Automatic Extraction of Highly Predictive Sequence Features that Incorporate Contiguity and Mutation

Hao Wan, Carolina Ruiz(✉), and Joseph Beck

Department of Computer Science,
Worcester Polytechnic Institute, Worcester, MA, USA
{hale, ruiz, josephbeck}@wpi.edu

Abstract. This paper investigates the problem of extracting sequence features that can be useful in the construction of prediction models. The method introduced in this paper generates such features by considering contiguous subsequences and their mutations, and by selecting those candidate features that have a strong association with the classification target according to the Gini index. Experimental results on three genetic data sets provide evidence of the superiority of this method over other sequence feature generation methods from the li-terature, especially in domains where presence, not specific location, of features within a sequence is pertinent for classification.

Keywords: Sequence feature generation · Sequence classification · Mutation

1 Introduction

Supervised sequence classification deals with the problem of learning models from labelled sequences. The resulting models are used to assign appropriate class labels to unlabelled sequences. Sequence classification methods can be used for example to predict whether a segment of DNA is in the promoter region of a gene or not.

In general, sequence classification is more difficult than classification of tabular data, mainly because of two reasons: in sequence classification it is unclear what features should be used to characterize a given data set of sequences (e.g., Fig. 1 shows three different candidate features whose presence, or lack of, in a sequence can be used to characterize the sequence); and the number of such potential features is very large. For instance, given a set of sequences of maximum length l, over an alphabet B, where $|B| = d$ symbols are considered as features, then there are d^k potential features. Furthermore, the number of potential features will grow exponentially as the length of subsequences under consideration increases, up to d^l. Thus, determining which features to use to characterize a set of sequences is a crucial problem in sequence classification.

A mutation is a change in an element of a sequence. Like in DNA sequences, this could result from unrepaired damage to DNA or from an error during sequence replication. We are interested in whether these changes affect the sequences' function. Thus, we focus on generating features that represent mutation patterns in the sequences, and on selecting the generated features that are most suitable for classification.

© Springer International Publishing Switzerland 2015
G. Plantier et al. (Eds.): BIOSTEC 2014, CCIS 511, pp. 212–229, 2015.
DOI: 10.1007/978-3-319-26129-4_14

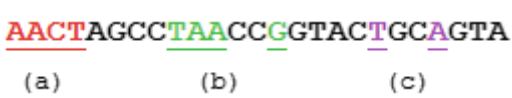

Fig. 1. Example of different candidate features for a given sequence. (a) 4-grams: AACT; (b) 4-grams with one gap of size 2; (c) 2-gapped pair (TA,2).

More specifically, our proposed Mutated Subsequence Generation (MGS) algorithm generates features from sequence data by regarding contiguous subsequences and mutated subsequences as potential features. It first generates all contiguous subsequences of a fixed length from the sequences in the data set. Then it checks whether or not each pair of candidate feature subsequences that differ in only one position should be joined into a mutated subsequence. The join is performed if the resulting joint mutated subsequence has a stronger association with the target class than the two subsequences in the candidate pair do. If that is the case, the algorithm keeps the joint subsequence and removes the two subsequences in the candidate pair from consideration. Otherwise, the algorithm keeps the candidate pair instead of the joint mutated subsequence. After all the generated candidate pairs of all lengths have been checked, a new data set is constructed containing the target class and the generated features.

The features in the resulting data set represent (possibly mutated) segments of the original sequences that have a strong connection with the sequences' function. We then build classification models over the new data set that are able to predict the function (i.e., class value) of novel sequences.

The contributions of this paper are the introduction of a new feature generation method based on mutated subsequences for sequence classification, and a comparison of the performance of our algorithm with that of other feature generation algorithms.

2 Related Work

Feature-based classification algorithms transform sequences into features for use in sequence classification [1]. A number of feature-based classification algorithms have been proposed in the literature. For example, *k*-grams are substrings of *k* consecutive characters, where *k* is fixed beforehand, see Fig. 1(a). Damashek used *k*-grams to calculate the similarity between text documents during text categorization [2]. In [3], the authors vary *k*-grams by adding gap constraints into the features, see Fig. 1(b). Another kind of feature, *k*-gapped pair, is a pair of characters with constraint $l_2 - l_1 = k$, where *k* is a constant, and l_1 and l_2 are the locations in the sequence where the characters in the pair occur, see Fig. 1(c). The *k*-gapped pair method is used to generate features for Support Vector Machines in [4, 5]. In contrast with our method, features generated by their approach cannot represent mutations in the sequences.

Another method is mismatch string kernel [6]. It constructs a (*k, m*) – mismatch tree for each pair of sequences to extract *k*-mer features with at most *m* mismatches. It then uses these features to compute the string kernel function. A similarity between this

method and our method is that both generate features that are subsequences with mutations. However, there are three major differences between them:

1. In mismatch string kernel, the features are generated from pairs of sequences and used to update the kernel matrix. In contrast, our MSG method generates features from the entire set of data sequences in order to transform the sequence data set into a feature vector data set.
2. In the process of computing candidate mutated subsequences, our MSG method does not only consider mutations in the subsequences, but also takes into account correlations between these mutated subsequences and the target classes. In contrast, the mismatch string kernel method disregards the latter part.
3. The mismatch string kernel method can be used in Support Vector Machines and other distance based classifiers for sequence data. Our MSG approach is more general as it transforms the sequences into feature vectors. In other words, the data set that results from the MSG transformation can be used with any classifier defined on feature vectors.

