine), T (for Thymine), C (for Cytosine) and G (for Guanine).These keywords are searched on same 200 (that are used to verify our proposed model) randomly selected genome sequence of a given length and a binary table/feature table is constructed in which presence or the absence of an attribute in a sequence is respectively denoted by 1 or 0. We can observe that, binary encoding method for 3-g resulted in very less classification accuracy. It is because, when the database is very large and sequences are similar, the searching keywords appear at least once and hence the feature values turn to be almost 1 for every keyword. This leads to reduced classification accuracy of 50–53 %. Thus binary encoding works when each family probably has its own motifs which characterize it and distinguish it from the others.

So, we can conclude that when the sequence database is very large and sequences are partially similar, our proposed

model based on fuzzy matching is good compared to existing methods.

6 Conclusions and future work

Genomic data mining and knowledge extraction is an important problem in bioinformatics. Only a few attempts are seen in literature focusing on unknown genome identification by using either dinucleotide or trinucleotide composition. Existing approaches are based on exact pattern matching. In most of the real world biological problems exact matching may not give desired results because, biological sequences are "similar" instead of exactly same. In this paper, a novel approach for identification of species based on fuzzy patterns is proposed. Genome data of two species namely, Yeast and *E. coli* is sampled into different sizes and then fuzzy patterns with given tolerance are extracted by using Levenshtein distance. Candidate length is varied so as to allow both positive and negative tolerance. Fuzzy matches/approximate matches for these candidates/subsequences are used as feature values for the classifiers. Classification has been done by using data mining techniques namely, Naïve Bayes and support vector machine, backpropagation and nearest neighbor. To extract fuzzy matching patterns from genome sequences, we used query of length 10 and allowed tolerance from 10 to 70 %. The proposed model is tested separately for fuzzy matching patterns extracted with each of the fault tolerance and the classification accuracies are monitored. The experimental results vary according to the tolerance allowed as well as according to sampling/sequence size. Total 200 samples are used to test the model (100 samples are from *E. coli* and 100 samples are from Yeast). Our experimental results show that, the classification accuracy achieved is 98.5 % by BP, 96.5 % by other classifiers, i.e., by NB, SVM and KNN for sampling/sequence size of 10,000 bp and with allowed tolerance of 50 % and at 70 % all classifiers achieved 99 % accuracy. It can be observed from the obtained results that classification accuracy increases with increase in tolerance and sampling size. We used a query of length 10 in our experiment, in future experimental results are to be verified with different query length and a relationship between query length and tolerance values is to be established.

Appendix

Definitions

Below, we give definitions for a few important terms [46].

- Edit distance
  Edit distance between two strings refers to the minimum number of transformations required to convert one string to another, where transformations depend on the kind of edit distance studied.
  In the above definition, strings need not be of same length.
- Levenshtein distance
  Levenshtein distance is a specific instance of edit distance, where the allowed transformations are insertion, deletion and substitution.
  For example, Levenshtein distance (LD) between candidate pattern (s) and the query pattern (t) is defined as follows:
  - If strings s and t are both "test", then LD(s,t) = 0, because no transformations are needed. The strings are already identical.
  - If s is "test" and t is "tent", then LD(s,t) = 1, because one substitution (change "s" to "n") is sufficient to transform s into t.
  - If s is "levenshtein" and t is "meilenstein", then LD(s,t) = 4, because two substitutions (change "l" to "m" and change "v" to "l"), one insertion(insert "i") and one deletion (delete "h") is sufficient to transform s into t.
    The greater the Levenshtein distance, the more different the strings are.
- Pattern
  A *pattern* is a contiguous substring of length m in a string of length n, where $n > m$.
  For example, substring *abcba* is a pattern in a string *abcbaabc*.
- Fuzzy occurrence/approximate occurrence
  We say strings x and y are k-fuzzily equal if the Levenshtein distance between them is less than or equal to k. A k-fuzzy occurrence of a pattern P in a string y is any substring of y which is k-fuzzily equal to P.
  For example, consider the pattern *abcba* and the string *abbcbaabc*, with $k = 2$. Among the possible 2-fuzzy occurrences are *abbcb* (first five characters), *abbcba* (first six characters) and *bcba* (characters three to six).
  To find a particular pattern in a string, we can try all its substrings and if the edit distance with the pattern is below a certain threshold, we can take that as an occurrence.

Measures for performance evaluation

In order to evaluate performance of classification algorithms, following measures are used.

