Kernel Design for RNA Classification Using Support Vector Machines

Jason T. L. Wang* and Xiaoming Wu†

Abstract

Support vector machines (SVMs) are a state-of-the-art machine learning tool widely used in speech recognition, image processing and biological sequence analysis. An essential step in SVMs is to devise a kernel function to compute the similarity between two data points. In this paper we review recent advances of using SVMs for RNA classification. In particular we present a new kernel that takes advantage of both global and local structural information in RNAs and uses the information together to classify RNAs. Experimental results demonstrate the good performance of the new kernel and show that it outperforms existing kernels when applied to classifying non-coding RNA sequences.

1 Introduction

Ribonucleic acid (RNA) is a molecule that can play several different roles in nature [31, 32, 33]. RNA sequences, composed of bases `a` (adenine), `c` (cytosine), `g` (guanine) and `u` (uracil), have a tendency to fold back on themselves to form double-stranded structures. Most commonly, Watson-Crick base pairs between `a` and `u` and between `g` and `c` are formed, but sometimes less stable pairings are also possible, such as `g` with `u`. These base pairs together with other morphological features such as bulge loops, hairpin loops, internal loops and multi-branched loops form structures known as RNA secondary structures. The secondary structure of an RNA molecule determines its 3D shape and hence its functional role [15]. Extensive research has been performed to investigate the structure-function relationship in RNAs [21, 22, 37].

To facilitate this investigation and related research, Griffiths-Jones *et al.* [9] developed a database of RNA families, called Rfam. This database can be used to annotate sequences (including complete genomes) for homologues to known non-coding RNAs. Rfam makes use of a large amount of available data, especially published multiple sequence alignments, and repackages these data in a single searchable and sustainable resource. The families, or classes, of sequences in Rfam have been widely used by researchers in testing their RNA mining and classification algorithms [13].

In this paper we review some of the recent RNA classification algorithms and propose a new one. Classification is a supervised learning process, which is to assign an unlabeled object to one of the existing classes or families of objects. This process has many applications in bioinformatics, for example, in detecting special signals in DNA sequences [17, 30, 35, 36] or in organizing protein sequences into (super)families [26, 27, 29]. The RNA classification methods presented in this paper can be used in automated analysis and categorization of uncharacterized RNA molecules or in

*Bioinformatics Center and Department of Computer Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA.

†Department of Computer Science, Math & Engineering, Shepherd University, Shepherdstown, WV 25443, USA.

genome-wide screens for RNA molecules from existing families. The presented methods are based on a state-of-the-art machine learning tool, namely support vector machines (SVMs), which are briefly described below.

1.1 Support Vector Machines

Support vector machines, originally proposed by Vapnik [25], have been highly successful in many application areas such as text categorization, handwriting recognition and face detection. SVMs are based on a simple yet intuitive idea, namely, the best hyperplane to separate two groups of points in a Euclidean space R^n is the hyperplane with the maximum margin (Figure 1).

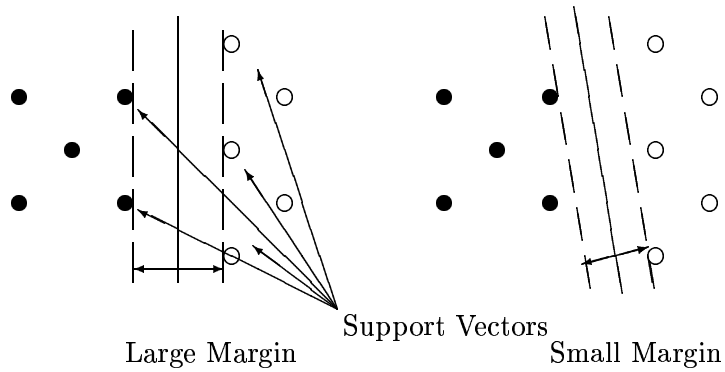


Figure 1: Maximization of the margin of the hyperplane separating two groups of points.

For data points that cannot be separated by a linear hyperplane, we can map the data points to a feature space so that the images of the data points in the feature space can be linearly separated. In general, the dimension of the feature space can be very high and the mapping would be highly complicated. Fortunately, we do not need to construct the mapping explicitly; as in SVMs, the mapping is implicit in the kernel function used in a quadratic programming problem. It is only necessary to solve the quadratic programming problem to accomplish the learning task of SVMs. In fact, the kernel function represents the inner product of images of two data points in the feature space. Some commonly used kernel functions include

$$\begin{aligned} \text{Gaussian RBF} \qquad & K(x,y) = e^{(-|x-y|^2/c)} \\ \text{Polynomial} \qquad & K(x,y) = ((x \cdot y) + \theta)^p \\ \text{Sigmoidal} \qquad & K(x,y) = \tanh(\kappa(x \cdot y) + \theta) \end{aligned}$$

The theoretical foundation of SVMs is the Vapnik-Chervonenkis theory and structural risk minimization principle [25]. For more information on SVMs, see [5]. For information on implementations of SVMs, see [3, 19].

Recently SVMs have been applied to many pattern recognition and classification problems in biological sequence analysis, ranging from protein homology detection, microarray gene expression analysis, to recognition of translation start sites [18]. Noble [18] stated the motivations behind the application of SVMs to computational biology and bioinformatics. First, many biological problems involve high-dimensional, noisy data, for which SVMs are known to behave well compared to other

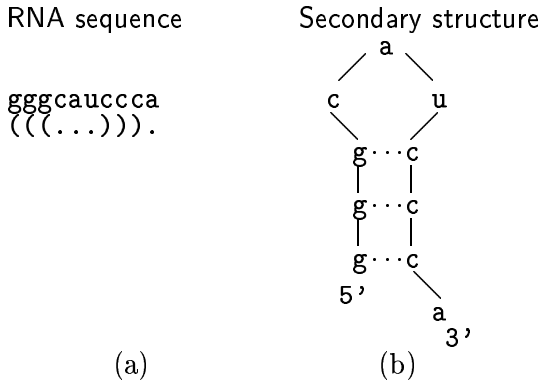


Figure 2: (a) A hypothetical RNA sequence where parentheses represent base-pairings. (b) The secondary structure of the RNA sequence in (a).

statistical or machine learning methods. Second, in contrast to most machine learning methods, kernel methods like SVMs can easily handle non-vector inputs, such as variable-length sequences or graphs. These types of data are common in computational biology and bioinformatics.

