ABSTRACT

CLASSIFYING RNA SECONDARY STRUCTURES USING SUPPORT VECTOR MACHINES

by
PrathyUsha Sunkara

In contrast to DNA, RNA prevails as a single strand. As a consequence of small self-complementary regions, RNA commonly exhibits an intricate secondary structure, consisting of relatively short, double helical segments alternated with single stranded regions. The amount of sequence data available is rising rapidly day by day. One of the problems encountered on a specific molecule is finding the relevant data between the massive number of other sequences to be done by reading lists with a short description of all new entries in large databases already existing. One of the main objectives of this work is to take the extracted structures of aligned ribosomal RNA sequences and their secondary structures and cluster them. The proposal is to apply existing dimensionality reduction algorithms to these extracted structures and then cluster them in a reduced dimensional space using Support Vector Machines.

CLASSIFYING RNA SECONDARY STRUCTURES USING SUPPORT VECTOR MACHINES

by
PrathyUsha Sunkara

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

January 2006

APPROVAL PAGE

CLASSIFYING RNA SECONDARY STRUCTURES USING SUPPORT VECTOR MACHINES

PrathyUsha Sunkara

Dr. Jason T. Wang, Dissertation Advisor Professor of Computer Science, NJIT	Date
Dr. Chengjun Liu, Committee Member Assistant Professor of Computer Science, NJIT	Date
Dr. Qun Ma, Committee Member Assistant Professor of Computer Science, NJIT	Date

BIOGRAPHICAL SKETCH

Author:	PrathyUsha Sunkara
Degree:	Master of Science
Date:	January 2006

Education:

Master of Science in Computer Science
New Jersey Institute of Technology, Newark, NJ, 2006

Bachelor of Engineering in Computer Science
R.M.K. Engineering College, Chennai, India, 2004

Major: Computer Science

The secret of health for both mind and body is not to mourn for the past, worry about the future, or anticipate troubles but to live in the present moment wisely and earnestly.
- Buddha

To my dearest dad, mom and brother…

ACKNOWLEDGMENT

I would like to thank all the people who've been a part of this work, my family, teachers and friends. This would not have been possible without the motivation and mentoring provided by Dr. Jason Wang, the best advisor and mentor I've had. Thank you very much professor for all the encouragement, support and advice at every step of these graduate years.

I would like to thank all the professors for their continuous encouragement and thought-provoking discussions which helped me a lot to improve this work, Dr. Marc Ma and Dr. Chengjun Liu for taking the time to review the thesis and offering me their valuable suggestions. I would like to thank Dr. Barry Cohen for some of his great advice which helped me at every step to face many challenges and reach my goals.

I would like to thank my brother, Dr. Gangadhar Sunkara for giving me tremendous encouragement, moral support and inspiration all along and moulding me into what I am now. I would like to thank my best friends Shirin, Abhishek and Girish for listening and supporting me constantly all throughout these graduate years through my frustrations and walking with me all along to make this a success. I would like to especially thank my father for encouraging me to strive for excellence and all his great advice to pursue what matters the most to me.

Finally, I would like to thank Dr. Kane and Ms. Gonzalez for their patience and valuable suggestions through all the reviews and I would like to thank NJIT for providing me this golden chance to do my masters.

TABLE OF CONTENTS

Chapter Page

1 INTRODUCTION..1

2 RNA SECONDARY STRUCTURES AND RSMATCH....................................8

2.1 Basic Concepts..9

2.2 Feature Analysis..10

2.3 Background..12

2.4 RSMatch..17

3 SUPERVISED LEARNING..21

3.1 Cluster Analysis...22

3.2 Machine Learning...23

4 SUPPORT VECTOR MACHINES...28

4.1 Theory and Principles of SVM..28

4.2 Data Generation...33

4.3 Kernels...35

4.4 Kernel Design for RNA Structures...37

5 IMPLEMENTATION..41

5.1 Theory..41

5.2 Using the software...43

6 CONCLUSION..50

APPENDIX..52

REFERENCES..57

LIST OF FIGURES

Figure	Page
2.1 Secondary Structures of RNA molecule	9
2.2 Different ways of RNA folding	18
3.1 Data Mining Methods	22
3.2 Clustering	23
3.3 Supervised and Unsupervised Learning	24
4.1 SVM Methodology	30
4.1 Over-fitting and Better Classifiers	31
4.2 Feature Projection	34
4.3 Simple VC dimension example	35
4.4 Counting scheme for a CFG matrix	39
5.1 True negatives, True positives, False negatives, False positives	49
6.1 Classification model	50
A1 Hepatitis features	52
A2 Data file	53
A3 Data file	54
A4 Training data	54
A5 Validation data	55
A6 Scaling data	55
A7 Classification based on result file	56

CHAPTER 1

INTRODUCTION

Just as photocopy sections of the instruction manual would be handled out to the different workers, rather than tearing sections out of the book for distribution, so individual genes are not themselves dispatched to the cytoplasm. Instead, copies of the instructions contained in genes are sent out. These copies are in the form of molecules of RNA, which is similar to DNA but has just one strand of bases instead of two and the base uracil (U) instead of thymine (T) – this is very important for building or making new proteins.

The exciting wider implication: if ribosomes can be induced to translate RNAs of any nucleotide sequence into protein, then RNAs of known nucleotide sequence could be incubated with ribosomes and watched to see what kind of amino acid sequence came out. This is the solution of the genetic code!

It is possible to use a computer to analyze RNA molecules, because it is much more predictable than DNA or proteins and lends itself well to computer analysis. RNA is incredible because it can do almost anything we can think of, including transporting and transmitting genetic information – just like DNA does – and performing catalytic functions, just like proteins do. The direct outcome of these developments is huge databases of sequence and physiological data. Associated with these databases are a number of specific questions such as homology determination between two or more sequences or structures, elucidation of sequence – structure – function relationships, functional motifs, etc.

The amount of sequence data available is rising rapidly day by day. One of the problems encountered on a specific molecule is finding the relevant data between the massive number of other sequences to be done by reading lists with a short description of all new entries in large databases already existing. The databases contain a number of sequences with or without any accession numbers – the aligned pair to their publication. For some cases, an update is needed, or the name of the sequence may already exist in the database, sometimes, the original database should be changed in the update process, change names in references, formats, and alignments. So, all these secondary structures indicated in the databases are based on comparative analysis. But there are several problems with this approach. Although clear cases of covariance portions of independent compensations and no counter-evidence, some possible duplexes show corresponding base-pairs as well as non-compensated characteristics of species or groups.