- Accuracy, sensitivity and specificity
  To understand the result of different classification techniques, the confusion matrix used for two class problems is presented in Table 5. A confusion matrix is calculated for each classification technique. The upper

row shows the actual result for the positive class and the lower row shows the result for the negative class. The true positive (TP) and true negative (TN) are correct classifications in samples of each class. A false positive (FP) is when a 'yes' sample of a class is incorrectly classified as a 'no' sample; a false negative (FN) is when a 'yes' sample of a class is classified as 'no' sample [47].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$Sensitivity = \frac{TP}{TP + FN} \quad (6)$$

$$Specificity = \frac{TN}{TN + FP} \quad (7)$$

- Kappa statistics: Kappa statistics is an index that compares the agreement between the prediction and expected outcome. Kappa can be thought of as the chance-corrected proportional agreement, where possible values range from +1 (perfect agreement) that signifies complete agreement between the classifier and the real world. Value 0 indicates no agreement, and −1 indicate complete disagreement. The Kappa value is calculated using the two-class confusion matrix values using the Eq. 8

$$K = [P(A) - P(E)]/[1 - P(E)] \quad (8)$$

where

$$P(A) = (TP + TN)/N \quad (9)$$

$$P(E) = [(TP + FN) * (TP + FP) * (TN + FN)]/N^2 \quad (10)$$

- Here N is the total number of instances used, $P(A)$ is the percentage of agreement between the classifier and underlying truth and P(E) is the chance of agreement.
- AUC (Area under the ROC Curve): The AUC is the measure of the difference between two class distributions and can be used for comparing the quality of a classifier or comparing class probability.

References

1. Baxevanis A, Ouellette FBF (1998) Bioinformatics: a practical guide to the analysis of genes and proteins. Wiley, NewYork
2. Luscombe NM, Greenbaum D, Gerstein M (2001) What is bioinformatics: a proposed definition and overview of the field. Methods Inform Med 40:346–358
3. Setubal JC, Meidanis J (1997) Introduction to computational molecular biology. PWS Publishing Company, Boston
4. Andrija T, Predrag J, Vlado K (2006) *n*-Gram-based classification and unsupervised hierarchical clustering of genome sequences. Comput Methods Programs Biomed 81:137–153
5. Hassan M, Muneer A (2008) Comparative genome sequence analysis by efficient pattern matching technique. WSEAS Trans Inform Sci Appl 12(5):1731–1740
6. Fell JW, Boekhout T, Fonseca A, Scorzetti G, Statzell-Tallman A (2000) Biodiversity and systematics of basidio-mycetous yeasts as determined by large-subunit rDNA D1/D2 domain sequence analysis. Int J Syst Evol Microbiol 50:1351–1371
7. Andréasson H, Asp A, Alderborn A, Gyllensten U, Allen M (2002) Mitochondrial sequence analysis for forensic identification using pyrosequencing technology. Biotechniques 32:124–6,128,130–3
8. Rastogi G, Dharne SM, Walujkara S, Kumara A, Patolea SM, Shouche SY (2007) Species identification and authentication of tissues of animal origin using mitochondrial and nuclear markers. Meat Sci 76:666–674
9. Gusfield D (1997) Algorithms on strings, trees, and sequences. Cambridge University Press, Cambridge
10. Navarro G (2001) A guided tour to approximate string matching. ACM Comput Surv 33:31–88
11. Hassan M, Muneer A (2009) Genome sequence analysis: a survey. J Comput Sci 5(9):651–660
12. Dictionary of cancer terms, National Cancer Institute. http://www.cancer.gov/dictionary. Accessed Mar 2005
13. A glossary of genetics, Rockfeller University. http://linkage.rockefeller.edu/wli/glossary/genetics.html. Accessed Mar 2005
14. Patil N, Toshniwal D, Garg K (2011) Species Identification Based on Approximate Matching. ACM Compute 2011
15. Saidi R, Maddouri M, Nguifo EM (2010) R protein sequences classification by means of feature extraction with substitution matrices. BMC Bioinformatics 11:175
16. Nakashima H, Ota M, Nishikawa K, Ooi T (1998) Genes from nine genomes are separated into their organisms in the dinucleotide composition space. DNA Res 5:251–259
17. Narasimhan S, Sen S, Konar A (2005) Species identification based on mitochondrial genomes. In: Proceedings of international conference of cognition and recognition, Mysore
18. Sen S, Narasimhan S, Konar A (2007) Biological data mining for genomic clustering using unsupervised neural learning. Eng Lett 14(2):61–71
19. Han J, Kamber M (2000) Data mining: concepts and techniques. Morgan Kaufmann Publishers, San Francisco
20. Gately E (1996) Neural network for financial forecasting. Wiley, New York
21. Misener S, Krawetz SA (2000) Bioinformatics: methods and protocols. Human Press Inc, Totowa
22. Mitra S, Pal SK, Mitra P (2002) Data mining in soft computing framework: a survey. IEEE Trans on Neural Networks 13:3–14
23. Sandberg R, Winberg G, Branden CI, Kaske A, Ernberg I, Coster J (2001) Capturing whole genome characteristics in short sequences using a naive Bayesian classifier. Genome Res 11:1404–1409
24. Zanoguera F, de Francesco M (2004) Protein classification into domains of life using Markov chain models. In: Proceedings of 2004 IEEE the computational systems bioinformatics conference, pp 517–519
25. Norashikin A, Damminda A, Rowena C (2010) Cluster identification and separation in the growing self-organizing map: application in protein sequence classification. Neural Comput Appl 19:531–542
26. Patil N, Toshniwal D, Garg K (2011) Clustering genome data based on approximate matching. Inderscience publisher, Geneva
27. Wei Y, Kun W, Huixiao L, Yang J, Xiaoqin W, Yaning D (2009) Classification of DNA sequences basing on the dinucleotide compositions. In: IEEE Second International Symposium on Computational Intelligence and Design, pp 390–394
28. Deschavanne P, Giron A, Vilain J, Fagot G, Fertil (1999) Genomic signature: characterization and classification of species