This paper discusses the application of SVMs to automatically classifying RNA sequences. An SVM for this purpose is based on a kernel function that computes the similarity between two RNA sequences. Existing methods for computing the similarity (or distance) of two strings such as edit distance [23] or alignment score [6] are not sufficient for RNA sequences because they do not take into account the secondary structures of RNA. An RNA molecule is not only a string of letters taken from the alphabet $\Sigma = \{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{u}\}$; it also folds itself into a-u, c-g and g-u pairs. These base pairs form the secondary structure of RNA that consists of geometric patterns such as bulges, hairpins and stacked pairs (Figure 2). The secondary structure of an RNA molecule determines its 3D shape and hence its functional role. Therefore an important step of using SVMs for RNA classification is to devise a kernel function that takes into account the secondary structure information of RNA.

There are different ways to represent RNA secondary structures, for example, using dual graphs [7] or trees [16]. Kernels that deal with structured data such as graphs and trees are proposed in [8, 14]. Only a handful of kernels are specifically devised for RNA secondary structures. For example, Tsuda *et al.* [24] associated each base in an RNA sequence with a state L, R or P (for the meaning of these states, see Section 2 or [24]), which reflects the secondary structure information of RNA. For RNA sequences with known secondary structures, Tsuda's method considers the occurrence frequencies of bi-grams such as ((a, L), (c, P)) in the sequences. Here (a, L) and (c, P) are base-state pairs. For RNA sequences with unknown secondary structures, Tsuda's method uses a probabilistic model, called stochastic context free grammar, to estimate the probability that each state L, R or P occurs at each base and uses this probability to calculate the kernel between two RNA sequences.

Another kernel for RNA was proposed in [13]. In this approach, an RNA sequence is first converted to a graphical representation, called labeled dual graph, based on its secondary structure. The similarity of two labeled dual graphs, and thus the kernel function, can then be computed by taking random walks in the two graphs and measuring similarities between sequences of labels resulted from these random walks.

In this paper we present a new kernel-based approach to RNA classification using support vector machines. We extract recurring substrings from RNA molecules; part of the kernel is based on these

recurring substrings. The other part of the kernel is based on counting bi-grams in RNA molecules, as done in Tsuda's method [24]. The idea behind the proposed kernel is to utilize both global and local structural information of RNA for classification.

The rest of the paper is organized as follows. Section 2 reviews Tsuda's method [24]. Section 3 describes the kernel based on labeled dual graphs [13]. Section 4 presents the new kernel based on bi-gram occurrence frequencies and recurring substrings. Section 5 reports experimental results. Section 6 concludes the paper.

2 Count Kernels and Marginalized Count Kernels

In [24], Tsuda *et al.* developed a kernel for RNA sequences with either known secondary structures or unknown secondary structures. It was assumed that an RNA secondary structure involves only canonical base pairs a-u and g-c.

2.1 RNA Sequences with Known Secondary Structures

An RNA secondary structure can be represented by a context free grammar (CFG). For example, the RNA secondary structure in Figure 2 can be represented as generative rules of the following CFG:

$$S \to R_1, R_1 \to P_1\mathtt{a}, P_1 \to \mathtt{g}P_2\mathtt{c}, P_2 \to \mathtt{g}P_3\mathtt{c},$$
$$P_3 \to \mathtt{g}L_1\mathtt{c}, L_1 \to \mathtt{c}L_2, L_2 \to \mathtt{a}L_3, L_3 \to \mathtt{u}E$$

Here P is the state that emits a canonical base pair. L and R are states that emit only one base to the left or right. S and E are special states that represent the beginning and end, respectively, of the CFG rules. The generative rules can be represented as a state-path in a matrix, called *CFG matrix*, which demonstrates how the states in the CFG are associated with the bases in the RNA secondary structure. For example, Figure 3 shows the CFG matrix of the RNA sequence in Figure 2. In Figure 3, P at $(1, 9)$ corresponds to the first base g and the ninth base c; L at $(4, 6)$ corresponds to the fourth base c; R at $(1, 10)$ corresponds to the tenth base a.

We can define two "count kernels" with respect to the CFG representation: the first order count kernel and the second order count kernel. In these kernels we convert an RNA sequence to a vector and the kernel of two RNA sequences is defined as the inner product of their corresponding vectors. The CFG matrix plays a central role in constructing the count kernels, as explained below.

In the first order count kernel, we count the occurrence numbers of base-state combinations and use them as the coordinates of the *count feature vector* of an RNA sequence. For example, for the RNA sequence in Figure 2, the occurrence numbers of base-state combinations are 1 for (a, R), 3 for (gc, P), 1 for (c, L), 1 for (a, L), and 1 for (u, L) respectively; the occurrence number of other combinations such as (au, P) and (g, L) is 0. More formally, let $x = x_1, x_2, \ldots, x_n$ be an RNA sequence and let W be the CFG matrix of the sequence. The count feature vector of $z = \{x, W\}$ is defined as follows:

$$c_P^{ab}(z) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\delta[W(i,j) = P, x_i = a, x_j = b] \quad (1)$$

$$c_L^{a}(z) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=i}^{n}\delta[W(i,j) = L, x_i = a] \quad (2)$$

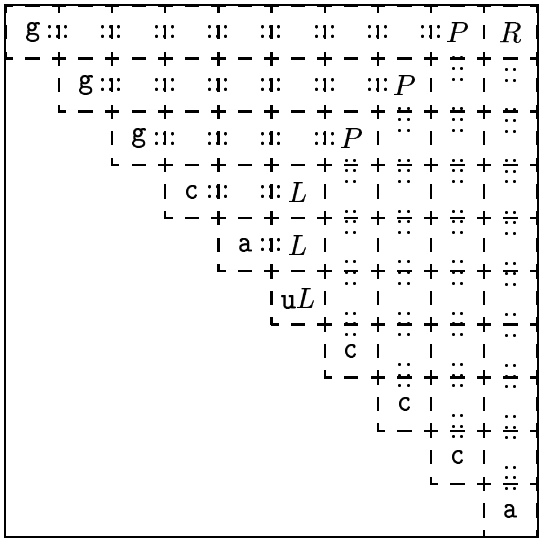


Figure 3: The CFG matrix of the RNA sequence in Figure 2.

$$c_R^b(z) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=i}^{n} \delta[W(i,j) = R, x_j = b] \tag{3}$$

Here $z = \{x, W\}$ is the sequence x combined with the CFG matrix W; a, b are nucleotides; $\delta[W(i,j) = P, x_i = a, x_j = b] = 1$ if $W(i,j) = P, x_i = a, x_j = b$ and 0 otherwise; $1/n$ is a normalization factor with respect to the sequence length n.