3 Background

3.1 Feature Selection

Feature selection methods focus on selecting the most relevant features of a data set. These methods can help prediction models in three main aspects: improving the prediction accuracy; reducing the cost of building the models; and making the models, and even the data, more understandable.

To obtain the features that maximize classification performance every possible feature set should be considered. However, exhaustively searching all the feature sets is known to be an NP-hard problem [7]. Thus, a number of feature selection algorithms has been developed based on greedy search methods like best-first and hill climbing (see [8]). These greedy algorithms use three main search strategies: *forward selection*, which starts with a null feature set and at each step adds the most important unselected feature (according to a specific metric); *backward deletion*, which starts at the full feature set and at each step removes an unimportant feature; and *bi-directional selection*, which also starts at a null feature set and at each step applies forward selection and then backward deletion until a stable feature set is reached.

3.2 CFS Evaluation

The Correlation-based Feature Selection (CFS) algorithm introduces a heuristic function for evaluating the association between a set of features and the target classes. It selects a set of features that are highly correlated with the target classes, yet uncorrelated with each other. This method was introduced in [9]. In this paper, we use this algorithm to select a subset of the generated features that is expected to have high classification performance.

3.3 Gini Index

In decision tree induction, the Gini index is used to measure the degree of impurity of the target class in the instances grouped by a set of features [10]. Similarly in our research, the Gini index is used to measure the strength of the association between candidate features and the target class. Specifically, we use it during feature generation to determine whether or not to replace a pair of subsequences (candidate features) that differ just in one position with their joined mutated subsequence. Details are described in Sect. 4.1. The Gini index of a data set is defined as: $1 - \sum_{c \in C} p(c)^2$, where C is the set of all target classes, and p denotes probability (estimated as frequency in the data set). Given a discrete feature (or attribute) X, the data set can be partitioned into disjoint groups by the different values of X. The Gini index can be used to calculate the impurity of the target class in each of these groups. Then, the association between X and the target class can be regarded as the weighted average of the impurity in each of the groups:$Gini(X) = \sum_{x \in X} p(x) * (1 - \sum_{c \in C} p(c|x)^2)$.

4 Our MSG Algorithm

4.1 Feature Generation

Our Mutated Subsequence Generation (MSG) algorithm belongs in the category of feature-based sequence classification according to the classification methods described in [1]. The MSG algorithm transforms the original sequences into contiguous subsequences and mutated subsequences, which are used as candidate features in the construction of classification models.

The MSG algorithm generates features of different lengths according to two user-defined input parameters: the minimum length $l_{\min}$ and the maximum length $l_{\max}$. It first generates the candidate subsequences of length $l_{\min}$, then length $l_{\min} + 1$, and all the way to the subsequences of length $l_{\max}$. Then it takes the union of all the generated subsequences of different lengths. Finally, the MSG algorithm constructs a new data set containing a Boolean feature for each generated subsequence (see Table 2 for an example). Each sequence in the original data set is represented as a vector of 0's and 1's in the new data set, where a feature's entry in this vector is 1 if the feature's corresponding subsequence is present in the sequence, and 0 if not.

The feature generation process consists of five main steps described below. Figure 2 shows an example of the transformation of a sequence data set into the subsequence features, from Step *a* to Step *d*.

(a) The MSG algorithm generates all the contiguous subsequences of a specific length from each original sequence;
(b) It divides the contiguous subsequences into n categories according to which class they are most frequent in, where n is the number of different classes;
(c) For each category, it generates mutated subsequences based on the Gini index;
(d) It combines together all the features from each category;

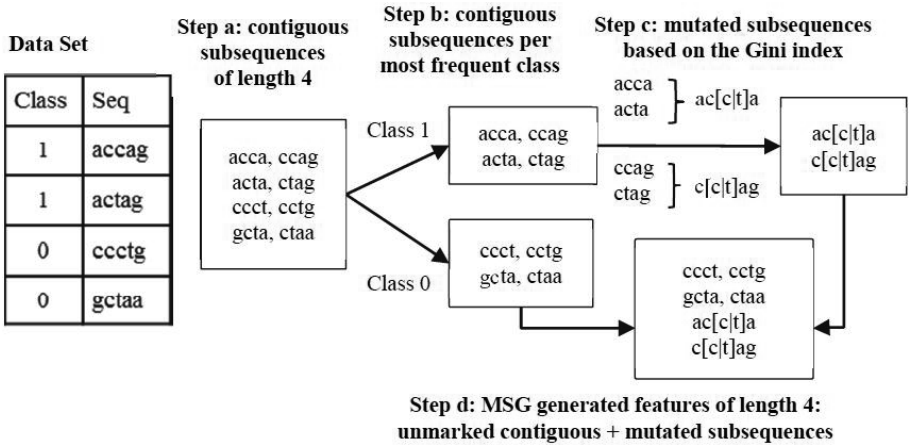

Fig. 2. Illustration of the mutated subsequence generation process from Step (a) to Step (d). In this example, features of length 4, including mutated subsequences and contiguous subsequences, are obtained. Here, the Gini index of each of the mutated subsequences ac[c|t]a and c[c|t]ag is better than the Gini indexes of its forming contiguous subsequences.