One of the overall objectives of this work is to provide a broad framework of principles for integrating individual information along with state – of – the – art tools for addressing specific homology classification by taking the extracted structures of aligned ribosomal RNA sequences and their secondary structures and group them into different organized homology collections.

Classification using clustering algorithms - a non-linear activity that generates ideas, images and feelings around a stimulus word and enable us to see patterns for different ideas. Biological data are complex, they are arguably the most complex data known. Apart from mere numerical richness, there are many similar but not identical entities that are challenging to model. Often incomplete biological objects are large and complete descriptions are time-taking to obtain because of the limited resources or because attempts to collect data fail. Many biological concepts are incompletely understood due to changing models and data, and the model of a given concept is likely to become a more sophisticated one over time. The biological domain is highly integrated where the information is collected in hierarchies with many facts having to bear the truth of other facts, having to alter based on the results of the applied semantic rules.

Clustering is used to discover sets of unstructured text documents and summarize them into labeled collections (Fig 1.1). It is used to organize and compactly summarize, disambiguate and navigate the results retrieved from large databases. Increasing number of molecular biological data collections and using them together requires interoperability since it is not practical to dump and organize a single monolithic model.

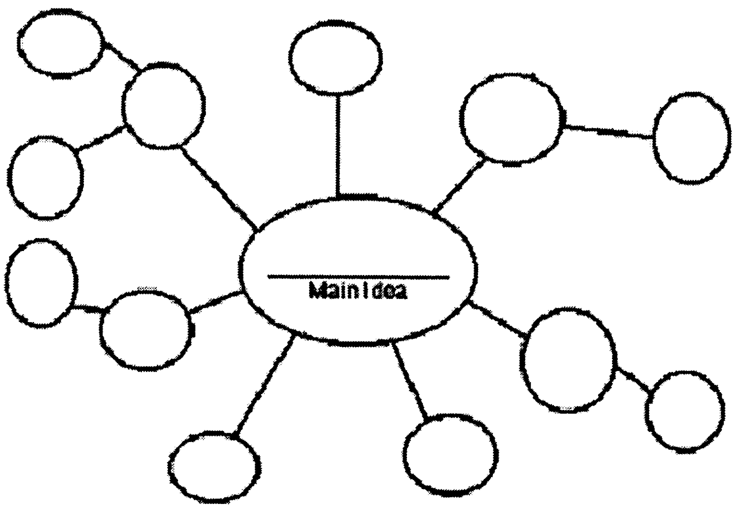


Figure 1.1 Main Idea is sub-divided and classified into different groups or clusters.

It is useful for personalized information delivery by providing a setup for routing new information such as that arriving from news-feed and new scientific publications. Traditional clustering algorithms such as the hierarchical agglomerative clustering and graph-theoretic methods have been explored in text mining literature. But now, this proposal is to apply vector based classifying methods to cluster the extracted sequences of RNA secondary structures and classify them into different homologies. The application of existing dimensionality reduction algorithms to these extracted structures and then clustering them in a reduced dimensional space may introduce a small inaccuracy in performing classification.

Evaluating a series of the extracted structures using the state of the art developed software RSMatch (Liu, J. et al.) for selecting features and its variants for the effectiveness of the proposed approach and training the Support Vector Machines (Vladimir Vapnik) for classification of different secondary structures according to their different topological and functional characteristics and the experimental goal is to enhance the accuracy of clustering of RNA secondary structures.

The RSMatch package is implemented in Java and Perl and run under a UNIX/Linux operating system. It takes an input data as RNA sequences (which are in FASTA format), which invokes automatically the Vienna RNA package to fold the sequences into structures and then align the secondary structures. The use of both the pairwise structure alignment or multiple structure alignment, using RSMatch and using Count Kernels for RNAs (Taishin Kin, et al.), as kernel function to train the Support Vector machines for classifying them into well-characterized families are studied here.

Classification of RNA secondary structures using Support Vector Machines (SVMs) need a representation of the secondary structure and also a kernel function that provides a reasonable similarity measure for the chosen representation. There are many other methods to do this classification, using clustering algorithms such as UPGMA, but choosing to train SVMs is very important because they are well-suited to deal with learning tasks, where the number of attributes is large with respect to the number of training examples. The hidden Markov models have been a great work for classification at the superfamily levels, but SVMs are very promising methods for classifications at the subfamily level.

Classification using SVMs is a very interesting problem, because usually they have a very small set of positive training examples and often have an extremely large number of negative example set. This is a big problem, which can be easily solved nowadays, by training Support Vector Machines using the related kernel functions.

RNA structures are classified according to different homology groups, such as functional classification, structural classification, according to RNA motifs, can be classified according to RNA Tertiary interactions. Classification according to their secondary structures is a very important problem because it is very common to find many examples of homologous RNAs that have a common secondary structure without sharing significant sequence similarity. A small change in these structures, correspond to a particular disease, so classifying and comparing the different structural dissimilarities can lead us to diagnose many disease states.

The chemical and biological properties of many RNAs are determined by their secondary structures. Nowadays, secondary structures of RNA have many applications in functional classification, where many RNA viruses have been classified according to their secondary structures instead of their sequences. Evolutionary studies need secondary structures of ribosomal RNAs of different species to place them on the evolutionary tree. Pseudogene detection needs secondary structures to detect differences between genes and pseudogenes of a DNA sequence which is homologous to a tRNA gene. All these applications need the secondary structures well-classified and well-characterized into different groups, and easier to retrieve all the information for any kind of analysis.