The *first order count kernel* of two RNA sequences is defined as the inner product of their count feature vectors:

$$K(z, z') = \sum_{V \in \{P,L,R\}} C_V(z, z'),$$

where

$$C_V(z,z') = \begin{cases} V = P : \sum_{ab \in \Omega} c_P^{ab}(z) c_P^{ab}(z'), & \Omega = \{\texttt{au, ua, gc, cg}\} \\ V = L : \sum_{a \in B} c_L^{a}(z) c_L^{a}(z'), & B = \{\texttt{a, u, c, g}\} \\ V = R : \sum_{b \in B} c_R^{b}(z) c_R^{b}(z') & \end{cases}$$

The count feature vector for the second order count kernel is obtained by counting the occurrence numbers of combinations of two consecutive base-state pairs in z. The following shows the formal definition of the count feature vector for the second order count kernel (only three out of nine equations are listed here):

$$c_{PP}^{abcd}(z) = \frac{2}{n} \sum_{i=1}^{n-1-\Delta_l^P-\Delta_r^P} \sum_{j=i+1+\Delta_l^P+\Delta_r^P}^{n} d_{PP}^{abcd}(i,j) \tag{4}$$

$$d_{PP}^{abcd}(i,j) \equiv \delta[W(i,j) = P, W(i+\Delta_l^P, j-\Delta_r^P) = P, x_i = a, x_j = b, x_{i+\Delta_l^P} = c, x_{j-\Delta_r^P} = d]$$

Values of $\Delta^V_{l|r}$ represent whether state V emits a symbol to the left or to the right; "1" indicates emission and "0" indicates no emission.

V	P	L	R
Δ^V_l	1	1	0
Δ^V_r	1	0	1

Figure 4: Definition of $\Delta^V_{l|r}$.

$$c^{abc}_{PL}(z) = \frac{2}{n}\sum_{i=1}^{n-1-\Delta^P_l}\sum_{j=i+1+\Delta^P_r}^{n} d^{abc}_{PL}(i,j) \quad (5)$$

$$d^{abc}_{PL}(i,j) \equiv \delta[W(i,j)=P, W(i+\Delta^P_l, j-\Delta^P_r)=L, x_i=a, x_j=b, x_{i+\Delta^P_l}=c]$$

$$c^{abd}_{PR}(z) = \frac{2}{n}\sum_{i=1}^{n-1-\Delta^P_l}\sum_{j=i+1+\Delta^P_r}^{n} d^{abd}_{PR}(i,j) \quad (6)$$

$$d^{abd}_{PR}(i,j) \equiv \delta[W(i,j)=P, W(i+\Delta^P_l, j-\Delta^P_r)=R, x_i=a, x_j=b, x_{j-\Delta^P_r}=d]$$

Here Δ^V_l and Δ^V_r are flags of a binary value indicating whether state V emits a symbol to the left or to the right (see the table in Figure 4).

The *second order count kernel* of two RNA sequences is defined as the inner product of their count feature vectors:

$$K(z,z') = \sum_{VY\in\Psi} C_{VY}(z,z'), \quad (7)$$

where

$$\Psi = \{PP, PL, PR, LP, LL, LR, RP, RL, RR\}$$

$$C_{VY}(z,z') = \begin{cases} VY = PP : \sum_{abcd\in\Omega\times\Omega} c^{abcd}_{PP}(z)c^{abcd}_{PP}(z') \\ VY = PL : \sum_{abc\in\Omega\times B} c^{abc}_{PL}(z)c^{abc}_{PL}(z') \\ VY = PR : \sum_{abd\in\Omega\times B} c^{abd}_{PR}(z)c^{abd}_{PR}(z') \\ \vdots \end{cases}$$

2.2 RNA Sequences with Unknown Secondary Structures

If we do not know the secondary structure of an RNA sequence *a priori*, we can use a stochastic context free grammar (SCFG) to estimate the probability that each state appears at entry (i,j) of the CFG matrix of the sequence. We then use these state probabilities instead of explicit states to construct count feature vectors. These vectors are called *marginalized count feature vectors* and kernels based on these vectors are called *marginalized count kernels* (MCKs). Let $x = x_1, x_2, \ldots, x_n$ be an RNA sequence and let W be the CFG matrix of the sequence. The *first order marginalized count feature vector* of $z = \{x, W\}$ is defined as follows:

$$g^{ab}_P(z) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n} p(W(i,j)=P|x)\delta[x_i=a, x_j=b] \quad (8)$$

$$g^{a}_L(z) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=i}^{n} p(W(i,j)=L|x)\delta[x_i=a] \quad (9)$$

$$g_R^b(z) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=i}^{n} p(W(i,j)=R|x)\delta[x_j=b] \tag{10}$$

Here $p(W(i,j)=V)$ is the probability that state V occurs at $W(i,j)$ and $0 \le p(W(i,j)=V) \le 1$. The *first order marginalized count kernel* of two RNA sequences is defined as

$$K(z,z') = \sum_{V\in\{P,L,R\}} G_V(z,z'), \tag{11}$$

where

$$G_V(z,z') = \begin{cases} V=P: \sum_{ab\in\Omega} g_P^{ab}(z) g_P^{ab}(z'), & \Omega = \{\texttt{au, ua, gc, cg}\} \\ V=L: \sum_{a\in B} g_L^{a}(z) g_L^{a}(z'), & B = \{\texttt{a, u, c, g}\} \\ V=R: \sum_{b\in B} g_R^{b}(z) g_R^{b}(z') \end{cases}$$

Let $\xi_{VY}(i,j)$ be the joint probability of having state V at $W(i,j)$ followed by state Y at $W(i+\Delta_l^V, j-\Delta_r^V)$. We have

$$\xi_{VY}(i,j) \equiv p(W(i,j)=V, W(i+\Delta_l^V, j-\Delta_r^V)=Y|x).$$

We define the *second order marginalized count feature vector* of an RNA sequence as follows (only three out of nine equations are listed here):

$$g_{PP}^{abcd}(z) = \frac{2}{n}\sum_{i=1}^{n-1-\Delta_l^P-\Delta_r^P}\sum_{j=i+1+\Delta_l^P+\Delta_r^P}^{n} h_{PP}^{abcd}(i,j) \tag{12}$$

$$h_{PP}^{abcd}(i,j) \equiv \xi_{PP}(i,j)\delta[x_i=a, x_j=b, x_{i+\Delta_l^P}=c, x_{j-\Delta_r^P}=d]$$

$$g_{PL}^{abc}(z) = \frac{2}{n}\sum_{i=1}^{n-1-\Delta_l^P}\sum_{j=i+1+\Delta_r^P}^{n} h_{PL}^{abc}(i,j) \tag{13}$$

$$h_{PL}^{abc}(i,j) \equiv \xi_{PL}(i,j)\delta[x_i=a, x_j=b, x_{i+\Delta_l^P}=c]$$

$$g_{PR}^{abd}(z) = \frac{2}{n}\sum_{i=1}^{n-1-\Delta_l^P}\sum_{j=i+1+\Delta_r^P}^{n} h_{PR}^{abd}(i,j) \tag{14}$$

$$h_{PR}^{abd}(i,j) \equiv \xi_{PR}(i,j)\delta[x_i=a, x_j=b, x_{j-\Delta_r^P}=d]$$