(e) It repeats Step *a* to Step *d* with a different length, until features of all the lengths in the user defined range are generated. Then it combines these features prior to constructing the new data set.

Computation of Contiguous Subsequences. Given $k \geq 1$, a *contiguous subsequence* of length k is a subsequence formed by taking k consecutive symbols from the original sequence. For example, *gc* is a contiguous subsequence of length 2 in the sequence *gcta*, while *gt* is not.

The MSG algorithm starts with the generation of contiguous subsequences. Suppose that the original sequences have length t, and that the contiguous subsequences to be computed have length k. First, each sequence is traversed and all of its contiguous subsequences of k symbols are extracted from starting locations 1 through $(t - k + 1)$. Then, duplicate subsequences are removed. For example, Table 1 shows the contiguous subsequence features of length 4 for the data set in Fig. 2, together with the number of occurrences of each feature in data set sequences of a given target class.

Separation. In this step, the algorithm separates the contiguous subsequences into n is the number of different classes in the data set. Each subsequence is assigned to a category according to which class it is most frequent in. For example, the first four features in Table 1 are assigned to Category 1, because they are more frequent in Class 1 than in Class 0, and the other four to Category 2. If a subsequence is maximally frequent equally in more than one class, it is randomly assigned to one of those classes.

Generation of Mutated Subsequences. In this step, the MGS algorithm generates mutated subsequences. The following terminology will be used in the description of this step.

Table 1. Contiguous subsequences of length 4 of the data sequences in Fig. 2, and their frequency counts in each data set class.

Feature	Class 0	Class 1
acca	0	1
ccag	0	1
acta	0	1
ctag	0	1
cact	1	0
actg	1	0
gcta	1	0
ctaa	1	0

- *Mutated Subsequence:* A mutated subsequence is a subsequence of length k ($k \geq 1$) which contains r mutative positions, where $1 \leq r \leq k$. In each mutative position, there are two or more substitutions. In this paper, we consider mutated subsequences with only one mutative position. For example, in the subsequence $g[c|a]t$, the second position is a mutative position, with two possible substitutions c and a. Thus, a mutated subsequence has many instantiations, in our example *gct* and *gat*. We say that a mutated subsequence is contained in an original sequence when any of its instantiations is contained in the original sequence. For example, $g[c|a]t$ is contained in the sequence *gata* as well as in the sequence *gata*.
- *Candidate Pair:* a candidate pair is a pair of subsequences which are different from each other in only one position. For instance, *acca* and *acta* is a candidate pair. The candidate pair could also contain mutated subsequences, like for instance *acca* and $ac[t|g]a$.
- *Joinable Checking:* joinable checking is used to determine whether or not to join together a candidate pair of subsequences depending on their correlation with the target class. In this paper, we use the Gini index to measure this correlation. For example, suppose *sub*1 is *acca* and *sub*2 is *acta* in Table 1. Their joint sequence *sub*3 is $ac[c|t]a$. The original data set in Fig. 2 can be split into two groups by each one of the subsequences in the pair: one group consists of the original data sequences which contain the subsequence, marked as $group_1^{subi}$; the other group consists of the rest of data sequences, marked as $group_2^{subi}$. Taking *sub*1 as an example, there is only one sequence in $group_1^{sub1}$, which is in class 1 (*accag*); and there are three sequences in $group_2^{sub1}$, two of which are in class 0 (*ccctg* and *gctaa*), and the other one is in class 1 (*actag*). Thus, by using the formula in Sect. 3.3, we can calculate the Gini index for *sub*1 as followings: $impurity(g_1^{sub1}) = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0$; $impurity(g_2^{sub1}) = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0.444$; $Gini(sub1) = p(g_1^{sub1}) * impurity(g_1^{sub1}) + p(g_2^{sub1}) * impurity(g_2^{sub1}) = 0 * \frac{1}{4} + 0.44 * \frac{3}{4} = 0.333$. Similarly, we calculate the values $Gini(sub2) = 0.333$, $Gini(sub3) = 0$. Since *sub*3 has the best measure value, implying that it has the strongest association with the target class, then the candidate pair *sub*1 and *sub2* is joinable.

```
Function Mutated_Subsequences_Generation(C):
•  Input: C: set of subsequences
•  Output: set of all features, contiguous and
   mutated subsequences, that can be generated from C

Initialization: S <- {}
for each pair of subsequences sub1 and sub2 in C do
    if sub1 and sub2 is a joinable pair then
        sub3 <- sub1 ⊕ sub2
        mark sub1
        mark sub2
        for each sequence subi in C do
            if sub3 and subi is a joinable pair then
                sub3 <- sub3 ⊕ subi
                mark subi
        S <- S ∪ {sub3}
for each sequence subj in C do
     if subj is marked then
         C <- C - subj
return C ∪ S
```

Fig. 3. The pseudo code for mutated subsequences generation. $\oplus$ is the join operator. For example, $acca \oplus acta = ac[c|t]a$ and $acca \oplus ac[t|g]a = ac[c|t|g]a$.

The MSG algorithm performs joinable checking on every candidate pair within each category. Once a candidate pair is determined to be joinable, then the two subsequences are joined together to create a new subsequence, called *subi*, and they are also marked. Then the joinable checking is performed on *subi* with every other subsequence in the category. If there are other subsequences that are joinable with, then they are also joined together with *subi* and marked. Finally, *subi* is added into the mutated subsequence set. After all candidate pairs are checked, the algorithm deletes the marked subsequences and the duplicate mutated subsequences. A simplified pseudo code of mutated subsequences generation is shown in Fig. 3.