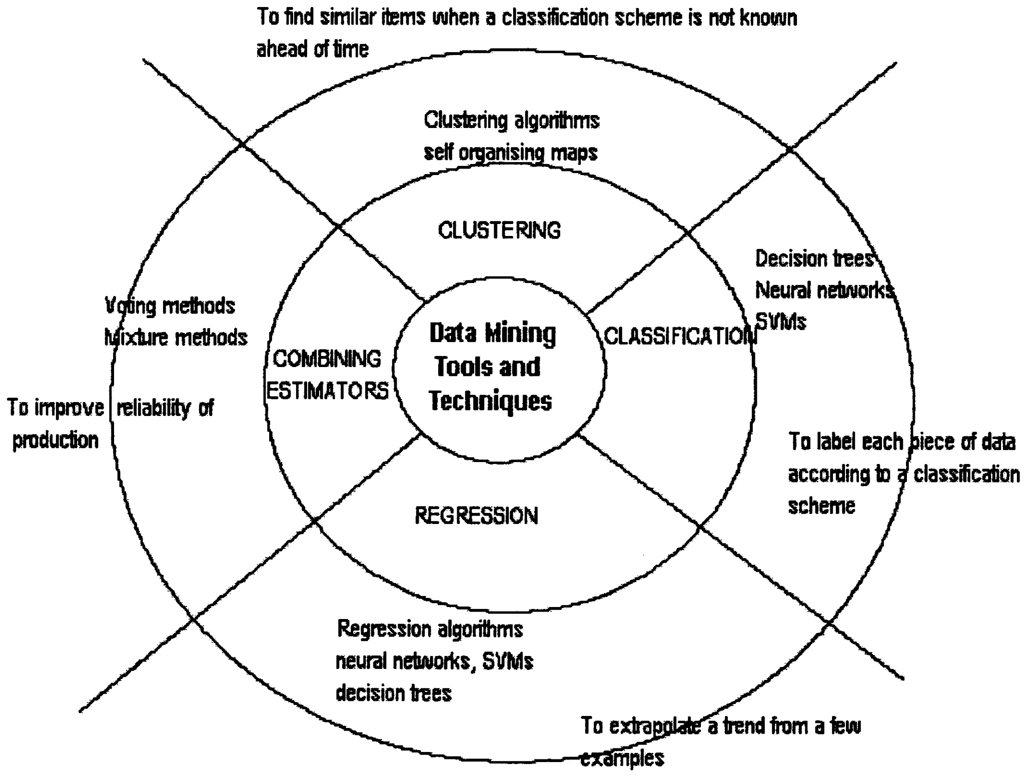

To find similar items when a classification scheme is not known ahead of time
Clustering algorithms self organising maps
CLUSTERING
Decision trees
Neural networks
SVMs
Voting methods
Mixture methods
COMBINING ESTIMATORS
Data Mining Tools and Techniques
CLASSIFICATION
To improve reliability of production
To label each piece of data according to a classification scheme
REGRESSION
Regression algorithms neural networks, SVMs decision trees
To extrapolate a trend from a few examples

Figure 1.2 Data Mining tools and techniques

This is an experimental analysis to study combination of some state – of – the - art programs to extract feature vectors from RNA secondary structure alignments taking the right subset of relevant features using different approaches and training these feature vectors into Support Vector Machines using kernel functions and compare our results with the manually-curated families in the RNA databases. The SVM classifier performance is assessed with values of sensitivity and specificity by computing area under ROC (Receiver Operating Characteristic Curve), which is a general measure of the discrimination ability.

CHAPTER 2

RNA SECONDARY STRUCTURES AND RSMATCH

2.1 Basic Concepts

RNA consists of a set of nucleotides that can be either adenine (A), cytosine (C), guanine (G), or uracil (U). Each of these nucleotides is known as the base and can be bonded with another one via hydrogen bonds. A base-pair is formed when two bases bond with each other. There are two types of base-pairs: canonical base-pair and wobble base-pair.

Canonical base-pair is formed by a double hydrogen bond between A and U, or a triple hydrogen bond between G and C. Wobble base-pair is formed by a single hydrogen bond between G and U. Apart from these two types of base-pairs, other base pairs like U-C and G-A are also feasible, but they are seen very rare. Contrast to DNA, RNA is single stranded. Due to the extra hydrogen bond in each RNA base, RNA bases in a RNA molecule hybridize with itself to form complex structures such as secondary and tertiary structures. The primary structure of an RNA molecule is just its sequence of nucleotides, whereas secondary structure specifies it's canonical and wobble base-pairs that occur in the molecule(Fig 2.1). Traditionally, the role of RNA in the cell was mainly in the context to protein gene expression, limiting RNA to its function as mRNA, tRNA, and rRNA. The discovery of a diverse array of transcripts which are not translated into proteins but are functioning as RNAs has changed this view profoundly.

The base sequence is essential in determining the secondary structure, and two secondary structures are distinct if the two sets of base-pairs are not equal. The structural components of secondary structure are base-pairs, bulges, interior loops, end loops, and multi-branch loops (Fig 2.1). A hairpin is a structure with base-pairs (at least one), bulges, interior loops, and exactly one end-loop. Multi-branch loops are junctions, such as in the cloverleaf, where more than one hairpin or more complex secondary structures are appended.

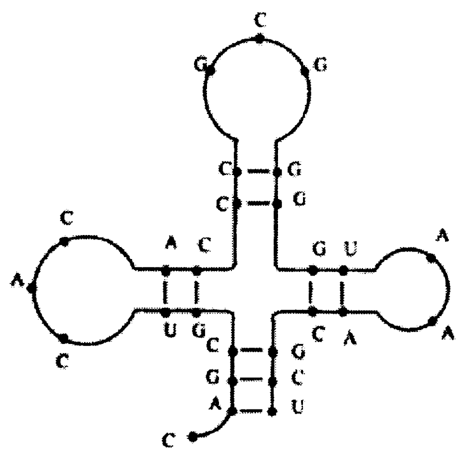


Figure 2.1 Secondary structure of an RNA molecule with bulges, loops and base-pairs.

The central dogma of molecular biology posits that in living cells, the genetic information is translated to proteins, which do the real work. The traditional view of RNA is as a helper molecule in the translation process. But this view has changed recently as RNA has a very important role in regulation of genes and as a catalyst in many cellular processes. The structure of RNA molecules is the key to their function. And classifying according to their structural information is of great value for various analysis and research.

During hybridization, pairs of bases in RNA form hydrogen bonds, with the complementary pairs and others. A set of bonds largely held together results in a folded molecule, which is its secondary structure. The knowledge of this secondary structure of a folded RNA molecule sheds valuable insight on its function.

Almost all RNA molecules form secondary structures. The presence of secondary structure in itself therefore does not indicate any functional significance. A small number of point mutations are very likely to cause large changes in the secondary structures. A very small difference in the nucleic acid sequence can lead to obtaining almost unrelated structures if the mutated sequence positions are chosen randomly. Selection acts to preserve a structural element which has some function. From the RNA secondary structures from a small sample of related RNA sequences, the function of conserved structures cannot be predicted but knowledge of their location can be used to follow up our specific queries.