The *second order marginalized count kernel* of two RNA sequences is defined as:

$$K(z,z') = \sum_{VY\in\Psi} G_{VY}(z,z'), \tag{15}$$

where

$$\Psi = \{PP, PL, PR, LP, LL, LR, RP, RL, RR\}$$

$$G_{VY}(z,z') = \begin{cases} VY=PP: \sum_{abcd\in\Omega\times\Omega} g_{PP}^{abcd}(z) g_{PP}^{abcd}(z') \\ VY=PL: \sum_{abc\in\Omega\times B} g_{PL}^{abc}(z) g_{PL}^{abc}(z') \\ VY=PR: \sum_{abd\in\Omega\times B} g_{PR}^{abd}(z) g_{PR}^{abd}(z') \\ \vdots \end{cases}$$

For more detailed information regarding how to use an SCFG to obtain the probabilities $p(W(i,j)=V)$ and $p(W(i,j)=V, W(i+\Delta_l^V, j-\Delta_r^V)=Y|x)$, see [6, 24].

In [24], three approaches for classifying RNA sequences were compared in a computational experiment: SVMs based on the first order MCK, SVMs based on the second order MCK, and a pure SCFG likelihood method. The dataset used in the experiment contained 74 sequences extracted from three human tRNA families. The experimental results showed that for all the three families, SVMs based on the second order MCK performed consistently better than the other two approaches.

3 Kernel Based on Labeled Dual Graphs

The idea behind this approach is to first convert an RNA secondary structure into a graph representation called labeled dual graph (LDG). The LDG is expected to capture some of the key topological features of the RNA secondary structure. Then a kernel is constructed to compute the similarity of the LDG representations of two RNA secondary structures. Karklin *et al.* [13] recently adopted this approach in which they used the marginalized kernel for labeled graphs [14] to classify non-coding RNA sequences.

3.1 Labeled Dual Graphs

The dual graph [7] representation of RNA captures important topological characteristics of RNA secondary structures such as the number and relative positions of helical regions. In this representation, a helical region is represented by a vertex in the graph. Single RNA strands are represented by edges; for example, an internal loop, multi-branched loop or bulge is represented by an edge connecting two vertices representing two helical regions and an external loop is represented by an edge connecting a vertex to itself. Thus a dual graph is a multi-graph in which two vertices are connected by at most two edges. We can attach biologically meaningful labels to the vertices and edges in a dual graph. For example, a vertex can be labeled with the number of base-pairs of the corresponding helical region and an edge can be labeled with the length of the corresponding single RNA strand (number of nucleotides) and the type of the corresponding loop (internal/external). The resulting graph is called a *labeled dual graph* (LDG). Figure 5 shows two RNA secondary structures and their LDG representations.

3.2 Marginalized Kernel for Labeled Dual Graphs

With the marginalized kernel for labeled dual graphs, we take random walks in two graphs and calculate the similarity of the label sequences produced by these random walks [14]. A score based on the similarity of the label sequences reflects the similarity between the two graphs, which is defined as the marginalized kernel for the two graphs.

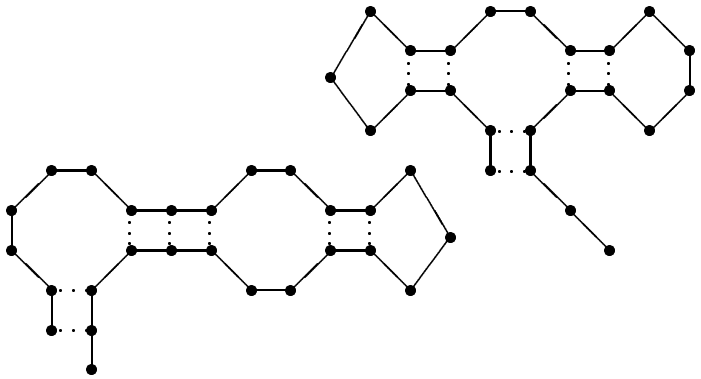
In order to generate random walks in an LDG, we assume a uniform starting probability over all vertices, a uniform probability to transit from a vertex to one of its neighbors, and a constant probability to terminate the walks at any step [13]. Suppose $z = v_1, e_{12}, v_2, e_{23}, v_3, \ldots$ and $z' = v'_1, e'_{12}, v'_2, e'_{23}, v'_3, \ldots$ are two sequences of vertex and edge labels produced by random walks h, h' in graphs G and G', respectively. The similarity measure between z and z' is defined as [13]:

$$K_z(z, z') = K_v(v_1, v'_1)K_e(e_{12}, e'_{12})K_v(v_2, v'_2)\ldots \tag{16}$$

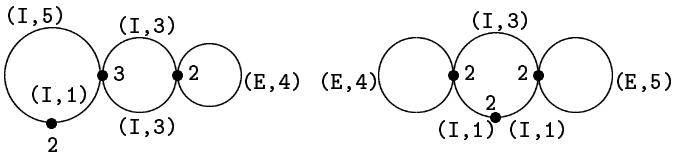
Here K_v and K_e are kernels defined for vertices and edges respectively. For two vertices v_i and v_j,

$$K_v(v_i, v_j) = \exp(-(\log(v_i/v_j))^2) \tag{17}$$

(a)



(b)



(c)

$L=1$	$\{2\},\{3\},\{2\}$	$\{2\},\{2\},\{2\}$
$L=2$	$\{2,(I,1),3\},\{2,(E,4),2\},\ldots$	$\{2,(I,1),2\},\{2,(E,4),2\},\ldots$
$L=3$	$\{\mathbf{2},(\mathbf{I},\mathbf{1}),\mathbf{3},(\mathbf{I},\mathbf{3}),\mathbf{2}\},\ldots$	$\{\mathbf{2},(\mathbf{I},\mathbf{1}),\mathbf{2},(\mathbf{I},\mathbf{3}),\mathbf{2}\},\ldots$
$L=4$	$\{2,(I,5),3,(I,3),2,(E,4),2\},$ $\ldots$	$\{2,(I,1),2,(E,4),2,(I,3),2\},$ $\ldots$
$\vdots$		$\vdots$

(d)

$$\begin{aligned} K_z &= K_v(2,2)K_e(1,1)K_v(3,2)K_e(3,3)K_v(2,2) \\ &= \exp(-(\log\frac{2}{2})^2-(\log\frac{1}{1})^2-(\log\frac{3}{2})^2-(\log\frac{3}{3})^2-(\log\frac{2}{2})^2) \end{aligned}$$

Figure 5: (a) Secondary structure diagrams of two RNA molecules. (b) Labeled dual graph representations of the RNA secondary structures in (a). Here a vertex represents a helical region and an edge represents a single RNA strand. (c) A subset of label sequences produced by taking random walks in the two labeled dual graphs in (b). Here L is the length of a random walk. (d) An example of the label sequence kernel applied to the highlighted pair of label sequences in (c).