Combination of Categories. In the previous step, some contiguous subsequences might remain intact. That is, they are not joined with other subsequences. Thus, in each category, there might be two types of features: mutated subsequences and unchanged contiguous subsequences. In this step, the algorithm combines the feature sets of all categories together into one feature set. In this set, all features have length t, as defined in Step (b).

Combination of Features of Different Lengths. After all the features of the lengths in the user-defined range are generated in Step (a) through Step (d), the MSG algorithm combines these features of different lengths together and constructs a new data set. Each data instance in the new data set corresponds to a sequence in the original data set. The instance's value for each feature is a Boolean value stating whether or not the feature is a subsequence of the instance. Table 2 shows the transformed data from the data set in Fig. 2 for the subsequence length range 3−4.

Table 2. Transformed data set obtained by applying the MSG algorithm to the data set in Fig. 2, with subsequence length range 3−4. Original data sequences are depicted as columns and extracted features as rows due to formating restrictions.

	accag	actag	ccctg	gctaa
taa	0	0	0	1
gct	0	0	0	1
ctg	0	0	1	0
cac	0	0	1	0
act	0	1	1	0
cca	1	0	0	0
acc	1	0	0	0
[t|c]ag	1	1	0	0
ctaa	0	0	0	1
gcta	0	0	0	1
actg	0	0	1	0
cact	0	0	1	0
c[t|c]ag	1	1	0	0
ac[t|c]a	1	1	0	0
Class	1	1	0	0

4.2 Feature Selection

Once that the set of candidate features has been generated as described in Sect. 4.1, we use bi-directional feature selection based on the CFS evaluation (see Sect. 4) to select the best feature set from the transformed data set. This feature set is then used to build classification models.

5 Experimental Evaluation

The performance of our MSG algorithm is compared with that of other feature generation algorithms which are commonly used for sequence classification. These algorithms are described below.

- **Position-based** [11]: Each position is regarded as a feature, the value of which is the alphabet symbol in that position.
- ***k*-grams** [4]: A k-gram is a sequence of length k over the alphabet of the data set (see Fig. 1(a)). The value of a k-gram induced feature for a sequence S is whether the k-gram occurs in S or not.
- ***k*-gapped Pair** [12]: In a k-gapped pair (xy, k), xy is an ordered pair of letters over the alphabet of the data set, k is a non-negative integer (see Fig. 1(c)). The value of a k-gapped pair induced feature for a sequence S is 1 if there is a position i in S, where $s_i = x$ and $s_{i+k+1} = y$. Otherwise, the value is 0.

Notice that a k-gapped pair can be regarded as a mutated subsequence where the k symbols between the pair can be any nucleotides. Thus, in order to perform a

meaningful comparison, we use the same subsequence length values for the three generation algorithms (k-grams, MSG, and k-gapped pair) in our experiments. Moreover, we use values of $k > 5$ to obtain features, since for $k \leq 5$ most of the k-gapped pairs are contained in all sequences, making them useless for classification.

To compare the performance of these algorithms, a number of experiments are carried out on three data sets, the first two are collected from UCI Machine Learning Repository [13], and the third one was collected in our prior work [14] from WormBase [15]:

- ***E.coli* Promoter Gene Sequences Data Set:** This data set consists of 53 DNA promoter sequences and 53 DNA non-promoter sequences. Each sequence has length 57. Its alphabet is {a, c, g, t}.
- **Primate Splice-junction Gene Sequences Data Set:** This data set contains 3190 DNA sequences of length 60. Also, its alphabet is {a, c, g, t}. Each sequence is in one of three classes: exon/intron boundaries (EI), intron/exon boundaries (IE), and non-splice (N). 745 data instances are classified as EI; 751 instances as IE; and 1694 instances as N.
- ***C.elegans* Gene Expression Data Set:** This data set contains 615 gene promoter sequences of length 1000.We use here expression in EXC cells as the classification target. 311 of the genes in this data set are expressed in EXC cells, and the other 304 genes are not.

The performance comparison in the following sections focuses on the prediction level of models built on the generated features, and on differences among the models. We implemented the four feature generation methods, MSG, k-grams, position-based, and k-gapped pairs, in our own Java code. To measure the prediction level, we utilize The WEKA System version 3.7.7 [16] to build three types of prediction models: J48 Decision Trees, Support Vector Machines (SVMs), and Logistic Regression (LR). We use n-fold cross validation to test the models. We regard the models' accuracy as their prediction level. To measure the difference between two models, we perform a pair t-test on their n-fold test results, and use p-values from the paired t-test to determine whether or not the difference in model performance is statistically significant.

5.1 Results on the *E.Coli* Promoter Gene Sequences Data Set

Patterns from the Literature. As found in the biological literature [17, 18], and summarized in [19], promoter sequences share some common DNA segments. Figure 4 presents some of these segments. As can be seen in the figure, these segments can contain mutated positions. Also the segments are annotated with specific locations where the segments occur in the original sequences. This is an important characteristic distinguishing these segments from the subsequences generated by our algorithm. The data set constructed by our MSG algorithm captures only presence of the subsequences (not their positions) in the original sequences.

