Maximum margin classifier
working in a set of strings

Hitoshi Koyano,¹ Morihiro Hayashida,² and Tatsuya Akutsu²

¹Department of Data Science, Institute for Advancement of Clinical and
Translational Science, Graduate School of Medicine, Kyoto University
²Bioinformatics Center, Institute for Chemical Research, Kyoto University

Abstract

Numbers and numerical vectors account for a large portion of data. Recently, however, the amount of string data generated has increased dramatically. Consequently, classifying string data is a common problem in many fields. The most widely used approach to this problem is to convert strings into numerical vectors using string kernels and subsequently apply a support vector machine that works in a numerical vector space. However, this non-one-to-one conversion involves information loss and makes it impossible to evaluate, using probability theory, the generalization error of a learning machine, considering that the given data to train and test the machine are strings generated according to probability laws.

We approach this classification problem by constructing a classifier that receives the strings themselves as inputs. To evaluate the generalization error of such a classifier theoretically, probability theory for strings is required. A string is an object of computer science rather than mathematics, and probability theory for strings has not been constructed. However, one of the authors and his colleague, in previous studies, first developed a probability theory on a metric space of strings provided with the Levenshtein distance and applied it to analyze biological sequences. Combined with this probability theory, our approach enables a theoretical analysis of the generalization error of a classifier working in a set of strings. We demonstrate that our learning machine classifies strings in an asymptotically optimal manner. Furthermore, we demonstrate the usefulness of our machine in practical data anal-
ysis by applying it to predictions of protein-protein interactions using amino acid sequences.

**Key words:** Statistical learning theory, probability theory on a metric space of strings, string classification, bioinformatics.

## 1 Introduction

Numbers and numerical vectors account for a large portion of data. However, in recent years, large amounts of text data have been produced on the Web. In the life sciences, large amounts of data regarding genes, RNAs, and proteins have been generated. These data are nucleotide or amino acid sequences and can be represented as strings. Classifying string data is a common problem in many fields, including computer science and life science. The most widely used approach to this problem is to convert strings into numerical vectors using a string kernel and subsequently apply a support vector machine (SVM) (see, for example, [1, 4, 7, 9, 36]) to the vectors. The earliest string kernels were developed by [12, 39, 27]. These papers proposed that the similarity between strings should be defined based on the number of subsequences common to them. [23, 31] used the spectrum kernel, a string kernel that quantifies the similarity between strings based on the number of common substrings, without considering common subsequences for which gaps are allowed. The spectrum kernel was subsequently extended by [21, 22, 38]. In addition to these kernels, a number of novel string kernels were developed and applied to problems in bioinformatics by [41, 37, 34, 26]. The spectrum kernel has become the most widely used of these various string kernels, although this kernel discards considerable amounts of the information concerning the order of the letters that compose the strings. Converting strings into numerical vectors involves information loss. Why are strings converted into numerical vectors, and why is a classifier that works in a numerical vector space applied to the vectors to classify the strings? To classify strings, a classifier that works in a set of strings would be reasonable. However, the ability of a classifier to receive strings themselves as inputs has not been examined because inputs to an SVM are generally numerical vectors. A more serious problem in the conventional approach is that this approach ignores the fact that the data used to train and test the learning machines are strings generated according to probability laws. Consequently, the perfor-
mance of a learning machine is evaluated based on whether the machine yields better results compared to other machines in a certain simulation experiment or in the application to a certain real data set, and the fundamental evaluation of a learning machine by theoretically evaluating its generalization error has been abandoned.

In this study, by constructing an analogy of an SVM that works in a set of strings, we develop a method for classifying strings without converting them into numerical vectors. To theoretically evaluate the generalization error of a classifier that works in a set of strings, probability theory for strings is required. Mathematicians have conducted detailed examinations of a large number of objects, such as numbers, operators, manifolds, equations, and functions, throughout the long history of mathematics, but they have not studied strings. A string is an object that computer scientists have addressed in depth. Stringology, a field of computer science, has thoroughly investigated algorithms and data structures for string processing (see, for example, [11] and [8]). However, computer scientists have not studied strings using a mathematical approach; for example, operations, functions, and probabilities on a set of strings have not been investigated. [18] first developed a probability theory on a set of strings provided with the Levenshtein distance [24]. An analogy of the strong law of large numbers for a sequence of random strings and a result on the asymptotic behavior for the variance for random strings were demonstrated in [18]; this study provided fundamental tools for developing a statistical theory for string data. [18] developed statistical methods based on this probability theory and applied them to the analysis of biological sequences. In this study, by applying this probability theory on a set of strings, we provide a theoretical evaluation of the generalization error of our developed learning machine. We also demonstrate the usefulness of our machine for practical data analysis by applying it to predicting protein-protein interactions based on amino acid sequences.

2 Specification of the problem

In the following, we refer to a classifier that decomposes a space into two disjoint subsets by choosing a hyperplane under the principle of margin maximization as an SVM, although an SVM also has other characteristics, such as (i) learning on the dual of a vector space, (ii) extracting features from input vectors, and (iii) using kernel func-
R and \( \mathbb{R}^p \) represent the set of real numbers and the \( p \)-dimensional real vector space, respectively. We consider a plane \( \mathbb{R}^2 \) for simplicity. A line in \( \mathbb{R}^2 \) is represented as \( \{ z \in \mathbb{R}^2 : z = \alpha x + \beta y, \alpha + \beta = 1, \alpha, \beta \in \mathbb{R} \} \) for \( x, y \in \mathbb{R}^2 \) and \( \{(x, y) \in \mathbb{R}^2 : y = ax + b, x \in \mathbb{R} \} \) for \( a, b \in \mathbb{R} \). The first representation uses the vector space structure of \( \mathbb{R}^2 \) because addition and scalar multiplication are used in the representation, and the second representation uses the field structure of \( \mathbb{R} \) because addition and multiplication are used. We denote a set of strings on an alphabet \( A = \{a_1, \cdots, a_c\} \) by \( A^* \). The intrinsic operation and distance on \( A^* \) are concatenation (hereafter denoted by \( \cdot \)) and the Levenshtein distance (hereafter denoted by \( d_L \)), respectively. Therefore, we provide \( A^* \) with algebraic and topological structures using \( \cdot \) and \( d_L \). \( A^* \) forms a noncommutative topological monoid, but it does not form a vector space or field. Therefore, “a line” cannot be defined in \( A^* \) using the above two forms. However, this does not mean that a line cannot be defined in \( A^* \). Thus, we consider the following two questions: (i) Can “a line” be defined in \( A^* \) in some way? (ii) If so, can \( A^* \) be decomposed into two disjoint subsets by using “the line”? The answer to the first question is “Yes”, whereas the answer to the second question is “No”.

By considering a curve in a space to be a subset of the space that is obtained by repeating the operation of connecting a point in the space to one of its contiguous points, we can roughly define “a curve” in \( A^* \), for example, in the following manner: If \( d_L(s_i, s_{i+1}) = 1, i = 1, \cdots, n - 1 \) holds for \( s_1, \cdots, s_n \in A^* \), we call \( \{s_1, \cdots, s_n\} \) “a curve” in \( A^* \). Furthermore, considering a segment between two points in a space to be the shortest curve that connects the two points, we can define “a segment” in \( A^* \) as follows: We suppose that \( d_L(s, s') = n \) for \( s, s' \in A^* \). \( s \) can be transformed into \( s' \) by performing one of three types of operation, insertion, deletion, and substitution, \( n \) times. We denote a string obtained by performing the first \( i \) operations of the \( n \) operations on \( s \) by \( s(i) \) for each \( i = 1, \cdots, n - 1 \). For the uniqueness of “a segment” that connects two given strings, we suppose that the order of priority is given among insertion, deletion, and substitution and that a series of operations is performed on \( s \) in ascending order with respect to the letter number in \( s \) and according to the order of priority among the three operations. We call \( \{s, s(1), \cdots, s(n-1), s'\} \) “a segment” in \( A^* \) that connects \( s \) and \( s' \).