For two edges e_{ij} and e_{kl}, representing two loops of the same type (internal or external), the kernel is defined as [13]:

$$K_e(e_{ij}, e_{kl}) = \exp(-(\log(e_{ij}/e_{kl}))^2) \tag{18}$$

If two loops are of different types, the kernel K_e is defined as 0. Finally, we calculate the expected value of $K_z(z, z')$ over all possible random walks, weighted by the probability generating the random walks. The result is defined as the marginalized kernel of G and G', denoted $K(G, G')$:

$$K(G, G') = < K_z(z, z') >_{h,h'} \tag{19}$$

Obviously, $0 \leq K(G, G') \leq 1$. The more similar the two graphs G and G', the higher the score $K(G, G')$ will be.

Karklin *et al.* [13] conducted experiments on RNA sequences taken from families in the Rfam database [9] to evaluate the performance of this kernel. An SVM classifier based on this kernel was trained to find out if the kernel was able to pick up important topological information of RNA secondary structures so as to differentiate sequences in a family from random sequences with the same di-nucleotide statistics. In their experiments, sequences of a family were treated as positive samples. The nucleotides in those sequences were shuffled randomly to generate negative samples. For 22 of 25 families tested, A_{ROC} was greater than 0.7. (A_{ROC} is the area under the receiver operating characteristic (ROC) curve, which is a general measure of the discrimination ability of a classification algorithm [2].) A multi-class classifier based on the kernel was trained and tested on nine families in the Rfam database. The experimental results showed good performance of this classifier for many of the tested families.

4 A New Kernel

We propose here a new kernel that takes into account both global and local structural information of RNA molecules. The global structural information is obtained by counting occurrence numbers of bi-grams in the molecules and the local structural information is obtained by considering recurring substrings in these molecules. The meaning of these terminologies will be made clearer in the following subsections. This kernel is an extension of our previously developed method [27] in which Bayesian neural networks (BNNs) were employed to classify protein sequences. In that method, the occurrence frequencies of all bi-grams in a protein sequence were calculated, which formed features reflecting the global structural information of the protein sequence. Then, the local motifs of a given protein family were extracted. These motifs were used to calculate another feature that reflected the local structural information of a protein sequence. The features of the training sequences used in protein classification were fed into a BNN to train its weight parameters.

In the case of proteins, the number of bi-grams is large. In fact, there are $20 \times 20 = 400$ possible bi-grams because there are 20 different amino acids. The number of local motifs may also be very large. If all these features are used to train the BNN, there would be too many weight parameters. This would make the training process hard, leading to a phenomenon called "curse of dimensionality." Therefore, a procedure was taken [27] to select only those features that, by intuition, had the greatest capability to differentiate sequences in a family from sequences outside the family. Here, the "curse of dimensionality" is not a serious problem because SVMs are known to perform well even with high-dimensional data. Furthermore, in applications like ours, SVMs usually yield better performance than BNNs [18].

4.1 Extracting Features for Global Structural Information

First, we obtain the secondary structure of an RNA sequence using the Vienna RNA folding prediction package [34]. Second, for each RNA molecule, we transform it to a sequence that reflects both the primary structure and the secondary structure of the molecule. The transformation is performed as follows:

$$\begin{array}{lll} (\mathtt{a} \rightarrow A & \mathtt{a}) \rightarrow B & \mathtt{a}. \rightarrow C \\ (\mathtt{c} \rightarrow D & \mathtt{c}) \rightarrow E & \mathtt{c}. \rightarrow F \\ (\mathtt{g} \rightarrow G & \mathtt{g}) \rightarrow H & \mathtt{g}. \rightarrow I \\ (\mathtt{u} \rightarrow U & \mathtt{u}) \rightarrow V & \mathtt{u}. \rightarrow W \end{array}$$

For example, with this transformation, nucleotide a that is on the left hand side of a base pair is converted to A. If nucleotide a is on the right hand side of a base pair, it is converted to B. If nucleotide a is unpaired, it is converted to C. Therefore, each RNA sequence can be transformed to a sequence of letters from alphabet $\Sigma' = \{A, B, C, D, E, F, G, H, I, U, V, W\}$. As an example, the RNA sequence in Figure 2 is transformed to

$$GGGFCWEEEC \tag{20}$$

In the following, the new sequence obtained based on this transformation scheme will be referred to as an *adjusted RNA sequence*.

We map each adjusted RNA sequence to a vector, called the *feature vector* of the sequence, in a Euclidean space R^k. The dimension k will be determined later. Each coordinate of the feature vector represents a feature of the sequence. After all adjusted RNA sequences are mapped to vectors in R^k, the kernel function of two adjusted RNA sequences can be as simple as the inner product of their corresponding feature vectors, or can be one of the commonly used kernel functions (Gaussian RBF, polynomial, or sigmoidal) defined in Section 1. In our approach, we use the Gaussian RBF kernel function.

Similar to [24, 27], we adopt the bi-gram model to calculate the first set of features in a feature vector. The bi-gram model is known to be able to represent implicit but essential information for identifying biological sequences [24]. In our approach, a bi-gram is a combination of any two letters taken from the alphabet Σ' for adjusted RNA sequences. There are 12 letters in Σ', so the total number of bi-grams is $12 \times 12 = 144$. The occurrence frequency of a bi-gram in an adjusted RNA sequence is defined as the number of occurrences of the bi-gram in the sequence divided by the total number of bi-grams in the sequence, which is obviously the length of the sequence minus 1.