2.2 Feature Analysis

RNA sequences have a specific base sequence, over the {A, U, G, C} alphabet. The base-pairs A-U and G-C are called Watson-Crick base-pairs. Folding RNA sequences into secondary structures is frequently used as a tool for classification and prediction of the function of RNA molecules. In many structural RNA molecules, G-U base-pairs also appear. All the components make some contribution to the stability of the overall structure. This stability is measured in free-energy, but many studies show that MFE-predictions cannot be used for detection of functional RNAs, as thermodynamic stability alone is not significant. It can be a valuable diagnostic feature since functional RNAs are more stable than random sequences to some degree.

Secondary structures decompose into the following structural elements which are assumed to contribute additively to MFE:

Stacks are double-helical regions of RNA, Loops are unpaired regions enclosed by stacks. The degree of a loop is the number of stacks attached to it. A hair-pin loop thus has degree 1, bulges and interior loops have degree 2, and all loops with degree larger than 2 are denoted as multiloops.

External elements are strains of unpaired bases that are not part of a loop. They are either free ends or joints connecting different components of the secondary structure.

Free energy of folding

The energy of a particular structure is assumed to be the sum of contributions from individual base pair stackings and loops strains. Three different Folding algorithms can be used for RNA secondary structure formation.

Maximum matching – algorithm (Nussinov et al. 1978) derives a structure with a maximum number of base pairs irrespectively of energy parameters.

Kinetically controlled folding – form structures that are determined by the kinetics of the folding process (Martinez 1984; Abrahams et al. 1990; Gultyaev 1991).

Minimum free energy – most common approach to predict RNA structures based on minimization of free energy (Tinoco et al. 1971) where loop matching is a special case of RNA mfe-folding.

The thermodynamic model for RNA structure formation posits that, out of the many optimal possibilities, an RNA molecule folds into that structure which has the minimum free energy (MFE). Free energy models typically assume that the total free energy of a given secondary structure for a molecule is the sum of independent contributions of adjacent, or stacked, base-pairs in stems (which tend to stabilize the structure) and of loops (which tend to destabilize the structure). The functions of length and base composition of RNA molecules can be computed very accurately from a relatively simple regression model which can be obtained by a standard implementation of a support vector machine (SVM) regression algorithm (Thorsten Joachims). The time-consuming sampling procedures to get these results can be replaced by the SVM estimate without a significant loss of accuracy, while saving a large factor of CPU time.

A comprehensive understanding of cellular processes is impossible without considering RNAs as key players. Identifying efficiently the functional RNAs (nc RNAs as well as cis-acting elements) in genomic sequences is one of the major goals in current bioinformatics. About half of the transcripts from Human chromosomes 21 and 22 are non-coding. Structured RNA motifs furthermore function as cis-acting regulatory elements within protein coding genes (Fig 2.2). Functional RNAs lack in their primary sequence common statistical signals that could be exploited for reliable detection algorithms. Many functional RNAs however depend on their defined secondary structural and in particular, evolutionary conservation of secondary structures serves as compelling evidence for biologically relevant RNA function. Therefore, comparative analysis seems to be the most promising approach. To date, many collections of complete genomic sequences are identified only sporadically, because of the simple reason, that there are no reliable tools available to scan multiple sequence alignments for functional RNAs. So, here we use the state of art software developed – RSMatch(Liu, J. et al.), a method to align RNA secondary structures and its application to RNA motif detection.

2.3 Background

This thesis is extended on studying some of these methods based on two recent papers related to classification of RNA secondary structures, which use methods different from the proposed traditional methods. One of them is "Classification of RNA Secondary Structures Using the Techniques of Cluster Analysis" (Nakaya, et al.) in which the authors have developed some very interesting methods to classify many suboptimal structures which are very similar and some entirely different of an RNA sequence.

Results are classified and presented analyzing the secondary structures of cadang-cadang coconut viroid, potato spindle tuber viroid and polio virus as examples. Then they applied a similarity metric to classify the secondary structures derived from a set of mutant RNA sequences. The discussion through this paper strongly advocates that the mutation of a given RNA sequence changes not only the optimal secondary structure, but also the population of the cluster of the secondary structures.

"Classification of Non-Coding RNA using Graph Representations of Secondary Structure" (Yan Karklin, et al.) is another paper, where the authors have developed a labeled dual graph representation of RNA secondary structure by adding biologically meaningful labels to the dual graphs proposed by Gan et al. Then, a similarity measure is defined directly on the labeled dual graphs using the recently developed marginalized kernels.

Using this similarity measure, the Support Vector Machine classifiers are trained to distinguish between RNAs of known families from random RNAs with similar statistics. In an RNA molecule, hydrogen bonds are formed between the nitrogen bases (G-C, A-U and G-U) to form stable base-pairs (Fig 2.2). The hydrogen bonds which are formed continuously in an RNA molecule are called stacking regions, and they greatly contribute to the stability of the structures of the RNA molecule. The secondary structure is determined by the stacking regions, which contains many aspects of biological information. All of the stacking regions cannot co-exist in a single feasible secondary structure due to their structural restrictions.

The secondary structures are obtained by selecting the stacking regions which can co-exist with each other. In the first paper, the authors (Nakaya et al.) have studied to enumerate all the feasible secondary structures using a search tree in a systematic manner. A tree is constructed, ordering the stacking region candidates according to their free energy. Feasible secondary structures are obtained from the search tree, whose nodes at depth i determine whether i th candidates are selected or not. Actually the searching can be done using massive parallel processors and efficient pruning and migration (Nakaya et al.). The constructed search tree gives two feasible structures at its leaf node and represented with a list of candidate numbers which participate in it. The classification is done by using clustering algorithm – UPGMA, "unweighted pair group methods using arithmetic average" (Hubert, 1974; Edelbrook & McLaughlin, 1980; Milligan & Issac, 1980; Milligan et al.; Milligan et al.).

Starting with the clusters containing a single secondary structure, mergers of clusters are iterated, and the pair of two existing clusters which are nearest is selected and put together into a new cluster whose members belong to the original clusters. This process is continued until only a single secondary structure cluster, which contains all the secondary structures, remains. Here a metric is introduced among clusters to select the clusters which are to be merged, by computing the hamming distance between two secondary structures. But in this study, the proposal is to use RSMatch (Liu, J. et al.), the state of art software for multiple structure alignment and iterative database search.