Therefore, we consider the decomposition of a sufficiently large subset for applications that are composed of strings whose length is less than or equal to that of \( s \), although
not the entire space of $A^*$, by choosing a sufficiently long string $s$ and drawing a segment between $s$ and the empty string (a string composed of zero letters). The alphabet $A = \{a_1, \cdots, a_{c-1}\}$ forms a metric space with the Hamming distance $d_H(a_i, a_j) = 0$ (if $i = j$) or 1 (if $i \neq j$). By comparison, the set of real numbers $\mathbb{R}$ also forms a metric space with the absolute value of the difference $d(x, y) = |x - y|$ as well as a totally ordered set with respect to the usual less-than-or-equal relation $\leq$. The distance $d$ and the total order $\leq$ on $\mathbb{R}$ are consistent in the sense that if $x \leq y$ and $y \leq z$, then $d(x, y) \leq d(x, z)$ holds for any $x, y, z \in \mathbb{R}$. Such an intrinsic total order as the less-than-or-equal relation $\leq$ on $\mathbb{R}$ does not exist on $A$. By defining a total order that is consistent with the Hamming distance $d_H$ in the sense mentioned above, can we make $A$ form a totally ordered set without destroying its structure as a metric space? This task is impossible due to the definition of $d_H$. Consequently, we have the following problem. $\mathbb{R}^2$ can be divided into upper and lower half-spaces $H^+ = \{(x, y) \in \mathbb{R}^2 : ax + b \leq y\} - \ell$ and $H^- = \{(x, y) \in \mathbb{R}^2 : y \leq ax + b\} - \ell$ with a line $\ell = \{(x, y) \in \mathbb{R}^2 : y = ax + b\}$. $H^+$ and $H^-$ are defined using the total order $\leq$ on $\mathbb{R}$. In other words, for the concepts of upper and lower areas of a line in the direct product space $\mathbb{R}^2$ to make sense, the total order on the direct product factor $\mathbb{R}$ is required. The analogies of a curve and segment can be defined in $A^*$ in the above manner using the Levenshtein distance. However, the concepts of upper and lower areas of a segment cannot make sense without destroying the structure of $A$ as a metric space because a total order that is consistent with the Hamming distance cannot be defined on $A$. Consequently, $A^*$ cannot be divided by determining such a non-closed subset as a line, in contrast to $\mathbb{R}^2$.

However, the above discussion does not indicate that $A^*$ cannot be divided into two disjoint subsets in any manner. As the Jordan curve theorem [17] and the Jordan–Brouwer separation theorem [5] of topology state, $\mathbb{R}^2$ and $\mathbb{R}^p$ ($p \geq 3$) can be divided into two disjoint subsets by choosing a closed curve and hypersphere without determining a line or hyperplane, respectively. Can we decompose $A^*$ into two disjoint subsets by using a method other than by drawing a line? We set $U(s, r) = \{t \in A^* : d_L(t, s) \leq r\}$ for $s \in A^*$ and $r \in \mathbb{Z}^+$ ($\mathbb{Z}^+$ represents the set of positive integers) and consider the decomposition of $A^*$ into $U(s, r)$ and $U(s, r)^c = A^* - U(s, r)$. In other words, we examine a method of drawing a sphere in $A^*$ and subsequently decomposing $A^*$ into its interior and exterior. In this manner, the decomposition does not require the concepts of upper
and lower areas. In the following, we refer to $\partial U(s, r) = \{t \in A^* : d_L(t, s) = r\}$ as a discriminant sphere and the number of strings in $U(s, r)$ as the size of $\partial U(s, r)$.

3 Analogy of an SVM working in a set of strings

To decompose $A^*$ in the manner described in the previous section, it is necessary to specify the center $s \in A^*$ and the radius $r \in \mathbb{Z}^+$ of a discriminant sphere $\partial U(s, r)$ given positive and negative examples. We say that the positive examples $X_m = \{s_1, \cdots, s_m\}$ and negative examples $Y_n = \{t_1, \cdots, t_n\}$ are spherically separable if there exists $s_0 \in A^*$ such that

$$\max_{1 \leq i \leq m} \{d_L(s_i, s_0)\} < \min_{1 \leq i \leq n} \{d_L(t_i, s_0)\}$$

holds and that $X_m$ and $Y_n$ are spherically inseparable if they are not spherically separable. We denote a set of $m$-tuples of strings for which a consensus sequence is uniquely determined by $[(A^*)^m]$. A formal definition of a consensus sequence is provided in Appendix. We suppose $s_1, \cdots, s_m \in [(A^*)^m]$ in the following and choose the consensus sequence $\bar{s}_m$ of positive examples $s_1, \cdots, s_m$ as the center of a discriminant sphere.

We first consider the problem of choosing the radius of a discriminant sphere for the case in which the positive and negative examples are spherically separable. Similarly to a discriminant hyperplane of an SVM in $\mathbb{R}^p$, the distance between a string and a discriminant sphere is the distance between the string and a string in the sphere that is nearest to the string, and given samples of positive examples and negative examples, the margin of a discriminant sphere is the distance between the sphere and an example in the samples that is nearest to the sphere. Under the principle of margin maximization, the following result can be immediately obtained: If the positive examples $X_m = \{s_1, \cdots, s_m\}$ and negative examples $Y_n = \{t_1, \cdots, t_n\}$ are spherically separable with respect to $\bar{s}_m$, the radius of a discriminant sphere that maximizes the margin is given by

$$r^* = \frac{\max_{1 \leq i \leq m} \{d_L(s_i, \bar{s}_m)\} + \min_{1 \leq i \leq n} \{d_L(t_i, \bar{s}_m)\}}{2}. \quad (1)$$

If $r^*$ is not an integer, we arbitrarily choose one of the integers closest to $r^*$.

Next, we consider the case in which the positive examples $X_m$ and negative examples $Y_n$ are spherically inseparable. We denote subsamples of the positive and negative examples that a discriminant sphere $\partial U(\bar{s}_m, r)$ with a center $\bar{s}_m$ and a radius $r$ correctly classifies by $X_m(\bar{s}_m, r)$ and $Y_n(\bar{s}_m, r)$, respectively. We denote the number of elements of
a finite set $S$ by $\sharp S$. The numbers of strings in $X_m$ and in $Y_n$ that $\partial U(\bar{s}_m, r)$ misclassifies are represented by $m - \sharp X_m(\bar{s}_m, r)$ and $n - \sharp Y_n(\bar{s}_m, r)$, respectively. If the two samples $X_m$ and $Y_n$ are spherically inseparable, we choose the radius of a discriminant sphere based on the principle of minimizing the number of misclassified inputs and maximizing the margin, which is a slight modification of the principle used by an ordinary SVM in $\mathbb{R}^p$ in soft margin optimization. If the positive and negative examples are spherically separable, the following procedure is reduced to choosing the radius according to equation (1).

Step 1 (minimizing the number of misclassified inputs). Search for a set of radii that minimize the number of misclassified inputs, i.e., a set of positive integers $\tilde{r}$ that satisfy

$$\tilde{r} = \arg \min_{r \in \mathbb{Z}^+} \{m - \sharp X_m(\bar{s}_m, r) + n - \sharp Y_n(\bar{s}_m, r)\},$$

or equivalently,

$$\tilde{r} = \arg \max_{r \in \mathbb{Z}^+} \{\sharp X_m(\bar{s}_m, r) + \sharp Y_n(\bar{s}_m, r)\}.$$

We denote this set by $\tilde{R}$. $\tilde{R}$ is a nonempty finite set.