For example, the occurrence frequency of GF in the adjusted RNA sequence shown in (20) is $1/9 = 0.111111$. The occurrence frequency of EE in this adjusted RNA sequence is $2/9 = 0.222222$. The first 144 coordinates of the feature vector of an adjusted RNA sequence are the occurrence frequencies of all 144 bi-grams in the sequence. We can adopt the formal notation in [24]; namely, letting $z = z_1, z_2, \ldots, z_n$ be an adjusted RNA sequence, the coordinate for a bi-gram e.g. GF, is defined as:

$$c_{GF} = 1/(n-1) \sum_{i=1}^{n-1} \delta[z_i = G, z_{i+1} = F] \tag{21}$$

where $\delta[z_i = G, z_{i+1} = F] = 1$ if $z_i = G, z_{i+1} = F$ and 0 otherwise; n is the length of the sequence. These coordinates can be thought of as features representing global RNA structural information. The second set of coordinates of the feature vector, discussed below, represents local RNA structural information.

4.2 Extracting Features for Local Structural Information

A recurring substring is a substring that occurs frequently in a set of bio-sequences. If significantly many sequences in a set share a similar substring, we can infer that the substring doesn't occur by chance [33]. Instead, the substring is related to some biological function. For example, the biological function of a shared substring might be a common protein binding site in the case of DNA sequences, or the active site of related enzymes in the case of protein sequences [1]. The problem of finding interesting recurring substrings in biological sequences is important and has been studied extensively. There are a number of algorithms that can be used, for example, MEME [1] and PRATT [12]; other algorithms can be found in [4, 10].

In our approach, we use our previously developed tool sdiscover [28] that is suitable for the application at hand to find recurring substrings. This tool can be downloaded from http://datalab.njit.edu/biodata/ssi/software.shtml. It can find frequently occurring substrings with length greater than or equal to *Len* that occur in at least *Occur* sequences within *Mut* edit operations in a set of sequences where *Len*, *Occur* and *Mut* are user-specified parameters. The discovered substrings can be of several different forms. In the study presented here, a substring is simply a segment of letters taken from the alphabet Σ'. An edit operation can be an insertion, deletion or modification of a letter in our case.

Given a set of sequences, sdiscover works by selecting a small subset of the sequences and builds a generalized suffix tree (GST) [11] for the sequences in the subset. GST is a compact data structure that enables us to locate all the substrings of a set of sequences efficiently. Sequences in the small subset are regarded as random samples of the original set. Candidate substrings, which are recurring (or frequently occurring) substrings of these sample sequences, are selected based on a statistical heuristic. The purpose of this heuristic is to select only substrings that are most likely to meet the length (*Len*), occurrence (*Occur*) and similarity (*Mut*) requirements specified by the user. The selected candidate substrings are then compared with all sequences in the original set. Those substrings that satisfy the length (*Len*), occurrence (*Occur*) and similarity (*Mut*) constraints are returned as the final result of sdiscover.

Training and Testing the Proposed SVM Classifier. Our classifier works in two phases. In the training phase, the classifier is trained by labeled adjusted RNA sequences where the label (class) of each sequence is specified. Let S_p be the set of positive training sequences and let S_n be the set of negative training sequences. We use sdiscover to extract recurring, or frequently occurring, substrings from the sequences in S_p (we do not extract recurring substrings from the sequences in S_n). Let N_p be the number of recurring substrings extracted from S_p. Each recurring substring X in S_p contributes to one coordinate in the feature vector of an adjusted RNA sequence, whether it is positive or negative. So, the dimension of the feature vector equals the total number of bi-grams, namely 144, plus N_p.

Let z be an adjusted RNA sequence in $S_p \cup S_n$ and let V be the feature vector of z. We calculate the coordinate in V that corresponds to the recurring substring X in S_p by matching X with z. If X occurs in z within Mut edit operations, the coordinate is $1/N_p$; otherwise, the coordinate is 0. More formally, the coordinate in V that corresponds to the recurring substring X in S_p is defined as

$$c_X = \begin{cases} 1/N_p & \text{if } dist(*X*, z) \leq Mut \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where $dist(*X*, z)$ is the edit distance between the substring $*X*$ and the sequence z and $*$ is a variable length don't care (VLDC) symbol. In calculating $dist(*X*, z)$, the VLDCs may substitute

for zero or more letters in z. The coordinates obtained from recurring substrings can be thought of as features that represent important local structural information shared by the majority of adjusted RNA sequences in the positive training data set.

In the testing phase, the proposed SVM classifier uses the following formula to calculate a score for an unlabeled test sequence:

$$u = \sum_{j=1}^{N} y_j \alpha_j K(x_j, x) - b \tag{23}$$

Here K is the Gaussian RBF kernel function as defined in Section 1 and x is the feature vector of the unlabeled test sequence. Like feature vectors for training sequences, x has $144 + N_p$ dimensions; the coordinates of x corresponding to the recurring substrings in S_p are calculated based on the formula (22). N is the total number of support vectors (the support vectors are special training sequences on the boundary of a class or family); each x_j, $j = 1, 2, \ldots, N$, is a support vector obtained through training the SVM classifier; y_j is the label of x_j where $y_j = +1$ or -1 depending on whether x_j is positive or negative. Each α_j is a Lagrange multiplier and b is a threshold parameter also obtained through training the SVM classifier. These parameters come from the quadratic programming problem associated with the SVM classifier [19]. The score u is the output value for the test sequence.

5 Experiments and Results

5.1 Data and Parameters

To evaluate the performance of the proposed SVM classifier, we used nine families of non-coding RNA sequences taken from the Rfam database [9] (Table 1). For each family, we trained the SVM classifier using the standard *one vs. all* methodology [13]. That is, the sequences in the family were considered as positive sequences while sequences in all other families were considered as negative sequences. For an unlabeled test sequence, we used the SVM classifier trained for each of the nine families to calculate an output value; cf. formula (23). The test sequence was assigned to the family that had the largest output value. We did our classification experiments on the seed sequences [9] of the nine families. The lengths of the sequences in the nine families ranged from 70 to 120 nucleotides. Although most of the nine families had less than 90 sequences, one family, namely Retroviral_psi, had 168 sequences. Half of the sequences in each family were used for training; the other half were used for testing.

The *Len*, *Occur* and *Mut* parameters used in the sdiscover tool were tuned in such a way that the number of substrings extracted by sdiscover was in the range $30 - 60$. In general, the sdiscover tool may find hundreds of recurring substrings from the positive training data set. Although we could use all the substrings to train our SVM classifier, the data is more than needed. In our experiments, we used a heuristic to select $30 - 60$ most "informative" substrings to train the SVM classifier. This heuristic is based on our previously developed minimum description length principle [27] originated from information theory [20]. In a nutshell, assume all sequences are encoded in bits using some optimal coding scheme. If a recurring substring represents important information for the sequences in a set, we are able to save a significant amount of bits if we encode the sequences with the aid of that substring. The amount of bits that we save indicates the informativeness of the substring. We used this heuristic to select the most informative recurring substrings, i.e., substrings that contained the most information, to differentiate positive sequences from negative sequences. In general, this heuristic favors longer substrings that occur more frequently, with less frequent letters, in the positive sequences [27].