However, after examining the occurrences of the aforementioned segments in the Promoter Gene Sequences data set, we found that for the most part each segment occurs at most once in each sequence. Hence computational models created over this

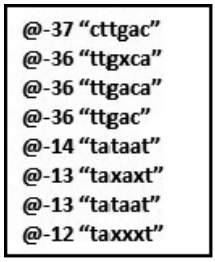

Fig. 4. Patterns taken from [19]. Promoter sequences share these segments at the given locations. In these segments, "x" represents the occurrence of any nucleotide. The location is specified as an offset from the Start of Transcription (SoT). For example, "-37" refers to the location 37 base pair positions upstream from SoT.

data set that deal only with presence of these patterns are expected to achieve a prediction accuracy similar to that of computational models that take location into consideration. Therefore, since the precise location of the patterns seems to be irrelevant, we expect our MSG algorithm to perform well on this data set.

Experimental Results. Each of the four feature generation methods under consideration (MSG, *k*-grams, position-based, and *k*-gapped pair) was applied to the Promoter Gene Sequence data set separately, yielding four different data sets. Parameter values used for the feature generation methods were the following: for MSG, the range for the length of transformed subsequences was 1−5; for *k* -grams, $k \leq 5$; and for *k*-gapped, $k \leq 10$. Then, Correlation-based Feature Selection (CFS), described in Sect. 3.2, was applied to each of these data sets to further reduce the number of features. The resulting number of features in each of the data sets was: 61 for the MSG transformed data set, 43 for *k*-grams, 7 for position-based, and 29 for *k*-gapped pair. 5-fold cross validation was used to train and test the models constructed on each of these data sets. Three different model construction techniques were used: J4.8 decision trees, Logistic Regression (LR), and SVMs. Figure 5 shows the prediction accuracy of the obtained models. Table 3 depicts the statistical significance of the performance difference between the models constructed over the MSG-transformed data set and the models constructed over data sets constructed by other feature generation methods.

From Fig. 5, we can observe that the prediction levels of the models constructed over features generated by MSG are superior to those of models constructed on other features. The t-test results in Table 3 indicate that this superiority of MSG is statistically significant at the $p < 0.05$ level in the cases highlighted in the table. As expected, the MSG algorithm generates a highly predictive collection of features for this data set. This in part due to the fact that for this data set, the presence alone, and not location, of certain subsequences (or segments) discriminates well between promoter and non-promoter sequences. Table 4 shows some of the features generated by MSG.

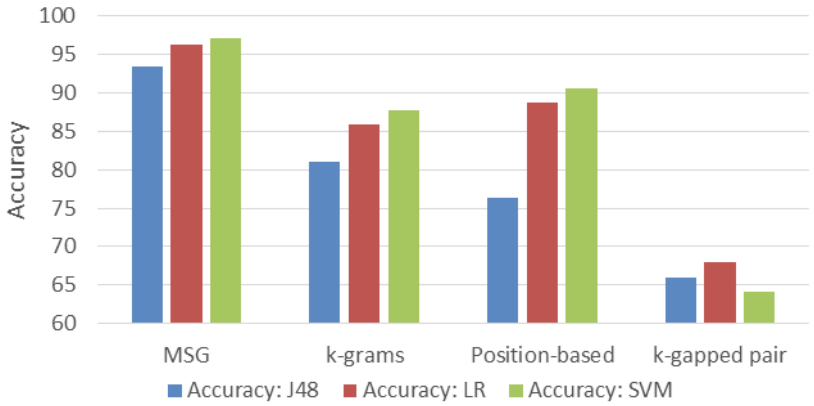

Fig. 5. Accuracy of models built on the features from four different feature generation methods (MSG, *k*-gram, Position-based, and *k*-gapped pair), using three classification algorithms on the Promoter Gene Sequences data.

Table 3. p-values obtained from t-tests comparing the prediction accuracies of models constructed over MSG-generated features and models constructed over data sets generated by the other 3 feature generation methods. t-tests were performed using 5-fold cross-validation over the Promoter Gene Sequence data set. Highlighted in the table are the cases in which the superiority of MSG is statistically significant at the $p < 0.05$ level.

Baseline: MSG	*k*-grams	Position-based	*k*-gapped pair
p-value: J48	0.08	0.02	0.03
p-value: LR	0.01	0.06	0.01
p-value: SVM	0.08	0.08	0.01

5.2 Results on the Primate Splice-Junction Gene Sequences Data Set

Patterns from the Literature. Some patterns in this data set have been identified in [20]. These patterns state that a sequence is in EI or IE if the nucleotide triplets "TAA", "TAG", and "TGA", known as stop codons, are absent in certain positions of the sequence. Conversely, if a sequence contains any stop codons in certain specified positions, the sequence is not in EI (IE). To examine the effect of position on these patterns, we generated the rules below, and calculated their confidence (i.e., prediction accuracy) on this data set.

- *Stop codons are present* → *not EI (74 %)*
- *Stop codons are present at specified positions* → *not EI (95 %)*
- *Stop codons are present* → *not IE (77 %)*
- *Stop codons are present at specified positions* → *not IE (91 %)*

As can be seen, the position information is very important in these patterns. Hence, we might expect that the MSG algorithm will not perform well on this data set, because

Table 4. Sample features constructed by the MSG algorithm over the Promoter Gene Sequences data set together with their correlation with the class feature.