Step 2 (maximizing the margin). Choose $r^* \in \tilde{R}$ that maximizes the distance to the closest string that is correctly classified (if such $r^*$ is not uniquely determined, we arbitrarily choose one of them). This step is formally written as follows: The distances between $s \in X_m(\bar{s}_m, r)$ and $\partial U(\bar{s}_m, r)$ and between $t \in Y_n(\bar{s}_m, r)$ and $\partial U(\bar{s}_m, r)$ are equal to $r - d_L(s, \bar{s}_m)$ and $d_L(t, \bar{s}_m) - r$, respectively. These distances are not necessarily equal when $s$ and $t$ are support strings, in contrast to support vectors for an ordinary SVM in $\mathbb{R}^p$, because their sum $r - d_L(s, \bar{s}_m) + d_L(t, \bar{s}_m) - r = d_L(t, \bar{s}_m) + d_L(s, \bar{s}_m)$ may be odd. The optimal radius $r^*$ is represented as $r^* = \arg \max_{\tilde{r} \in \tilde{R}} \rho(\tilde{r})$ for

$$\rho(\tilde{r}) = \min_{(s, t) \in X_m(\bar{s}_m, \tilde{r}) \times Y_n(\bar{s}_m, \tilde{r})} \min\{\tilde{r} - d_L(s, \bar{s}_m), d_L(t, \bar{s}_m) - \tilde{r}\}, \tilde{r} \in \tilde{R},$$

and the support strings are given by

$$(s^*, t^*) = \min_{(s, t) \in X_m(\bar{s}_m, r^*) \times Y_n(\bar{s}_m, r^*)} \min\{r^* - d_L(s, \bar{s}_m), d_L(t, \bar{s}_m) - r^*\}.$$  

To search for the set $\tilde{R}$, it is sufficient to examine only those radii between $\max_{1 \leq i \leq m} \{d_L(s_i, \bar{s}_m)\}$ and $\min_{1 \leq i \leq n} \{t_i, d_L(\bar{s}_m)\} - 1$. 

7
4 Asymptotic optimality of the proposed learning machine

We consider whether the analogy of an SVM in $A^*$ constructed in the previous section has an optimality property in terms of the generalization error. We address this problem by applying the fundamental framework of probability theory on $A^*$ proposed by [18] and a result demonstrated in this framework on the asymptotic behavior of the consensus sequence. Definitions and the theorem used in this section are cited in Appendix. Let $p_1$ and $p_2$ be probability functions of the distributions that generate the positive and negative examples, respectively. We denote the consensus sequence of $p_1$ by $m'_1$ ($p_1$ and $p_2$ are introduced as $p_\sigma$ and $m'_1$ is introduced as $m'_1(\sigma)$ in Appendix). $D_1$ and $D_2$ represent the supports of $p_1$ and $p_2$, respectively, i.e., $D_i = \{s \in A^*: p_i(s) > 0\}$ for $i = 1, 2$. We assume that $D_1 - D_2 \neq \emptyset$ and $D_2 - D_1 \neq \emptyset$. If $D_1 \cap D_2 = \emptyset$, the probability that the generalization error becomes zero after finite times of learning is equal to one. Therefore, we consider the case of $D_1 \cap D_2 \neq \emptyset$ in the following. The generalization error $E_0(s, r)$ of a discriminant sphere $\partial U(s, r)$ is written as $E_0(s, r) = E_1(s, r) + E_2(s, r)$ for

$$E_1(s, r) = \sum_{t \in D_1 \cap U(s, r)} p_1(t), \quad E_2(s, r) = \sum_{t \in D_2 \cap U(s, r)} p_2(t).$$ (2)

We formally set $(s^\dagger, r^\dagger) = \arg\min_{(s, r) \in A^* \times \mathbb{Z}^+} E_0(s, r)$ and $r^\dagger(s_0) = \arg\min_{r \in \mathbb{Z}^+} E_0(s_0, r)$ for each $s_0 \in A^*$. $r^\dagger(s_0)$ is the radius of a discriminant sphere that is optimal in terms of the generalization error given a center. We denote the relative frequencies of $t$ in $X_m$ and in $Y_n$ by $\hat{p}_1(t)$ and $\hat{p}_2(t)$, respectively, for any $t \in A^*$. We set

$$\hat{E}_1(s, r) = \sum_{t \in X_m \cap U(s, r)} \hat{p}_1(t), \quad \hat{E}_2(s, r) = \sum_{t \in Y_n \cap U(s, r)} \hat{p}_2(t)$$ (3)

for $s \in A^*$ and $r \in \mathbb{Z}^+$. Assuming that $\bar{s}_m$ is used as the center of a discriminant sphere, we first consider whether $r^*$ converges to an optimal radius in terms of the generalization error as our learning machine updates $r^*$ through a learning process. Note that under the conditions of Theorem [8] described in Appendix, the optimal radius is $r^\dagger(m'_1)$ because $\bar{s}_m$ is equal to $m'_1$ with probability one, given a sufficient number of positive examples.
Theorem 1 (Asymptotic optimality of $r^*$) If (i) the positive examples $s_1, \cdots, s_m$ are realizations of random strings $\sigma_1, \cdots, \sigma_m$ that are independent and have the identical probability function $p_1$ and the negative examples $t_1, \cdots, t_n$ are realizations of random strings $\tau_1, \cdots, \tau_n$ that are independent and have the identical probability function $p_2$, (ii) the conditions of Theorem 3 in Appendix are satisfied, and (iii) there uniquely exists $r^*(m')$, we have

$$r^* \xrightarrow{a.s.} r^*(m') \ (m, n \to \infty),$$

where $\xrightarrow{a.s.}$ represents almost sure convergence. In other words, $r^*$ converges to a radius that is asymptotically optimal given $\bar{s}_m$ as the center of a discriminant sphere with probability one.

Proof. (Step 1) Setting $W_k = \{ s \in A^* : |s| = k \}$ for any $k \in \mathbb{N}$, we have $\sharp W_k = (c-1)^k$. Therefore, the number of strings whose length is less than or equal to $\ell$ can be represented as $\sum_{k=0}^{\ell} (c-1)^k$. Thus, noting that $\sum_{k=0}^{\ell} (c-1)^k < \infty$ holds for $\ell < \infty$ and the definition of a string described in Appendix, we have $\sharp A^* < \infty$. Therefore, there exists $h \in \mathbb{Z}^+$ such that we can write $A^* = \{ u_1, \cdots, u_h \}$. We first consider the probability function $p_1$ of the population distribution of the positive examples. We define an $h$-dimensional random vector $X_i = (X_{i1}, \cdots, X_{ih})$ by setting

$$X_{ij} = 1 \text{ and } X_{ij'} = 0 \text{ for } j' \in \{1, \cdots, h\} - \{j\}$$

if $u_j$ is generated in the $i$-th learning from this population for each $i \in \{1, \cdots, m\}$. $X_i$ has a multinomial distribution with the number of trials one and the success probabilities $p_1(u_1), \cdots, p_1(u_h)$, and thus the expectation vector of $X_i$ is given by $(p_1(u_1), \cdots, p_1(u_h))$. From assumption (i) of the theorem, $X_1, \cdots, X_m$ are independent. Therefore, by applying the strong law of large numbers in $\mathbb{R}^h$, we have

$$\frac{1}{m} \sum_{i=1}^{m} X_i \xrightarrow{a.s.} (p_1(u_1), \cdots, p_1(u_h)) \ (m \to \infty).$$

Thus, noting $(1/m) \sum_{i=1}^{m} X_i = (\hat{p}_1(u_1), \cdots, \hat{p}_1(u_h))$ gives

$$\hat{p}_1(u_j) \xrightarrow{a.s.} p_1(u_j) \ (m \to \infty) \quad (4)$$
for each \( j \in \{1, \cdots, h\} \). For the probability function \( p_2 \) of the population distribution of
the negative examples, we obtain

\[
p_2(u_j) \xrightarrow{a.s.} p_2(u_j) \ (n \to \infty)
\]

for each \( j \in \{1, \cdots, h\} \) in the same manner.