Using RSMatch to extract features from the RNA secondary structures according to the scores obtained, and these score matrices are used to train the Support Vector Machine classifiers. These are studied for better performance and time – efficiency than the clustering algorithms, such as UPGMA. These clustering algorithms cannot be concluded as not being efficient, but using the state of art programs and combining with the latest techniques of SVM classifiers, the intention is to propose more exciting and efficient methods to classify RNA secondary structures into different subfamilies and compare them against the already classified, manually-curated families in the open web RNA databases.

RNA secondary structures, the pattern of base-pairing contain the critical information for determining the three dimensional structure and function of the molecule (Karklin et al.). A labeled dual graph representation of RNA secondary structure is developed by adding biologically meaningful labels to the dual graphs proposed by Gan et al. A similarity measure is defined on the labeled dual graphs using the marginalized kernels to train the Support Vector Machine classifiers to distinguish RNAs of known families from random RNAs with similar statistics.

Non-coding RNAs are defined as those RNAs that do not encode proteins, but instead serve so many other functions in the cell. They play a variety of critical roles and their function is uniquely determined by the three dimensional structure of the molecule. Development of computational tools based on RNA secondary structure is essential for discovery of new RNAs and classifying them according to their structural and functional roles is of high demand now.

To reach the functional form, a single stranded RNA undergoes folding by itself, forming G-C, A-U, and G-U base-pairing and stacking interactions – to form short helices and various single stranded loop regions that define its secondary structures. Classification of RNA secondary structures using Support Vector Machines requires a secondary structure representation and a kernel function that provides a reasonable similarity measure for the chosen representation.

This can be done using SVM classifier, but SVM classifier on graph objects need a kernel function to determine the similarity measure between two labeled dual graphs. The simplest and computationally efficient kernel is the marginalized kernel, which computes the similarity measure between two arbitrary labeled graphs by comparing the label sequences and the highest similarity score for the pair of graphs. Many biophysical parameters can serve as the basis for similarity comparisons – base composition, sequence or structural alignment, feature lengths, etc. The ability of the classifier is tested to learn RNA secondary structure and predict RNA family labels. And then the SVM classifier is trained to distinguish between random structures and the structures from known families in the existing databases.

In order to classify alignments as a "functional RNA" or "other", the separating matrix scores between functional RNAs and other sequences should be defined, and SVM classifier should be trained according to that.

There have been many experiments using CLUSTALW, but CLUSTALW aligns sequences, it can show the sequence similarity, RSMatch (Liu, J et al.) is the state of the art program which compares and aligns the secondary structures of RNA, and the use of RSMatch in this experimental study is to classify secondary structures, because secondary structures define the functionality of an RNA molecule, and classification of different homology classes of secondary structures can help in numerous applications such as diagnosing diseases and exploiting their structural motifs for several reasons.

2.4 RSMatch

It is the state of art program, developed recently by Dr. Wang with other professors at UMDNJ. This has many unique features which other software have not yet developed or are in still at their infancy. RSMatch provides four functions – 1) regular database search, 2) multiple structure alignment, 3) iterative database search and 4) pair-wise structure alignment.

It is implemented in Java and Perl and run under a UNIX/Linux operating system. In this thesis, the features of multiple structure alignment are used which obtain the scores, for different structures and extract feature vectors. Many random structures are taken and using RSMatch, and are aligned to generate scores between them as a similarity metric, to extract features, applying default parameters and training the SVM classifier feeding these features and defining a kernel for RNA sequences from experiments of Tsuda et al.

RSMatch constructs a multiple local alignment for a given set of RNA structures, by progressively expanding the alignment at each stage. It is very useful when small set of RNAs are functionally related by a shared motif and this shared motif can be located by the multiple local alignment function. A motif is structural conservation seen in some local regions.

RSMatch will automatically invoke Vienna RNA v1.4 to fold the input sequences into structures and then align the structures. There are two types of input data. The first type is the nested parenthesized notation representing an RNA secondary structure. For each structure, it has three lines: header line, primary sequence line and structure notation line. The second type is the FASTA format for RNA sequences. For the sequence data, RSMatch1.0 will automatically invoke Vienna RNA v1.4 to fold the sequences into structures and then align the structures.

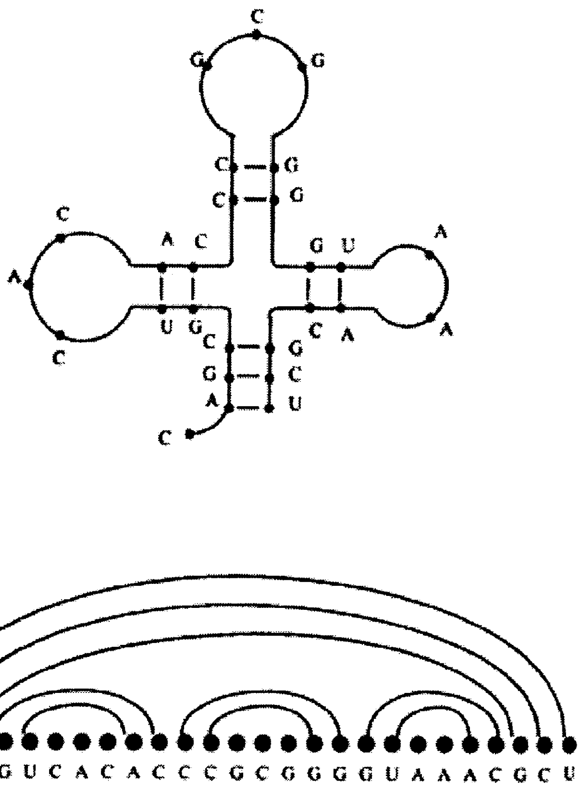


Figure 2.2(a) Different ways of folding into RNA secondary structures.