MSG features	Correlation with target
t[t|a]ta	-0.61
[a|c]aaa	-0.58
ta[a|g|c|t]aa	-0.58
a[a|g|c|t]aat	-0.56
ata[t|c|a|g]t	-0.53
at[g|a|c]at	-0.51
aatt[c|a|t|g]	-0.51
aaa[g|a|t|c]t	-0.5
tta[t|a|c]a	-0.44
aaa[t|c|a|g]c	-0.44
ccc[a|g]	-0.41
ct[g|t|c|a]tt	-0.41
at[t|a]	-0.37
c[a|c|g|t]ggt	0.42
tgag[g|a]	0.43

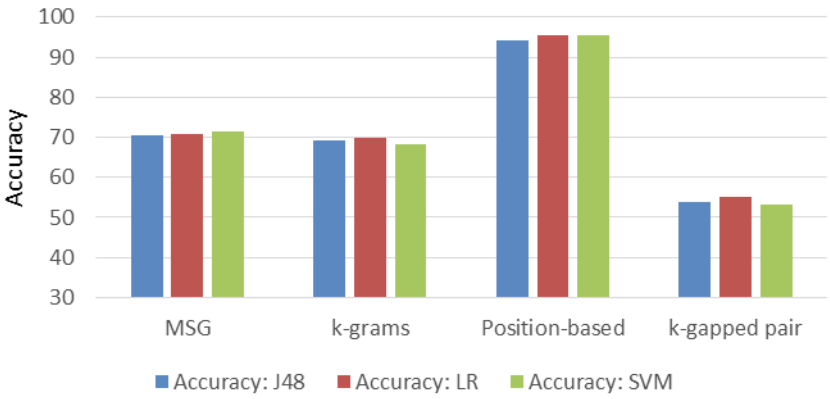

Fig. 6. Accuracy of models built on the features from four different feature generation methods (MSG, *k*-gram, Position-based, and *k*-gapped pair), using three classification algorithms on the Splice-junction Gene Sequence data.

its generated features do not contain information about the location where subsequences appear in the original sequences.

Experimental Results. Once again, each of the four feature generation methods under consideration (MSG, *k*-grams, position-based, and *k*-gapped pair) was applied to the Splice-junction Gene Sequences data set separately, yielding four different data sets.

The parameters used for the feature generation algorithms were: for MSG, the range for the length of transformed subsequences was 1−5; for *k*-grams, $k \leq 5$; and for *k*-gapped, $k \leq 10$. Then, Correlation-based Feature Selection (CFS), described in Sect. 3.2, was applied to each of these resulting data sets. The size of the feature set generated by the MSG algorithm was 28, by *k*-grams was 29, by position-based was 22, and by *k*-gapped pair was 49.

10-fold cross validation was used to construct and test models over these four data sets. Average accuracies of the resulting models are shown in Fig. 6, and the t-test results in Table 5. On this data set, the position-based algorithm performed the best. This is expected given that location information is relevant for the classification of this data set's sequences, as discussed above. The MSG generated features yielded prediction performance at the same level of that of *k*-grams; and statistically significantly higher performance (at the $p < 0.05$ significance level) than that of *k*-gapped pair. Some of these MSG generated features are shown in Table 6.

Table 5. p-values obtained from t-tests comparing the prediction accuracies of models constructed over MSG-generated features and models constructed over data sets generated by the other 3 feature generation methods. t-tests were performed using 10-fold cross-validation over the Splice-junction Gene Sequences data set. Highlighted in the table are the cases in which the superiority of MSG is statistically significant at the $p < 0.05$ level.

Baseline: MSG	*k*-grams	Position-based	*k*-gapped pair
p-value: J48	0.23	6.5E-18	2.12E-13
p-value: LR	0.72	6.78E-16	7.3E-13
p-value: SVM	0.15	2.3E-17	3E-14

Table 6. Sample features constructed by the MSG algorithm over the Splice-junction Gene Sequences data set together with their correlation with the class feature.

MSG features	Correlation with target
gt[a|g]ag	-0.5
ggt[a|g]a	-0.44
ggt[a|g]	-0.38
gt[a|g]a	-0.35
gtg[c|a]g	-0.35
aggt[a|g]	-0.35
gta[g|a]g	-0.34
[g|t|a]ggta	-0.32
tc[t|c]t	-0.03
[t|c]ag	-0.02
t[c|t]tc	0.01

Table 7. A PWM for PHA-4, found in [23]. It records the likelihood of each nucleotide at each position of the PHA-4 motifs.

	A	C	G	T
1	0.097	0.144	0.52	0.238
2	0.003	0.755	0.003	0.238
3	0.003	0.097	0.003	0.896
4	0.003	0.896	0.097	0.003
5	0.003	0.99	0.003	0.003
6	0.849	0.003	0.144	0.003
7	0.99	0.003	0.003	0.003
8	0.614	0.05	0.191	0.144

5.3 Results on the *C.Elegans* Gene Expression Data Set

Patters from the Literature. Motifs are short subsequences in the promoter sequences that have the ability to bind transcription factors, and thus to affect gene expression. For example, a transcription factor CEH-6 is necessary for the gene aqp-8 to be expressed in the EXC cell, by binding to a specific subsequence (ATTTGCAT) in the gene promoter region [22]. The binding sites for a transcription factor are not completely identical, as some variation is allowed. These potential binding sites are represented as a position weight matrix (PWM), see Table 7 for an example. A motif is a reasonable matching subsequence according to a specific PWM.