(Step 2) We have \( \sharp D_1 < \infty \) for the support \( D_1 \) of \( p_1 \) because \( D_1 \subset A^* \), and therefore we write \( D_1 = \{u'_1, \cdots, u'_h\} \). Combining equation (4) and \( p(u'_1), \cdots, p(u'_h) > 0 \), we see that for any \( j \in \{1, \cdots, h'\} \), there exists \( m_j \in \mathbb{Z}^+ \) such that if \( m \geq m_j \),

\[
X_{ij} \geq 1 \text{ a.s.}
\]

holds for at least one \( i \in \{1, \cdots, m\} \), where a.s. indicates that a statement in front of it holds with probability one. Thus, setting \( m^* = \max\{m_1, \cdots, m_{h'}\} \), we have \( S_m = D_1 \) a.s. for any \( m \geq m^* \). By using equation (5), we find that choosing a sufficiently large \( n^* \in \mathbb{Z}^+ \) gives us \( T_n = D_2 \) a.s. for any \( n \geq n^* \) in the same manner.

(Step 3) Noting equations (2) and (3) and using the results obtained in Steps 2 and 3, we obtain \( \hat{E}_1(\bar{s}_m, r) \xrightarrow{a.s.} E_1(\bar{s}_m, r) \) and \( \hat{E}_2(\bar{s}_m, r) \xrightarrow{a.s.} E_2(\bar{s}_m, r) \) as \( m, n \to \infty \) for any \( r \in \mathbb{Z}^+ \). Consequently, it follows that \( \hat{E}_0(\bar{s}_m, r) \xrightarrow{a.s.} E_0(\bar{s}_m, r) \) as \( m, n \to \infty \) by the Mann–Wald theorem [29]. In our procedure for choosing a radius described in the previous section, a positive integer \( r^* \) that minimizes \( m - \frac{1}{m}X_m(\bar{s}_m, \cdot) + n - \frac{1}{n}Y_n(\bar{s}_m, \cdot) \)
or, equivalently, \( \hat{E}_0(\bar{s}_m, \cdot) \) is chosen at each learning step. From assumption (ii) of the theorem, we have \( \bar{s}_m = m'_1 \) a.s. as \( m \to \infty \) according to Theorem [3]. Therefore, noting assumption (iii) of the theorem, we see that \( r^* \) almost surely converges to \( r^\dagger(m'_1) \) that is the limit of the minimizers \( \{r^\dagger(\bar{s}_m)\} \) of \( \{E_0(\bar{s}_m, \cdot)\} \).

In the above proof of the asymptotic optimality of \( r^* \), only the principle of minimizing the number of misclassified inputs was used, and the principle of maximizing the margin was not required because the samples of the positive and negative examples accurately reflected their population distributions in the asymptotic setting. This suggests that the reason an ordinary SVM in \( \mathbb{R}^p \) has a high predictive performance in a number of applications is that margin maximization plays a role in reducing the probability of misclassifying examples in a test sample when the positive and negative examples in a training sample do not necessarily accurately reflect their population distributions be-
cause, for example, a training sample is not sufficiently large; in other words, margin maximization is a reasonable principle for classifying data in a sample in cases where the sample does not include sufficient information on the population distribution.

We next consider the optimality of $\bar{s}_m$. We address this problem in the following setting, which models the situation in which the positive and negative examples are spherically inseparable: $D_1 \cap D_2 \neq \emptyset$ holds and $p_1$ and $p_2$ satisfy the conditions that (i) $p_1(s)$ is monotonically non-increasing with respect to $d_L(s, m'_1)$ on $D_1$ and (ii) there exists $d_0 \in \mathbb{Z}^+$ such that $p_2(s)$ is monotonically non-decreasing with respect to $d_L(s, m'_1)$ on $D'_2 = \{s \in D_2 : d_L(s, m'_1) \leq d_0\}$ and $D_1 \not\supset D'_2$. We assume that $d_0$ is sufficiently large and consider only discriminant spheres that are disjoint with $(D_1 \cup D'_2)^c$. We do not assume that $r^*$ is chosen as the radius of a discriminant sphere. $\sharp U(s, r)$ increases monotonically with respect to the length of $s$ and $r$. We denote sets of pairs $(s', r') \in A^* \times \mathbb{Z}^+$ such that $\sharp U(s', r') = \sharp U(\bar{s}_m, r)$, $\sharp U(s', r') \leq \sharp U(\bar{s}_m, r)$, and $\sharp U(s', r') \geq \sharp U(\bar{s}_m, r)$ by $B_0(r), B_1(r)$, and $B_2(r)$, respectively, for any $r \in \mathbb{Z}^+$.

**Theorem 2 (Asymptotic optimality of $\bar{s}_m$)** In the setting described above, if the conditions of Theorem 3 in Appendix are satisfied, we have

$$E_j(\bar{s}_m, r) \leq E_j(s', r') \text{ a.s.}$$

as $m, n \longrightarrow \infty$ for any $r \in \mathbb{Z}^+$, $(s', r') \in B_j(r)$, and $j \in \{0, 1, 2\}$. In other words, for any radius $r$, a discriminant sphere with a center $\bar{s}_m$ is asymptotically optimal in a class of discriminant spheres that are equal to it in size and has asymptotically minimum probabilities of false negatives of discriminant spheres of equal or smaller size and of false positives of discriminant spheres of equal or larger size.

**Proof.** We first consider the case of $j = 0$. It is sufficient to show that $E_0(m'_1, r) \leq E_0(s', r')$ holds for any $r \in \mathbb{Z}^+$ and $(s', r') \in B_0(r)$ by Theorem 3.

(Step 1) We consider the case of $U(m'_1, r) \subset D_1$. We put

$$F = (D_1 - U(s', r')) \cap U(m'_1, r), \ G = (D_1 - U(m'_1, r)) \cap U(s', r'), \ H = U(s', r') - D_1$$

for $(s', r') \in B_0(r)$. $F$ is the set of strings that $\partial U(m'_1, r)$ correctly discriminates but that $\partial U(s', r')$ incorrectly discriminates, whereas $G$ is the set of strings that $\partial U(s', r')$
correctly discriminates but that $\partial U(m'_1, r)$ incorrectly discriminates. $F$ and $G \cup H$ can be rewritten as

$$F = U(m'_1, r) - (U(m'_1, r) \cap U(s', r')),$$

$$G \cup H = U(s', r') - (U(m'_1, r) \cap U(s', r')),$$

respectively. Hence, noting $G \cap H = \emptyset$, we have

$$\sharp F = \sharp \{U(m'_1, r) - (U(m'_1, r) \cap U(s', r'))\} = \sharp U(m'_1, r) - \sharp (U(m'_1, r) \cap U(s', r')),$$

$$\sharp G + \sharp H = \sharp (G \cup H) = \sharp \{U(s', r') - (U(m'_1, r) \cap U(s', r'))\}$$

$$= \sharp U(s', r') - \sharp (U(m'_1, r) \cap U(s', r')).$$

Combining these equations with $\sharp U(s', r') = \sharp U(m'_1, r)$ gives $\sharp F = \sharp G + \sharp H$. Consequently,

$$\sharp F \geq \sharp G$$

holds. We have $d_L(s, m'_1) < d_L(t, m'_1)$ for any $s \in F$ and $t \in G$. Thus, we obtain

$$p_1(s) \geq p_1(t)$$

by assumption (i) of the theorem. From equations (6) and (7), we have

$$\sum_{s \in F} p_1(s) \geq \sum_{t \in G} p_1(t),$$

i.e., the probability of false negatives of $\partial U(m'_1, r)$ is less than or equal to that of $\partial U(s', r')$. Using assumption (ii) of the theorem, we can show that the probability of false positives of $\partial U(m'_1, r)$ does not exceed that of $\partial U(s', r')$ in a similar manner.