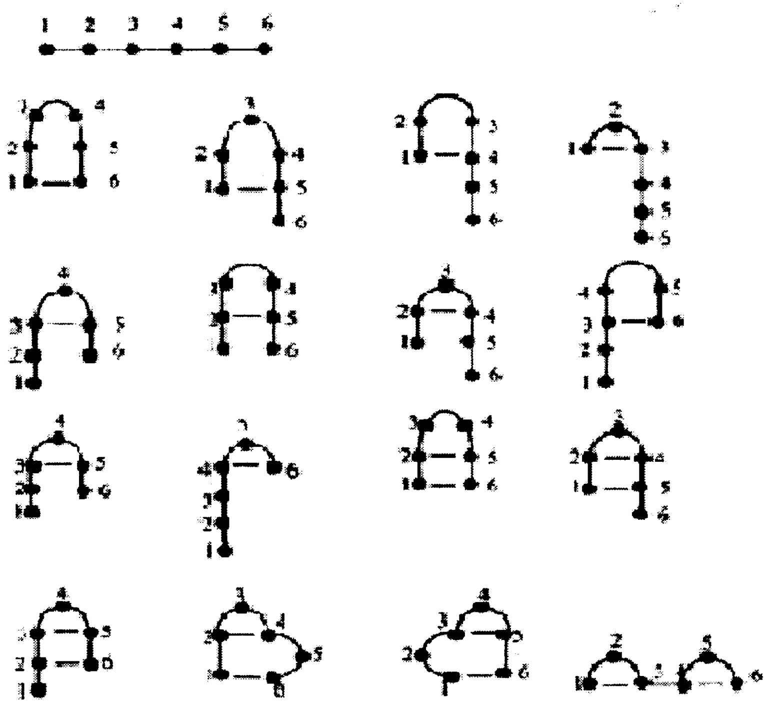


Figure2.2(b) Different ways of folding into RNA secondary structures.

Many options are provided to choose any kind of program to use, to choose between either regular database search or iterative database search, to compute score matrix, choosing between different queries and alignments. The following syntax can be used to do a multiple sequence alignment of different RNA sequences taken randomly from the RNA databases available on the web for open access, which are already manually curated into homology families.

Syntax: RSMatch -p mrsa -d sample1.structDB

will construct multiple structure alignment for the structures in the sample1.structDB (sample1.structDB contains all the random sequences we've chosen from the RNA databases).

RSMatch uses two scoring schemes, i.e. position independent and position dependent schemes. The position independent scheme entails two scoring matrices, one for single-stranded regions and the other for double-stranded regions. This scoring scheme is used in pair-wise comparisons and database searches. The position dependent scheme, also known as profile, scores individual structure positions and is used by the multiple structure alignment and iterative database search functions. RSMatch provides both global and local alignment options even though the latter is more useful in most cases. In addition, RSMatch can take pattern-based structures as input.

Consequently, in this study, the results of RSMatch are presented, taking the RNA sequences randomly from different families in the RNA databases, which invokes automatically the Vienna RNA package and sequences are converted to secondary structures and a multiple sequence alignment for these secondary structures is done to compute scores, from which features can be extracted and applying related kernel functions to these features, train the SVM classifier to classify different RNA secondary structures accordingly and compared against the manually curated structures in the RNA databases can be done.

This is the proposal of this study - combination of new techniques, using the state of art programs to develop new methods to obtain better performance and accurate classification results, which have not been yet developed or proposed.

CHAPTER 3

SUPERVISED LEARNING

Machine learning is the name used in the artificial intelligence community for the larger family of programs that adapt their behavior with experience. Programs are said to 'learn' or be 'trained' but they are always just following well defined sets of instructions. Data mining tools are supplements, rather than substitutes, for human knowledge and intuition. No program is smart enough to take a huge raw data and produce interesting outputs. As the name itself indicates, it refers lightly to finding relevant information or discovering knowledge from a large volume of data. It attempts to discover statistical rules and patterns automatically from data, in a similar way as knowledge discovery found in Artificial Intelligence.

Knowledge-discovery systems may have elements of both models, with the system discovering some rules automatically, and the user guiding the process of rule discovery. Knowledge discovery techniques developed by the AI community, attempt to discover automatically statistical rules and efficient implementation techniques that enable them to be used on extremely large databases. Knowledge discovered from a database can be represented by a set of rules.

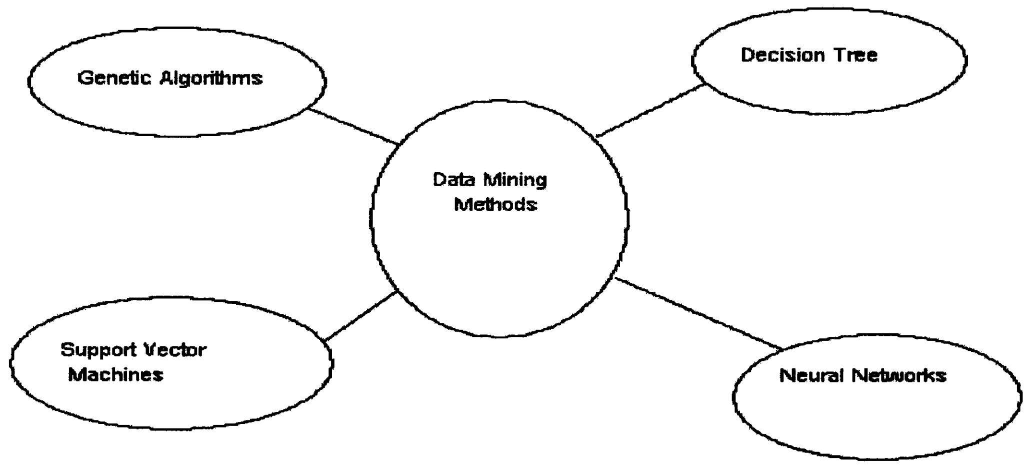


Figure 3.1 Data Mining Methods.

3.1 Cluster Analysis

The Cluster Analysis is a kind of unsupervised learning, which is a procedure of classifying data such that similar data end up in the same class, while dissimilar data doesn't, when the actual classes are not known ahead of time. It is a standard technique for working with various applications of multidimensional data. It is popular for the ability to work on datasets with minimum or no a priori knowledge. This makes clustering practical for real world applications. In high dimensional space, the distance from a record to its nearest neighbor can approach its distance from a record to its nearest neighbor can approach its distance to the farthest record. In the context of clustering, the problem causes the distance between two records of the same cluster to approach the distance between two records of different clusters. Traditional clustering methods may fail to identify the correct clusters. In high dimensional datasets, clusters can be formed in sub-spaces. Only a subset of attributes is relevant to each cluster, and each cluster can have a different set of relevant attributes. An attribute is relevant to a cluster if it helps identify the member records of it. The values at the relevant attributes are distributed around some specific values in the cluster, while the records of other clusters are less likely to have such values. Subspace clustering is to find clusters and their relevant attributes from a dataset, which can be on different sample sets. Identifying the relevant attributes may help improve the clustering accuracy and the selected attributes may also suggest a smaller set of samples to focus on clusters with optimal quality.