Family number	Family name
1	U6
2	yybP-ykoY
3	S_box
4	SRP_bact
5	Retroviral_psi
6	sno_14q_I_II
7	ctRNA_pND324
8	Entero_5_CRE
9	K_chan_RES

Table 1: RNA families used in multi-class classification.

After calculating the global and local features of an adjusted RNA sequence as discussed in Section 4, the sequence can be represented by a vector in a Euclidean space R^k where k equals the total number of bi-grams, namely 144, plus the number of recurring substrings found in the positive training data set. We use the Gaussian RBF kernel to train our SVM classifier. If x and y are two feature vectors, the kernel is

$$K(x, y) = e^{(-|x-y|^2/2\sigma)} \tag{24}$$

Here σ is a parameter that can be tuned. Basically, choosing a very small σ results in "over-fitting". In other words, the SVM classifier can fit the training data very well, but its generalization ability suffers. It cannot correctly predict labels for untrained data in general. A good indication of whether "over-fitting" occurs is when there are a large number of support vectors. Choosing a very large σ renders the SVM classifier incapable of fitting the training data accurately. Therefore, in choosing the σ value there is a trade-off between accuracy and generalization. In our experiments, for the data on which we tested our SVM classifier, σ was in the range $0.15 - 0.5$. We used Platt's sequential minimal optimization approach to implement the SVM classifier [19].

5.2 Results

Table 2 shows the result of our experiments. In Table 2 we evaluate the performance of our SVM classifier using the sensitivity and specificity measures originally developed in [2]. For each family, the sensitivity (Q^D) represents the percentage of test sequences correctly classified relative to the total number of test sequences in that family; the specificity (Q^M) represents the percentage of test sequences correctly predicted relative to the total number of test sequences predicted to be in that family. Formally, let x_{ij} be the entry at row i and column j in Table 2; x_{ij} is the number of sequences of family i predicted to be in family j. The sensitivity for family i is

$$Q^D = \frac{x_{ii}}{\sum_{k=1}^{9} x_{ik}}.$$

The specificity for family i is

$$Q^M = \frac{x_{ii}}{\sum_{k=1}^{9} x_{ki}}.$$

Table 3 compares the proposed approach that uses both bi-gram occurrence frequencies and recurring substrings with the approach using only bi-gram occurrence frequencies as suggested in

Family	1	2	3	4	5	6	7	8	9	Q^D
1	27	0	0	0	0	0	0	0	0	1.00
2	0	36	0	1	0	0	0	0	0	0.97
3	0	0	36	0	0	0	0	0	0	1.00
4	1	0	0	33	1	0	0	0	0	0.94
5	0	0	0	0	84	0	0	0	0	1.00
6	0	0	0	0	0	29	0	0	1	0.97
7	0	0	0	0	0	0	24	0	0	1.00
8	0	0	0	0	0	0	0	30	0	1.00
9	0	0	0	0	0	0	0	1	42	0.98
Q^M	0.96	1.00	1.00	0.97	0.99	1.00	1.00	0.97	0.98	

Table 2: Performance evaluation of the proposed SVM classifier on nine RNA families.

Family	Kernel of bi-gram count and recurring substring		Kernel of bi-gram count only		Kernel based on labeled dual graph	
	sensitivity	specificity	sensitivity	specificity	sensitivity	specificity
tmRNA	61.3	66.4	73.9	67.4	74.5	65.5
U6	90.0	98.3	94.0	92.0	82.1	72.4
RRE	100.0	100.0	100.0	100.0	76.9	80.0
THI	88.1	96.1	78.9	83.0	84.5	78.5
S_box	95.7	100.0	90.7	92.8	85.9	88.2
SRP_bact	91.4	98.8	87.1	87.8	88.4	89.3

Table 3: Comparison of the three studied kernels.

[24], as well as the labeled dual graph (LDG) kernel described in [13]. The comparison of these three approaches was done on six families, namely tmRNA, U6, RRE, THI, S_box and SRP_bact taken from the Rfam database. For each of these families, positive sequences were sequences in the family; negative sequences were obtained from randomly shuffling the nucleotides of the sequences while preserving the di-nucleotide frequencies in them, as done in [13]. For each family, the performance of a classifier was evaluated using 10-fold cross-validation. The results for the LDG kernel in the table are taken from [13].

Table 3 shows that for all the tested families except tmRNA, our kernel outperforms the other two approaches. One possible reason that the performance of our kernel deteriorates for family tmRNA is that this family may have no recurring substrings that are statistically significant. Under this circumstance, the use of recurring substrings does not improve the performance of the kernel. Rather, it may even weaken the kernel's performance because of including statistically insignificant information.

6 Conclusion

In this paper we presented several different approaches that use support vector machines for RNA classification. We proposed a new kernel that uses a bi-gram model to calculate global features, and uses recurring substrings to calculate local features of RNA molecules. This kernel considers both the primary structure and the secondary structure of an RNA molecule. Experimental results showed that the SVMs based on this new kernel perform well, and are better than existing methods in many cases.

The accuracy of our SVM classifier depends on the accuracy of the algorithm used to predict RNA secondary structures. We believe the accuracy of our classifier can be further improved if a more accurate folding algorithm is employed. One interesting direction of future research is to modify our approach so that it does not need to use any folding algorithm. This is reminiscent of the approach developed by Tsuda *et al.* [24], which can handle the situation where the secondary structure of an RNA sequence is unknown. In this case, they used a probabilistic model, namely stochastic context free grammar, to compute the kernel. We may change our kernel so that it is based on a similar probabilistic model. The challenge is to figure out a way to exploit local structural information in the kernel when the secondary structure of RNA is only implied in the probabilistic model. Under this circumstance, the technique for extracting recurring substrings described in Section 4.2 needs to be modified or new techniques need to be developed.

Acknowledgment

We would like to thank Taishin Kin, Koji Tsuda, Kiyoshi Asai, Yan Karklin, Richard Meraz and Stephen Holbrook for useful discussions and for permissions to present their work in [13, 24].

References

[1] Bailey, T. L. and Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pp. 28-36.

[2] Baldi, P., Brunak, S., Chauvin, Y., Andersen, C. A. and Nielsen, H. (2000) Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, **16(5)**, 412-424.

[3] Boser, B. E., Guyon, I. and Vapnik, V. N. (1992) A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144-152.