In the case of $U(m'_1, r) \supset D_1$, the probability of false negatives of $\partial U(m'_1, r)$ is equal to zero and that of $\partial U(s', r')$ is nonnegative. Therefore, it is sufficient to show that the probability of false positives of $\partial U(m'_1, r)$ does not exceed that of $\partial U(s', r')$, which can be demonstrated using assumption (ii) in a similar manner.

(Step 2) We consider the case of $U(m'_1, r) \cap D'_2 \neq \emptyset$. We put

$$I = (D'_2 - U(m'_1, r)) \cap U(s', r'),$$

$$J = (D'_2 - U(s', r')) \cap U(m'_1, r),$$

$$K = (D_1 - D'_2) - U(s', r')$$
for \((s', r') \in B_0(r)\). \(I\) is the set of strings that \(\partial U(s', r')\) incorrectly discriminates, whereas \(J\) is the set of strings that \(\partial U(m'_1, r)\) incorrectly discriminates. We can obtain \(\sharp I \geq \sharp J\) in a manner similar to that in Step 1. From assumption (ii), \(p_2(s) \geq p_2(t)\) holds for \(s \in I\) and \(t \in J\). Thus, we have

\[
\sum_{s \in I} p_2(s) \geq \sum_{t \in J} p_2(t),
\]

i.e., the probability of false positives of \(\partial U(m'_1, r)\) is less than or equal to that of \(\partial U(s', r')\).

When \(U(m'_1, r) \cap D'_2 \neq \emptyset\) holds, there are the two cases: \(U(m'_1, r) \supset D_1\) or \(U(m'_1, r) \subset D_1\). In the case of \(U(m'_1, r) \supset D_1\), the probability of false negatives of \(\partial U(m'_1, r)\) is equal to zero and that of \(\partial U(s', r')\) is nonnegative. In the case of \(U(m'_1, r) \subset D_1\), noting assumption (i), we can show that the probability of false negatives of \(\partial U(m'_1, r)\) does not exceed that of \(\partial U(s', r')\) in a similar manner.

(Step 3) Combining the results of Steps 1 and 2 leads to the desired conclusion for \(j = 0\). In the case of \(j = 1\), we can obtain the desired result by replacing \(B_0(r), \sharp U(s', r') = \sharp U(m'_1, r),\) and \(\sharp F = \sharp G + \sharp H\) with \(B_1(r), \sharp U(s', r') \leq \sharp U(m'_1, r),\) and \(\sharp F \geq \sharp G + \sharp H\), respectively, in the evaluation of the probability of false negatives of \(\partial U(m'_1, r)\) in Step 1. In the case of \(j = 2\), the proof is completed by replacing \(B_0(r)\) with \(B_2(r)\) in the evaluation of false positive probability in Step 2.

\[\square\]

5 Application to predicting protein-protein interactions

Our learning machine classifies strings in an almost optimal manner under the conditions described in the previous section when the training samples are sufficiently large. However, large training samples are not necessarily obtainable in all problems of classifying strings. How accurately does our machine classify strings in such cases? A protein is a polymer of 20 types of amino acids and can be represented as a string on an alphabet composed of 20 letters. Predicting protein-protein interactions is one of the most impor-
tant problems in bioinformatics because most proteins fulfill their functions after forming a complex with other proteins. A domain of a protein generally interacts with multiple domains of other proteins. However, only a few proteins have domains that interact with a number of domains of other proteins and function as a hub in a protein-protein interaction network [15, 14]. Thus, large numbers of positive examples cannot necessarily be obtained in the problem of predicting protein-protein interactions. We formulated this prediction problem as the problem of classifying domains of proteins into two classes of domains that interact and do not interact with a given domain and applied the analogy of an SVM in $A^*$ to this problem. We examined the classification accuracy of our machine through a comparison with the SVM with the 2-spectrum kernel.

We first prepared positive examples by using the three-dimensional interacting domains (3did) database [30], which contains high-resolution, three-dimensional structural data for domain-domain interactions obtained from the Protein Data Bank (PDB) [33]. The PDB includes three-dimensional structures of proteins and protein complexes obtained from experiments such as X-ray crystal structural analysis and nuclear magnetic resonance spectroscopy. We selected 10 protein domain sequences, '1il1 A:134-215' (which denotes the amino acid subsequence from residue 134 to 215 in chain A of PDB id 1il1), '2bnq E:127-220', '1inq B:1011-1092', '1it9 H:133-216', '1it9 L:123-210', '1ikv A:317-419', '1cff A:84-145', '1iza A:1-20', '1ifh L:119-206', and '1p2c F:1501-1627'. For each of the amino acid sequences of these 10 domains, we collected sequences of interacting domains from 3did without any redundancy in sequences. Sequences having extremely different lengths relative to the sequences collected from the database were not included in the samples of positive examples. It should be noted that the same amino acid sequence can be included in different entries of the PDB.

We next consider the procedure for preparing negative examples. Real sequences that would not be positive examples and artificial and randomly generated sequences have been used as negative examples in the development and validation of classifiers for string and sequence data. For example, in promoter prediction in *E. coli*, [13] used sequences randomly chosen from coding regions as the negative examples. However, as [20] indicated, the use of such negative examples is very different from a real biological discrimination problem. For the problem of predicting the interaction between miRNA and mRNA, [10, 16, 40] used randomly generated artificial sequences as the negative
examples. However, as [25, 32, 19] demonstrated experimentally, such sequences often interact with miRNA, and therefore it is uncertain if they are negative examples. Even if the randomly generated sequences are real negative examples, they may be unrealistically different from positive examples. In this case, as [2] indicated, the positive and negative examples are easily distinguishable, and a classifier that yields poor performance on other independent data sets may be produced.

Thus, in this study, we considered a procedure for preparing negative examples that were somewhat similar to the positive examples but were not likely to interact based on biophysical chemistry data. [3] compiled a database of 2,325 alanine mutants of heterodimeric protein-protein complexes and examined which amino acids are located in the interfaces of protein-protein complexes with a relatively high frequency. [28] identified amino acids that are located in the interfaces with a high frequency on the basis of 1,629 two-chain interface entries in the PDB. Based on the results of these studies, Arg, Asp, Trp, and Tyr are located at the interface of protein-protein interactions with high frequency, whereas Lys and Glu are located at interfaces with low frequency. Arg and Lys have a positive charge, and Arg tends to be located at interfaces; conversely, Lys does not tend to be located at interfaces. By contrast, Asp and Glu are negatively charged; while Asp tends to be located at interfaces, Glu does not. These observations imply that amino acids in which the side chain has relatively low entropy tend to be located at interfaces. π electrons in the side chains of aromatic amino acids interact strongly with positively charged amino acids [6], and for this reason, aromatic amino acids such as Trp and Tyr are located at interfaces with high frequency. Therefore, we generated a negative example from each positive example using the following steps: (i) We first substituted Glu for Arg and Lys for Asp, Trp, and Tyr in a positive example with a probability of 0.5 (because if all the Arg, Asp, Trp, and Tyr were replaced, the resulting negative examples would not contain the four types of letters that represent these amino acids). According to [35], the frequencies of Arg, Asp, Trp, and Tyr in common proteins are 5.1%, 5.3%, 1.4%, and 3.2%, respectively. Hence, in this step, approximately 7.5% of the letters in the positive example that represented amino acids that tend to be located in interfaces were replaced with letters representing amino acids that do not tend to be located in interfaces. (ii) Next, we trisected a positive example with the substitutions in Step (i), chose a letter in the intermediate substring at random, and transposed the order of the
first half substring, from the first letter in the substituted positive example to the chosen letter, and the second half substring, composed of the other letters in the substituted positive example.