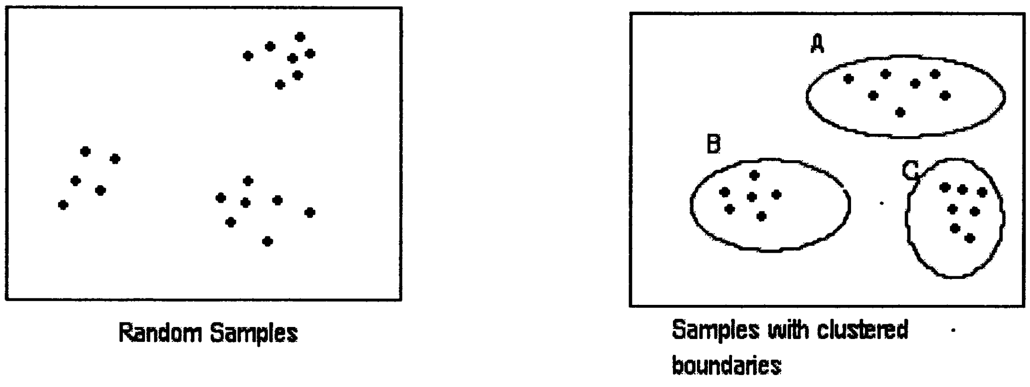


Figure 3.2 Clustering.

3.2 Machine Learning

Machine Learning benefits from the challenge of true applications. Real world applications require many capabilities that can be ignored in learnability proofs or performance measurements with respect to a dataset. Detecting a suitable learning task in an application is one of the hardest problems when dealing with real-world applications. Another problem is when applying a learning algorithm to a new application, how to start the feature selection or feature construction. Third, the use of background knowledge given by a domain expert determines our efforts in preparing the application. Fourth, the validation of machine learning results is an issue, when using a dataset, accuracy of prediction is an important criterion to be considered. In real world applications, accuracy-criteria include understandability and embeddedness. Understandability – how well an expert of the application domain can inspect the learning results in order to verify them. Embeddedness – how well the learning algorithm can be integrated into the overall application system. Pre- and post-processing are focused on the learning part of the application which simplifies the integration of several processes.

The integration of numerical and knowledge-based methods allows us validation of the processes carefully. Knowing the representation class for each process we can detect learning tasks. For learning state-action rules, the Support Vector Machine is applied, because it is capable of handling high dimensional numerical data. Since the support vector machine is a binary classifier, the overall task of finding state-action rules had to split into several particular learning tasks.

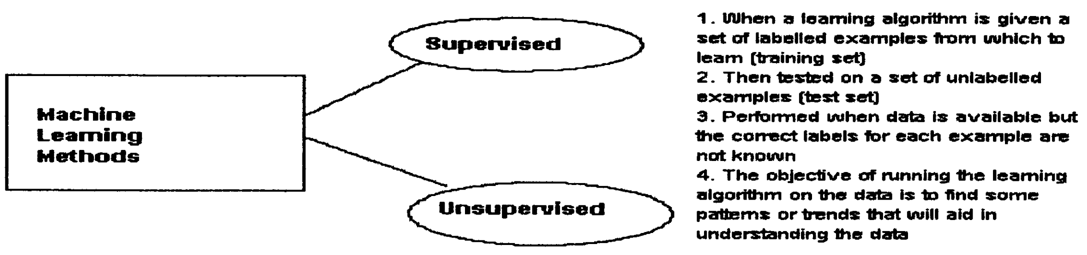

Machine Learning Methods
Supervised
Unsupervised
1. When a learning algorithm is given a set of labelled examples from which to learn (training set)
2. Then tested on a set of unlabelled examples (test set)
3. Performed when data is available but the correct labels for each example are not known
4. The objective of running the learning algorithm on the data is to find some patterns or trends that will aid in understanding the data

Figure 3.3 Supervised and Unsupervised Learning.

Support vector machines (Vapnik, 1998) are based on the Structural Risk Minimization principle from statistical learning theory. The idea of structural risk minimization is to find a hypothesis h for which the lowest error is guaranteed. Vapnik connects the bounds on the true error with the margin of separating hyperplanes. In their basic form support vector machines find the hyperplane that separates the training data with maximum margin. Since there are a lot of unbalanced number of positive and negative examples, cost factors C+ and C- are introduced to adjust the cost of false positives vs false negatives. This hyperplane value can be translated into an optimization problem.

Making a decision, atleast theoretically is possible only when history is considered in some form, after cross-validating the training set with optimized parameters. The parameters C+ and C- of the SVM are so chosen to obey the ratio -

C+ / C- = number of negative training examples/ number of positive training examples.

Classifying according to respective functionality homologies is very significant in studying biological processes. SVM can classify into different functional families which may involve structures with diverse sequence distributions. As the gap between the amount of structural information and functional characterization widens, increasing efforts should be directed towards developing a computational tool for retrieving homologous functionality secondary structures. In the absence of clear sequence or structural similarities, the criteria for comparison of distantly-related RNA secondary structures become increasingly difficult to formulate. A statistical learning method, Support Vector Machines is needed to classify these distantly related, may be of different structures, but with the same functionalities, for development of many biological studies, from theoretical concepts to practicality.

Samples of RNA secondary structures known to be in a functional class(positive samples) and those not in the class (negative samples) are used to train a SVM system to recognize specific features and classify them into either functional class or outside of the class. This approach may be applied to functional prediction for both distantly-related and closely-related structures. Given sufficient samples of RNA secondary structures of specific function, SVM can be trained and used to recognize RNA secondary structures with characteristics for a particular function. The functionally distinguished classes of proteins are collected from several databases and specifically trained and tested on each of the functional classes collected. First, every structure is represented by specific feature vector assembled from encoded representations of tabulated residue properties, which is scaled according to the optimal parameters. Different descriptors can be computed for different properties and all these descriptors can be combined to form the feature vector. Then SVM is fed and trained with examples of RNA secondary structures of a particular functional family (positive samples) and those that do not belong to this family (negative samples). The feature vectors of these positive and negative samples are input into the SVM system. The trained SVM system can then be used to classify a structure into either the positive group (structure predicted to be in the family) or the negative group (structure predicted to not belong to the family). The training system for each family is optimized and tested using separate testing sets of both positive and negative samples.