assessed by chaos game representation of sequences. Biol Evol 16(10):1391–1399
29. Giron A, Vilain J, Serruys C, Brahmi D, Deschavanne P, Fertil (1999) Analysis of parametric images derived from genomic sequences using neural network based approaches. In: Proceedings of international joint conference on neural networks, pp 3604–3608
30. Nair VV, Anto LP, Nair AS (2009) Naive Bayesian classification of unknown sequence fragments based on chaos game representation of mitochondrial genomes. Communications of SIWN 7:27–33
31. Weston J, Pérez-Cruz F, Bousquet O, Chapelle O, Elisseeff A, Schölkopf B (2003) Feature selection and transduction for prediction of molecular bioactivity for drug design. Bioinformatics 19:764–771
32. Hasani M, Emami F (2007) Evaluation of feed-forward back propagation and radial basis function neural networks in simultaneous kinetic spectrophotometric determination of nitroaniline isomers. Talanta Elsevier 75:116–126
33. John GH, Langley P (1995) Estimating continuous distributions in Bayesian classifiers. In: Proceedings of international conference on uncertainty in artificial intelligence, San Mateo, pp 338–345
34. De Ferrari L, Aitken S (2006) Mining housekeeping genes with a naive Bayes classifier. BMC Genomics 7:277
35. Domingos P, Pazzani M (1997) On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, Kluwer Academic, Netherlands, pp 103–130
36. Yuan X, Yang F, Peng J, Buckles BP (2003) Gene Expression Classification: Decision Trees versus SVMs. In Proceedings of sixteenth international artificial intelligence research society conference (FLAIRS 2003), pp 92–96
37. Cui Q, Jiang T, Liu B, Ma S (2004) Esub8: a novel tool to predict protein subcellular localizations in eukaryotic organisms. BMC Bioinformatics 5:66
38. Wang A, Zhao Y, Hou Y, Li Y (2010) A novel construction of SVM compound kernel function, CORD Conference Proceedings, pp 1462–1465
39. Khan M, Ding Q, Perrizo W (2002) k-Nearest neighbor classification on spatial data streams using p-trees. In: Proceedings of the 6th pacific-Asia conference on advances in knowledge discovery and data mining, Taipei
40. Dasarathy BV (1991) Nearest-neighbor classification techniques. IEEE Computer Society Press, Los Alomitos
41. Blattner FR, Plunkett G, Bloch CA, Perna NT, Burland V, Riley M (1997) The complete genome sequence of *Escherichia coli* K-12. Science 277:1453–1462
42. Cherry JM, Ball C, Weng S, Juvik G, Schimidt R, Alder C, Dunn B, Dwight S, Riles L (1997) Genetic and physical maps of *Saccharomyces cerevisiae*. Nature 387(suppl 6632):67–73
43. Genome-ftp.stanford.edu (directory/yeast/sequence/genomic_sequence/)
44. National Center for Biotechnology Information, ftp://ftp.ncbi.nlm.nih.gov/
45. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques with Java implementations. Morgan Kaufman, San Francisco
46. Kulkarni A, Noronha A. Roy S Angadi S (2010) Fuzzy pattern extraction for classification of protein sequences. In: Proceedings of the international symposium on biocomputing, New York
47. Delen D, Walker G, Kadam A (2005) Predicting breast cancer survivability: a comparison of three data mining methods. Artif Intell Med 34(2):113–127