The sum of the numbers of positive and negative examples is given as $N$ in Table 5. Our examination included cases with an insufficient sample size $N = 20$ up to a case with a large sample size $N = 126$. We evaluated the performance of the machines using the four indices of accuracy $= \frac{TP + TN}{N}$, precision $= \frac{TP}{TP + FP}$, recall $= \frac{TP}{TP + FN}$, and F-measure $= 2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$, where $TP, FP, TN,$ and $FN$ represent the numbers of true positives, false positives, true negatives, and false negatives, respectively. After randomly dividing each of the samples of positive and negative examples into three subsamples of equal size, we used the union of two subsamples as a training sample and the other subsample as a test sample to compute the above indices. We calculated the mean accuracy, precision, recall, and F-measure by repeating this process 50 times. The mean values obtained in this procedure are shown in Table 5. From this table, we see that the performance of the analogy of an SVM in $A^*$ is high compared to the SVM using the 2-spectrum kernel.

Table 1. Results of the simulation experiments. The first row in each of the two panels presents "the PDB ID of the protein the chain:the initial residue number in the interaction site–the last residue number". $N$ and $\bar{l}$ denote the sample size and the mean length of the positive and negative examples, respectively. SVM S.K. and SVM $A^*$ represent the SVM with the spectrum kernel and the analogy in $A^*$ of an SVM constructed in this study, respectively.

|          | 1hil A:134-215 | 2bng E:127-220 | 1mon B:1011-1092 | 1e99 H:133-216 | 1l09 L:123-210 |
|----------|----------------|----------------|------------------|----------------|----------------|
| N        | 20             | 22             | 82               | 40             | 40             |
| $\bar{l}$| 87.20          | 79.45          | 83.63            | 84.05          | 84.05          |
| SVM S.K. | 0.8267         | 0.9433         | 0.9475           | 1.0000         | 0.8380         |
| SVM $A^*$| 0.9140         | 0.8267         | 0.9817           | 0.9045         | 0.9140         |
| accuracy | 0.8359         | 0.9316         | 0.9446           | 1.0000         | 0.7919         |
| precision| 0.8833         | 0.9500         | 1.0000           | 1.0000         | 0.8452         |
| recall   | 0.7933         | 0.8950         | 0.9520           | 0.6567         | 0.6543         |
| F-measure| 0.8359         | 0.9316         | 0.9446           | 1.0000         | 0.7919         |

|          | 1ikv A:317-419 | 1cff A:84–145 | 1iza A:1–20     | 1ifh L:119–206 | 1p2c F:1501–1627 |
|----------|----------------|---------------|-----------------|----------------|------------------|
| N        | 46             | 20            | 100             | 80             | 126              |
| $\bar{l}$| 171.83         | 20.57         | 25.02           | 82.08          | 111.84           |
| SVM S.K. | 0.8000         | 0.8000        | 0.8000          | 0.8000         | 0.8000           |
| SVM $A^*$| 0.8000         | 0.8000        | 0.8000          | 0.8000         | 0.8000           |
| accuracy | 0.8359         | 0.8359        | 0.8359          | 0.8359         | 0.8359           |
| precision| 1.0000         | 1.0000        | 1.0000          | 1.0000         | 1.0000           |
| recall   | 0.7311         | 0.6543        | 0.9114          | 0.9610         | 0.9546           |
| F-measure| 0.8359         | 0.8359        | 0.8359          | 0.8359         | 0.8359           |
6 Concluding remarks

In recent years, the amount of string data generated has increased dramatically. Consequently, a random string that randomly generates strings based on a probability law has become necessary for string data analysis, as a random variable that randomly generates numbers and a stochastic process that randomly generates functions are essential in various fields. Statistical methods for numerical data were rigorously constructed based on probability theory. Similarly, the development and systematization of methods on the basis of probability theory on a set of strings will be required for text mining techniques and methods for analyzing biological sequences in bioinformatics. Developing a method based on both simulation experiments and theoretical analyses and systematizing various methods in a theoretical framework represent important challenges in future methodological research on the analysis of string data.

Appendix

In this appendix, we describe the definitions of several concepts and the theorem in probability theory on a set of strings used in the main text. See the online supplemental material of [18] for details. In the following, we refer to a set of a finite number of letters

\[ A = \{a_1, \cdots, a_{c-1}\} \]

as the alphabet. For example, \( A = \{a, c, g, t\} \) is an alphabet for gene sequences. We denote the empty letter by \( e \) and set \( \bar{A} = A \cup \{e\} \). We denote a set of \( (x_1, \cdots, x_n) \in \bar{A}^n \) of which a letter with the maximum frequency is uniquely determined by \( [\bar{A}^n] \). A mapping \( m : [\bar{A}^n] \to \bar{A} \) is defined as

\[ m(x_1, \cdots, x_n) = \text{a letter with the maximum frequency of } x_1, \cdots, x_n \]

and is called a consensus letter on \( [\bar{A}^n] \).

Let \( (\Omega, \mathcal{F}, P) \) be a probability space. We denote the power set of a set \( X \) by \( 2^X \). We call an \( \bar{A} \)-valued random variable on \( \Omega \) a random letter and denote the set of all random letters by \( \mathcal{M}(\Omega, \bar{A}) \). For the mapping \( \epsilon : \Omega \to \bar{A} \), which is defined as \( \epsilon(\omega) = e \)
for all $\omega \in \Omega$, we have $\epsilon \in \mathcal{M}(\Omega, \bar{A})$. The independence of \{\alpha_n : n \in \mathbb{Z}^+\} $\subset \mathcal{M}(\Omega, \bar{A})$ is defined in the same manner as that of usual random variables. We denote a set of $\alpha \in \mathcal{M}(\Omega, \bar{A})$ for which there exists $x \in \bar{A}$ such that for any $y \in \bar{A} - \{x\}$, $q(x) > q(y)$ holds by $[\mathcal{M}(\Omega, \bar{A})]$, where $q$ is a probability function of a distribution of $\alpha$. A mapping $m' : [\mathcal{M}(\Omega, \bar{A})] \rightarrow \bar{A}$ is defined as

$$m'(\alpha) = x \in \bar{A} \text{ such that } [q(x) > q(y), \forall y \in \bar{A} - \{x\}]$$

and is called a consensus letter on $[\mathcal{M}(\Omega, \bar{A})]$. We denote a set of $(\alpha_1, \cdots, \alpha_n) \in \mathcal{M}(\Omega, \bar{A})^n$ for which a consensus letter of $\alpha_1(\omega), \cdots, \alpha_n(\omega)$ is uniquely determined for any $\omega \in \Omega$ by $[\mathcal{M}(\Omega, \bar{A})]^n$. A mapping $\mu : [\mathcal{M}(\Omega, \bar{A})]^n \rightarrow \mathcal{M}(\Omega, \bar{A})$ is defined as

$$\mu(\alpha_1, \cdots, \alpha_n)(\omega) = m(\alpha_1(\omega), \cdots, \alpha_n(\omega))$$

and called a consensus letter on $[\mathcal{M}(\Omega, \bar{A})]^n]$.  