It has been shown that motifs at different positions in the promoter have different importance in controlling transcription [23], and that the order of multiple motifs and the distance between motifs can also affect gene expression [14].

Experimental Results. Each of the four feature generation methods under consideration (MSG, k-grams, position-based, and k-gapped pair) was applied to the *C.elegans* Gene Expression data set separately, yielding four different feature vector data sets. The parameters used for the feature generation algorithms were: for MSG, the range for the length of transformed subsequences was 1−6; for k-grams, $k \leq 6$; and for k-gapped, $k \leq 10$. After CFS, the size of the feature set generated by the MSG algorithm was 97, by k-grams was 63, by position-based was 122, and by k-gapped pair was 4. The average accuracies of the resulting models with 10-fold cross validation are shown in Fig. 7.

On this data set, MSG achieved the best performance among the methods tested. The p-values in Table 8 indicate that MSG prediction performance is significantly better than those of position-based and k-gapped pair at the $p < 0.05$ significance level. MSG performed slightly better than k-grams, but not significantly better.

5.4 Discussion

Computational Complexity Comparison of the Methods. Suppose that a data set consists of n sequences of length l over an alphabet B, where $|B| = d$ (for the three data

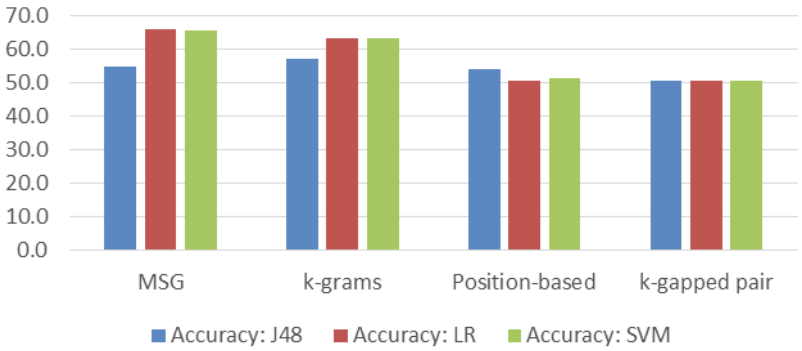

Fig. 7. Accuracy of models built on the features from four different feature generation methods (MSG, *k*-gram, Position-based, and *k*-gapped pair), using three classification algorithms on the Gene Expression data.

Table 8. p-values obtained from t-tests comparing the prediction accuracies of models constructed over MSG-generated features and models constructed over data sets generated by the other 3 feature generation methods. t-tests were performed using 10-fold cross-validation over the Gene Expression data set. Highlighted in the table are the cases in which the superiority of MSG is statistically significant at the $p < 0.05$ level.

Baseline: MSG	*k*-grams	Position-based	*k*-gapped pair
p-value: J48	0.18	0.41	0.01
p-value: LR	0.18	6.89E-6	1.45E-5
p-value: SVM	0.21	8.39E-5	4.59E-5

sets considered in this paper, $d = 4$). The position-based method has the lowest computational complexity out of the four feature generation methods employed in this paper. It takes $O(l)$ time to extract each location as a feature for each sequence, so its total complexity is $O(nl)$. The *k*-gapped pair method needs to compute $l - k - 1$ pairs of symbols for each sequence for a given gap size k. In our experiments, we considered pairs with gap $\leq k$, and since $k \ll l$, the time complexity for each sequence is $O(kl)$. Its total complexity is $O(nkl)$. Similarly, the *k*-gram method takes $O(nkl)$ time complexity to generate features of length $\leq k$.

The MSG method has the highest computational complexity among the four methods. Suppose that m subsequences are given as input to MSG (pseudo code in Fig. 3). There are two outer loops and one inner loop in this process. The first outer loop goes over all the $\binom{m}{2}$ pairs of subsequences, and its inner loop takes at most m iterations. The second outer loop traverses m subsequences to delete the marked ones. So the time complexity of this method is $C\left(m * \binom{m}{2} + m\right) = o(m^3)$ subsequences from the sequence data. Thus, its computational complexity is $O(d^{3k})$ in the worst case.

Experimental Comparison of the Methods. The experimental results on the three data sets provide evidence of the usefulness of the MSG algorithm. As we discussed above, patterns in the *E.coli* promoter gene sequences data set are position-independent, while patterns in the primate splice-junction gene sequences data set are position-dependent. Given that the MSG-generated features do not take location into consideration, MSG was expected to perform very well on the first data set but not on the second data set. Our experimental results confirm this hypothesis. In summary, MSG-generated features are most predictive in domains in which location is irrelevant or plays a minor role. Nevertheless, even in domains in which location is important, our MSG algorithm performed at the same level, or higher, than other feature generation algorithms from the literature.

In the *C.elegans* gene expression data set, patterns are much more complex than in the other two data sets considered. The MSG algorithm does not produce high classification accuracies on this data set. However, when compared to the other algorithms under consideration, MSG generates features that yield more accurate prediction models. One aspect that contributes to MSG's comparably better performance on this data set is its ability to represent mutations in the data sequences.