[4] Brazma, A., Jonassen, I., Ukkonen, E. and Vilo, J. (1996) Discovering patterns and subfamilies in biosequences. *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pp. 34-43.

[5] Burges, C. J. C. (1998) A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2(2)**, 955-974.

[6] Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1998) *Biological Sequence Analysis.* Cambridge University Press, England.

[7] Gan, H. H., Pasquali, S. and Schlick, T. (2003) Exploring the repertoire of RNA secondary motifs using graph theory with implications for RNA design. *Nucleic Acids Res.*, **31(11)**, 2926-2943.

[8] Gartner, T. (2003) A survey of kernels for structured data. *ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations*, **5(1)**, 49-58.

[9] Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A. and Eddy, S. R. (2003) Rfam: an RNA family database. *Nucleic Acids Res.*, **31(1)**, 439-441.

[10] Hart, R., Royyuru, A. K., Stolovitzky, G. and Califano, A. (2000) Systematic and automated discovery of patterns in PROSITE families. *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, pp. 147-154.

[11] Hui, L. C. K. (1992) Color set size problem with application to string matching. In Apostolico, A., Crochemore, M., Galil, Z. and Manber, U. (eds) *Combinatorial Pattern Matching. Lecture Notes in Computer Science*, **644**, Springer-Verlag, pp. 230-243.

[12] Jonassen, I., Collins, J. F. and Higgins, D. G. (1995) Finding flexible patterns in unaligned protein sequences. *Protein Science*, **4**, 1587-1595.

[13] Karklin, Y., Meraz, R. F. and Holbrook, S. R. (2005) Classification of non-coding RNA using graph representations of secondary structure. *Proceedings of the Pacific Symposium on Biocomputing*, pp. 5-16.

[14] Kashima, H., Tsuda, K. and Inokuchi, A. (2003) Marginalized kernels between labeled graphs. *Proceedings of the International Conference on Machine Learning*, pp. 321-328.

[15] Liu, J., Wang, J. T. L., Hu, J. and Tian, B. (2005) A method for aligning RNA secondary structures and its application to RNA motif detection. *BMC Bioinformatics*, **6(89)**.

[16] Ma, B., Wang, L. and Zhang, K. (2002) Computing similarity between RNA structures. *Theoretical Computer Science*, **276**, 111-132.

[17] Ma, Q., Wang, J. T. L., Shasha, D. and Wu, C. H. (2001) DNA sequence classification via an expectation maximization algorithm and neural networks: a case study. *IEEE Transactions on Systems, Man, and Cybernetics*, **31(4)**, 468-475.

[18] Noble, W. S. (2004) Support vector machine applications in computational biology. In Schoelkopf, B., Tsuda, K. and Vert, J.-P. (eds) *Kernel Methods in Computational Biology*, MIT Press, pp. 71-92.

[19] Platt, J. (1998) Sequential minimal optimization: a fast algorithm for training support vector machines. Technical Report 98-14, Microsoft Research, Redmond, Washington.

[20] Shannon, C. E. (1948) A mathematical theory of communication. *Bell System Technical Journal*, **27**, 379-423, 623-656.

[21] Shapiro, B. A., Kasprzak, W., Wu, J. C. and Currey, K. (1999) RNA structure analysis: a multifaceted approach. In Wang, J. T. L., Shapiro, B. A. and Shasha, D. (eds) *Pattern Discovery in Biomolecular Data: Tools, Techniques and Applications*, Oxford University Press, pp. 183-215.

[22] Shapiro, B. A., Bengali, D., Kasprzak, W. and Wu, J. C. (2003) Exploring RNA intermediate conformations with the massively parallel genetic algorithm. In Wang, J. T. L., Wu, C. H. and Wang, P. P. (eds) *Computational Biology and Genome Informatics*, World Scientific, pp. 1-33.

[23] Smith, T. F. and Waterman, M. S. (1981) Comparison of biosequences. *Advances in Applied Mathematics*, **2**, 482-489.

[24] Tsuda, K., Kin, T. and Asai, K. (2002) Marginalized kernels for biological sequences. *Proceedings of the Tenth International Conference on Intelligent Systems for Molecular Biology*, pp. 268-275.

[25] Vapnik, V. (1995) *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

[26] Wang, J. T. L., Ma, Q., Shasha, D. and Wu, C. (2000) Application of neural networks to biological data mining: a case study in protein sequence classification. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 305-309.

[27] Wang, J. T. L., Ma, Q., Shasha, D. and Wu, C. (2001) New techniques for extracting features from protein sequences. *IBM Systems Journal*, Special Issue on Deep Computing for the Life Sciences, **40(2)**, 426-441.

[28] Wang, J. T. L., Marr, T., Shasha, D., Shapiro, B. A. and Chirn, G. (1994) Discovering active motifs in sets of related protein sequences and using them for classification. *Nucleic Acids Res.*, **22(14)**, 2769-2775.

[29] Wang, J. T. L., Marr, T., Shasha, D., Shapiro, B. A., Chirn, G and Lee, T. Y. (1996) Complementary classification approaches for protein sequences. *Protein Engineering*, **9(5)**, 381-386.

[30] Wang, J. T. L., Rozen, S., Shapiro, B. A., Shasha, D., Wang, Z. and Yin, M. (1999) New techniques for DNA sequence classification. *Journal of Computational Biology*, **6(2)**, 209-218.

[31] Wang, J. T. L., Shapiro, B. A. and Shasha, D. (eds) (1999) *Pattern Discovery in Biomolecular Data: Tools, Techniques and Applications*. Oxford University Press, New York.

[32] Wang, J. T. L., Wu, C. H. and Wang, P. P. (eds) (2003) *Computational Biology and Genome Informatics*. World Scientific, Singapore.

[33] Wang, J. T. L., Zaki, M. J., Toivonen, H. T. T. and Shasha, D. (eds) (2005) *Data Mining in Bioinformatics*. Springer, London/New York.

[34] Wuchty, S., Fontana, W., Hofacker, I. L. and Schuster, P. (1999) Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, **49(2)**, 145-165.

[35] Yin, M. M. and Wang, J. T. L. (2001) Effective hidden Markov models for detecting splicing junction sites in DNA sequences. *Information Sciences*, **139**, 139-163.

[36] Yin, M. M. and Wang, J. T. L. (2004) GeneScout: a data mining system for predicting vertebrate genes in genomic DNA sequences. *Information Sciences*, **163**, 201-218.

[37] Zhang, K. (2005) RNA structure comparison and alignment. In Wang, J. T. L., Zaki, M. J., Toivonen, H. T. T. and Shasha, D. (eds) *Data Mining in Bioinformatics*, Springer, pp. 59-81.