In common usage in computer science, a string on the alphabet $A = \{a_1, \cdots, a_{c-1}\}$ is a finite sequence of elements of $A$. However, in this study, we define a string as follows, although both definitions are essentially identical: A sequence $s = \{x_i \in \bar{A} : i \in \mathbb{Z}^+\}$ of elements of $\bar{A}$ is a string on $A$ if it satisfies the following conditions:

(i) there exists $k \in \mathbb{Z}^+$ such that $x_k = e$, and (ii) $x_\ell = e$ implies $x_{\ell+1} = e$.

In other words, we define a string on $A$ as a finite sequence of elements of $A$ to which the infinite sequence $(e, \cdots)$ of the empty letter is appended. In the following, by naturally extending the above definition of a string, we define a random string in a manner in which it can realize strings of varying lengths. We denote the set of all strings on $A$ by $A^\ast$. A function $|\cdot| : A^\ast \rightarrow \mathbb{N}$ ($\mathbb{N}$ represents the set of natural numbers including zero) is defined as

$$|s| = \min\{n \in \mathbb{Z}^+ : x_n = e\} - 1, \ s = \{x_n : n \in \mathbb{Z}^+\}$$

and called the length on $A^\ast$. Letting $(s_1, \cdots, s_n) \in (A^\ast)^n$ and $s_i = \{x_{ij} : j \in \mathbb{Z}^+\}$ for each $i = 1, \cdots, n$, we denote a set of $(s_1, \cdots, s_n)$ for which a consensus letter of $x_{1j}, \cdots, x_{nj}$ is uniquely determined for any $j \in \mathbb{Z}^+$ by $[(A^\ast)^n]$. A mapping $m : [(A^\ast)^n] \rightarrow A^\ast$ is defined
\( m(s_1, \cdots, s_n) = \{m(x_{1j}, \cdots, x_{nj}) : j \in \mathbb{Z}^+ \}, \ s_i = \{x_{ij} : j \in \mathbb{Z}^+ \}, \ i = 1, \cdots, n \)

and is called a consensus sequence on \( [(A^*)^n] \). A consensus sequence of \( s_1, \cdots, s_n \) is denoted as \( m(s_1, \cdots, s_n) \) here to explicitly show that it is a function of \( s_1, \cdots, s_n \), but it is abbreviated as \( \bar{s}_n \) in the main text.

We next introduce a random string. A sequence of random letters \( \sigma = \{\alpha_i \in \mathcal{M}(\Omega, \bar{A}) : i \in \mathbb{Z}^+ \} \) is a random string if it satisfies the following conditions:

(i) for any \( \omega \in \Omega \) there exists \( k \in \mathbb{Z}^+ \) such that \( \alpha_k(\omega) = e \), and

(ii) \( \alpha_\ell(\omega) = e \) for \( \omega \in \Omega \) implies \( \alpha_{\ell+1}(\omega) = e \).

We denote the set of all random strings by \( \mathcal{M}(\Omega, A^*) \). A function \( | \cdot | : \mathcal{M}(\Omega, A^*) \rightarrow \mathbb{N} \) is defined as

\[ |\sigma| = \min\{j \in \mathbb{Z}^+ : \alpha_j = e \} - 1, \ \sigma = \{\alpha_i : i \in \mathbb{Z}^+ \} \]

and is called the length on \( \mathcal{M}(\Omega, A^*) \). A random string defined above can be regarded as a special case of a discrete stochastic process. Therefore, a distribution of a random string can be defined as follows: Let \( \sigma = \{\alpha_i : i \in \mathbb{Z}^+ \} \in \mathcal{M}(\Omega, A^*) \). A set function \( Q_{\sigma;i_1,\cdots,i_k} : 2^{\bar{A}^k} \rightarrow [0,1] \) is defined as

\[ Q_{\sigma;i_1,\cdots,i_k}(B) = P(\{\omega \in \Omega : (\alpha_{i_1}(\omega), \cdots, \alpha_{i_k}(\omega)) \in B \}) \]

for each \( k \in \mathbb{Z}^+ \) and \( i_1, \cdots, i_k \in \mathbb{Z}^+ \) that satisfies \( i_1 < \cdots < i_k \). \( Q_{\sigma;i_1,\cdots,i_k} \) is a probability measure on \( 2^{\bar{A}^k} \) and is called a finite-dimensional distribution of \( \sigma \) at sites \( i_1, \cdots, i_k \). A function \( q_{\sigma;i_1,\cdots,i_k} : \bar{A}^k \rightarrow [0,1] \) is defined as

\[ q_{\sigma;i_1,\cdots,i_k}(x_1, \cdots, x_k) = Q_{\sigma;i_1,\cdots,i_k}(\{(x_1, \cdots, x_k)\}) \]

and is called a probability function of \( Q_{\sigma;i_1,\cdots,i_k} \). For the probability function \( q_{\sigma;|\sigma|} \) of the finite-dimensional distribution at sites \( 1, \cdots, |\sigma| \) of \( \sigma \in \mathcal{M}(\Omega, A^*) \), we define the
function \( q_\sigma : A^* \to [0, 1] \) as

\[
q_\sigma(s) = \begin{cases} 
q_{\sigma_1, \ldots, |\sigma|}(x_1, \ldots, x_{|\sigma|}) & \text{(for } x_1, \ldots, x_{|\sigma|} \in \bar{A} \text{ such that)} \\
0 & \text{(if } |\sigma| < |s|) 
\end{cases}
\]

\( q_\sigma \) is a probability function on \( A^* \). The independence of the random strings is defined in the following manner. (1) For the finite case, \( \sigma_1 = \{\alpha_j : j \in \mathbb{Z}^+\}, \ldots, \sigma_n = \{\alpha_{nj} : j \in \mathbb{Z}^+\} \in \mathcal{M}(\Omega, A^*) \) are independent if \((\alpha_{ij} : j \in I_1), \ldots, (\alpha_{nj} : j \in I_n)\) are independent for any nonempty finite set \( I_1, \ldots, I_n \subset \mathbb{Z}^+ \). (2) For the countably infinite case, \( \{\sigma_i : i \in \mathbb{Z}^+\} \in \mathcal{M}(\Omega, A^*) \) are independent if \( \sigma_{i_1}, \ldots, \sigma_{i_k} \) are independent for any \( k \in \mathbb{Z}^+ \) and \( i_1, \ldots, i_k \in \mathbb{Z}^+ \).

We denote a set of \( \{\alpha_j : j \in \mathbb{Z}^+\} \in \mathcal{M}(\Omega, A^*) \) for which a consensus letter of \( \alpha_j \) is uniquely determined for any \( j \in \mathbb{Z}^+ \) by \([\mathcal{M}(\Omega, A^*)]\). A mapping \( m' : [\mathcal{M}(\Omega, A^*)] \to A^* \) is defined as

\[
m'(\sigma) = \{m'(\alpha_j) : j \in \mathbb{Z}^+\}, \quad \sigma = \{\alpha_j : j \in \mathbb{Z}^+\}
\]

and is called a consensus sequence on \([\mathcal{M}(\Omega, A^*)]\). Letting \((\sigma_1, \ldots, \sigma_n) \in \mathcal{M}(\Omega, A^*)^n \) and \( \sigma_i = \{\alpha_{ij} : j \in \mathbb{Z}^+\} \) for each \( i = 1, \ldots, n \), we denote a set of \((\sigma_1, \ldots, \sigma_n)\) for which a consensus letter of \( \alpha_{ij}(\omega), \ldots, \alpha_{nj}(\omega) \) is uniquely determined for any \( j \in \mathbb{Z}^+ \) and \( \omega \in \Omega \) by \([\mathcal{M}(\Omega, A^*)]^n\). A mapping \( \mu : [\mathcal{M}(\Omega, A^*)]^n \to \mathcal{M}(\Omega, A^*) \) is defined as

\[
\mu(\sigma_1, \ldots, \sigma_n)(\omega) = \{\mu(\alpha_{ij}, \ldots, \alpha_{nj})(\omega) : j \in \mathbb{Z}^+\}
\]

and is called a consensus sequence on \([\mathcal{M}(\Omega, A^*)]^n\).