6 Conclusion and Future Work

In this work we present a novel feature generation method for sequence classification, Mutated Subsequence Generation (MSG). This method considers subsequences, possibly containing mutations, as potential features for the classification of the original sequences. It uses the Gini index to select the best features. We compare this method with other feature generation methods on three genetic data sets, focusing on the accuracy of the classification models built on the generated features. The experimental results show that MSG outperforms other feature generation methods in domains where presence, not specific location, of features within a sequence is relevant; and can perform at the same level or higher than other non-position-based feature generation methods in domains in which specific location, as well as presence, is important. Additionally, MSG is capable of identifying one-position mutations in the subsequence features that are highly associated with the classification target.

Future work includes further experimentation on much larger data sets; refinement of our MSG algorithm to reduce its time complexity; extension of MSG to allow for mutations in more than one subsequence position; and investigation of approaches to and the effects of incorporating location information in the MSG generated features.

References

1. Xing, Z., Pei, J., Keogh, E.: A brief survey on sequence classification. ACM SIGKDD Explor. **12**(1), 40–48 (2010)
2. Damashek, M.: Gauging similarity with n-grams: language-independent categorization of text. Science **267**(5199), 843–848 (1995)

3. Ji, X., Bailey, J., Dong, G.: Mining minimal distinguishing subsequence patterns with gap constraints. In: Proceedings of the Fifth IEEE International Conference on Data Mining (2005)
4. Chuzhanova, N.A., Jones, A.J., Margetts, S.: Feature selection for genetic sequence classification. Bioinformatics **14**(2), 139–143 (1998)
5. Huang, S.-H., Liu, R.-S., Chen, C.-Y., Chao, Y.-T., Chen, S.-Y.: Prediction of outer membrane proteins by support vector machines using combinations of gapped amino acid pair compositions. In: Proceedings of the 5th IEEE Symposium on Bioinformatics and Bioengineering (BIBE 2005) (2005)
6. Leslie, C.S., Eskin, E., Cohen, A., Weston, J., Noble, W.S.: Mismatch string kernels for discriminative protein classification. Bioinformatics **20**(4), 467–476 (2004)
7. Amaldi, E., Kann, V.: On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. Theor. Comput. Sci. **209**(1–2), 237–260 (1998)
8. Kohavi, R., Johnb, G.H.: Wrappers for feature selection. Artif. Intell. **97**(1–2), 273–324 (1997)
9. Hall, M.A., Smith, L.A.: Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. In: Proceedings of the Twelfth International FLAIRS Conference, Orlando (1999)
10. Gini, C.: Italian: Variabilità e mutabilità "(Variability and Mutability)," C. Cuppini, Bologna, p. 156. In: Pizetti, E., Salvemini, T. (eds.) Memorie di metodologica statistica. Libreria Eredi Virgilio Veschi, Rome (1912) (1955, reprinted)
11. Dong, G., Pei, J.: Sequence Data Mining, pp. 47–65. Springer, US (2009)
12. Park, K.-J., Kanehisa, M.: Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. Bioinformatics **19**(13), 1656–1663 (2003)
13. Bache, K., Lichman, M.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine (2013). (http://archive.ics.uci.edu/ml)
14. Wan, H., Barrett, G., Ruiz, C., Ryder, E.F.: Mining association rules that incorporate transcription factor binding sites and gene expression patterns in C. elegans. In: Proceeding Fourth International Conference on Bioinformatics Models, Methods and Algorithms BIOINFORMATICS2013, pp. 81–89. SciTePress, Barcelona (2013)
15. WormBase, 1 April 2012. (http://www.wormbase.org/)
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. **11**(1), 10–18 (2009)
17. Hawley, D.K., McClure, W.R.: Compilation and analysis of Escherichia coli promoter DNA sequences. Nucleic Acids Res. **11**(8), 2237–2255 (1983)
18. Harley, C.B., Reynolds, R.P.: Analysis of E. coli promoter sequences. Nucleic Acids Res. **15**(5), 2343–2361 (1987)
19. Towell, G.G., Shavlik, J.W., Noordewier, M.O.: Refinement of approximate domain theories by knowledge-based neural networks. In: Proceedings of the Eighth National Conference on Artificial Intelligence (1990)
20. Noordewier, M.O., Towell, G.G., Shavlik, J.W.: Training knowledge-based neural networks to recognize genes in DNA sequences. Adv. Neural Inf. Process. Syst. **3**, 530–536 (1991)
21. Mah, K., Tu, D.K., Johnsen, R.C., Chu, J.S., Chen, N., Baillie, D.L.: Characterization of the octamer, a cis-regulatory element that modulates excretory cell gene-expression in Caenorhabditis elegans. BMC Mol. Biol. **11**(1), 19 (2010)
22. Reece-Hoyes, J.S., Shingles, J., Dupuy, D., Grove, C.A., Walhout, A.J., Vidal, M., Hope, I.A.: Insight into transcription factor gene duplication from Caenorhabditis elegans Promoterome-driven expression patterns. BMC Genom. **8**(1), 27 (2007)

23. Ao, W., Gaudet, J., Kent, W., Muttumu, S., Mango, S.E.: Environmentally induced foregut remodeling by PHA-4/FoxA and DAF-12/NHR. Science **305**, 1743–1746 (2004)
24. Tan, P.-N., Kumar, V., Steinbach, M.: Introduction to Data Mining. Addison-Wesley, Boston (2005)