We can obtain the following result:

**Theorem 3 (Strong law of large numbers for a sequence of random strings)** If \( \{\sigma_n = \{\alpha_{nj} : j \in \mathbb{Z}^+\} : n \in \mathbb{Z}^+\} \subset [\mathcal{M}(\Omega, A^*)] \) holds, \( \{\alpha_{nj} : n \in \mathbb{Z}^+\} \) are independently and identically distributed for each \( j \in \mathbb{Z}^+ \), and \( (\sigma_1, \ldots, \sigma_n) \in [\mathcal{M}(\Omega, A^*)]^n \) holds for each \( n \in \mathbb{Z}^+ \), then there exists \( n_0 \in \mathbb{Z}^+ \) such that

\[
\mu(\sigma_1, \ldots, \sigma_n) = m'(\sigma_1) \text{ a.s.}
\]

holds for any \( n \geq n_0 \).
See the online supplemental material of [18] for the proof of this result.

References

[1] M. A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Autom. Remote Control*, 25:821–837, 1964.

[2] S. Bandyopadhyay and R. Mitra. TargetMiner: microRNA target prediction with systematic identification of tissue-specific negative examples. *Bioinformatics*, 25(20):2625–2631, 2009.

[3] A. A. Bogan and K. S. Thorn. Anatomy of hot spots in protein interfaces. *J. Mol. Biol.*, 280(1):1–9, 1998.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Houssler, editor, *Proc. 5th Annu. Workshop Comput. Learn. Theory*, pages 144–152, 1992.

[5] L. E. J. Brouwer. Beweis des Jordanschen Satzes für den n-dimensionalen Raum. *Mathematische Annalen*, pages 314–319, 1911.

[6] S. K. Burley and G. A. Petsko. Weakly polar interactions in proteins. In C. B. Anfinsen, J. T. Edsall, F. M. Richards, and D. S. Eisenberg, editors, *Advances in Protein Chemistry*, volume 39, pages 125–189. Academic Press, Waltham, MA, 1988.

[7] C. Cortes and V. N. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

[8] M. Crochemore and W. Rytter. *Jewels of Stringology*. World Scientific, 2002.

[9] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Adv. Neural Inf. Process. Syst. 9*, pages 155–161. MIT Press, Cambridge, MA, 1997.

[10] A. J. Enright, B. John, U. Gaul, T. Tuschl, C. Sander, and D. S. Marks. Microrna targets in drosophila. *Genome Biol.*, 5(1):R1, 2004.
[11] D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, 1997.

[12] D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California, Santa Cruz, Santa Cruz, CA, 1999.

[13] P. B. Horton and M. Kanehisa. An assessment of neural network and statistical approaches for prediction of E. coli promoter sites. *Nucleic Acids Res.*, 20(16):4331–4338, 1992.

[14] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[15] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[16] B. John, A. J. Enright, A. Aravin, T. Tuschi, C. Sander, and D. S. Marks. Human microRNA targets. *PLoS Biol.*, 2(11):1862–1879, 2004.

[17] C. Jordan. *Cours d’analyse de l’École Polytechnique.* Gauthier-Villars, Paris, 1887.

[18] H. Koyano and H. Kishino. Quantifying biodiversity and asymptotics for a sequence of random strings. *Phys. Rev. E*, 81(6):061912, 2010.

[19] A. Krek, D. Grün, M. N. Poy, R. Wolf, L. Rosenberg, E. J. Epstein, P. MacMennamin, I. da Piedade, K. C. Gunsalus, M. Stoffel, and N. Rajewsky. Combinatorial microRNA target predictions. *Nat. Genet.*, 37(5):495–500, 2005.

[20] N. I. Larsen, J. Engelbrecht, and S. Brunak. Analysis of eukaryotic promoter sequences reveals a systematically occurring CT-signal. *Nucleic Acids Res.*, 23(7):1223–1230, 1995.

[21] C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Adv. Neural Inf. Process. Syst. 15*, pages 1417–1424. MIT Press, Cambridge, MA, 2003.

[22] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *J. Mach. Learn. Res.*, 5:1435–1455, 2004.
[23] C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In R. B. Altman, A. K. Dunker, L. Hunter, T. E. Klein, and K. Lauderdale, editors, *Proc. Pacific Symp. Biocomput.*, volume 7, pages 566–575, 2002.

[24] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.*, 10:707–710, 1966.

[25] B. P. Lewis, I.-h. Shih, M. W. Jones-Rhoades, D. P. Bartel, and C. B. Burge. Prediction of mammalian microRNA targets. *Cell*, 115(7):787–798, 2003.

[26] H. Li and T. Jiang. A class of edit kernels for SVMs to predict translation initiation sites in eukaryotic mRNAs. *J. Comput. Biol.*, 12(6):702–718, 2005.

[27] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernel. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Adv. Neural Inf. Process. Syst. 13*. MIT Press, Cambridge, MA, 2001.

[28] B. Ma, T. Elkayam, H. Wolfson, and R. Nussinov. Protein-protein interactions: Structurally conserved residues distinguish between binding sites and exposed protein surfaces. *Proc. Natl. Acad. Sci. USA*, 100(10):5772–5777, 2003.

[29] H. B. Mann and A. Wald. On stochastic limit and order relationships. *Ann. Math. Stat.*, 14:217–226, 1943.

[30] R. Mosca, A. Céol, A. Stein, R. Olivella, and P. Aloy. 3did: A catalog of domain-based interactions of known three-dimensional structure. *Nucleic Acids Res.*, 42(D1):D374–D379, 2014.

[31] G. Paaß, E. Leopold, M. Larson, J Kindermann, and S. Eickeler. SVM classification using sequences of phonemes and syllables. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proc. 6th Eur. Conf. Principles Data Min. Knowl. Discov.*, pages 373–384. Springer, 2002.

[32] A. Rodriguez, S. Griffiths-Jones, J. L. Ashurst, and A. Bradley. Identification of mammalian microRNA host genes and transcription units. *Genome Res.*, 14(10a):1902–1910, 2004.
[33] P. W. Rose, B. Beran, C. Bi, W. F. Bluhm, D. Dimitropoulos, D. S. Goodsell, A. Prlić, M. Quesada, G. B. Quinn, J. D. Westbrook, J. Young, B. Yukich, C. Zardecki, H. M. Berman, and P. E. Bourne. The RCSB Protein Data Bank: Redesigned web site and web services. *Nucleic Acids Res.*, 39(suppl 1):D392–D401, 2011.

[34] H. Saigo, J.-P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

[35] K. E. van Holde, W. C. Johnson, and P. S. Ho. *Principles of Physical Biochemistry*. Prentice Hall, Upper Saddle River, NJ, 2006.

[36] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[37] J.-P. Vert. Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings. In R. B. Altman, A. K. Dunker, L. Hunter, T. E. Klein, and K. Lauderdale, editors, *Proc. Pacific Symp. Biocomput.*, volume 7, pages 649–660, 2002.

[38] S. V. N. Vishwanathan and A. J. Smola. Fast kernels for string and tree matching. In K. Tsuda, B. Schölkopf, and J.-P. Vert, editors, *Kernel Methods in Computational Biology*, pages 113–130. MIT Press, Cambridge, MA, 2004.

[39] C. Watkins. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Computer Science Department, University of London, Royal Holloway, 1999.

[40] M. Yousef, S. Jung, A. V. Kossenkov, L. C. Showe, and M. K. Showe. Naïve Bayes for microRNA target predictions-machine learning for microRNA targets. *Bioinformatics*, 23(22):2987–2992, 2007.

[41] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.