Some very important strategies to consider is to give more importance to the Data Collection, data collected from different databases, preparing the data (also known as preprocessing or data cleansing) which is the least glamorous and least discussed part. Preprocessing can be as simple as making sure the data is in the right format for the program that reads it, or it can involve extensive calculations. Another criterion is the Feature Selection problem in a "learning from samples" approach, which can be defined as choosing a subset of features that achieve the lowest error according to a certain loss function. This problem can be tackled using filter and wrapper methods. Filter methods use an indirect measure of the quality of selected features, e.g. evaluating the correlation function between each input feature and the observed output. Wrapper methods use as selection criteria the goodness-of-fit between the inputs and the output provided by the learning machine under consideration. Filter methods might fail to select the right subset of features if the used criteria deviates from the one used for training the learning machine, whereas wrapper methods can be computationally intensive due to the learning machine has to be retained for each new set of features.

CHAPTER 4

SUPPORT VECTOR MACHINES

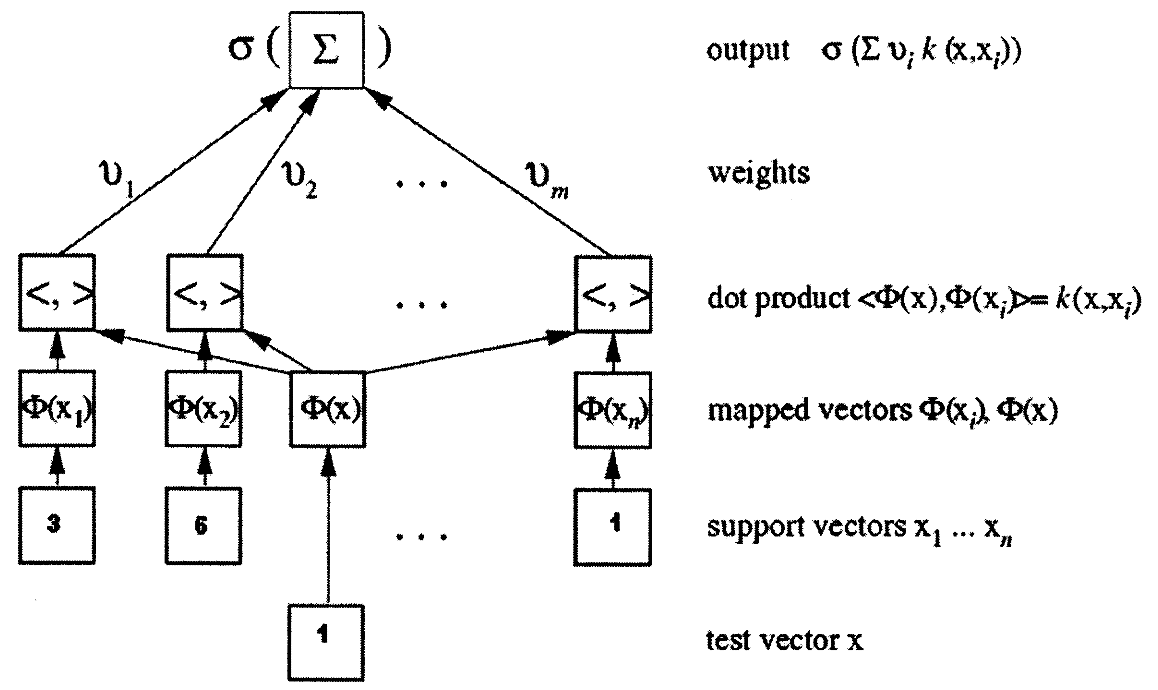
Support Vector Machines (SVM) algorithm (Boser et al.; Vapnik, 1998) is a classification algorithm that provides state-of-the-art performance in a wide variety of application domains (William Stafford Noble, 2003). Support Vector Machines combine two key ideas, an optimum margin classifier, which is a linear classifier that constructs a separating hyperplane which maximizes the distance to the training points. SVMs are well-suited to deal with learning tasks where the number of attributes is large with respect to the number of training examples. And the second concept is of a kernel, which is a function that calculates the dot product of two training vectors. SVM kernel methods can easily handle non-vector inputs, such as variable length sequences or graphs. Many biological problems involve high dimensional, noisy data, for which SVMs are known to behave well compared to other statistical or machine learning methods.

4.1 Theory and Principles of SVM

Support Vector Machines were invented by Vladimir Vapnik. They are a method for creating functions from a set of labeled training data. SVM learning is based on simple ideas and provides a clear intuition of learning from examples. It can lead to high performances in practical applications, especially in computational biological applications these days. SVMs can solve more complex models and algorithms, which cannot be solved even by neural networks that are much harder to analyze theoretically (Fig 4.1). It can be analyzed mathematically very easily, because it corresponds to a linear method in a high-dimensional feature space nonlinearly related to input space. Hence, using kernels is very essential in these cases, so that all computations can be performed directly in input space.

SVM methods deal with complex algorithms for non-linear pattern recognition, regression, or feature extraction. The function that is created by SVM can be a classification function or a general regression function. For classification, SVMs operate by finding a hypersurface in the space of possible inputs. The hypersurface attempts to split the positive test data from the negative test data. The split will be chosen so that it has the largest distance from the hypersurface to the nearest of the positive and negative data set. Intuitively, this makes the classification correct for testing data that is near, but not identical to the training data (Fig 4.1). To enable the machine learning algorithms to learn some sequences, the input sequence is mapped into a feature vector in the feature space.

There are three stages here, feature generation, feature selection, and feature integration. Given a set of labeled training vectors (positive and negative input examples), a SVM learns a linear decision boundary to discriminate between the two classes. The result would be a linear classification rule that can be used to classify new test examples. SVMs exhibit excellent generalization performance (accuracy) in practice and have strong theoretical motivation in statistical learning theory.



$\sigma(\Sigma)$
υ_1
υ_2
υ_m
<, >
$\Phi(x_1)$
$\Phi(x_2)$
$\Phi(x)$
$\Phi(x_n)$
3
6
1
1
output $\sigma(\Sigma \upsilon_i k(x,x_i))$
weights
dot product $<\Phi(x),\Phi(x_i)>= k(x,x_i)$
mapped vectors $\Phi(x_i)$, $\Phi(x)$
support vectors $x_1 \ldots x_n$
test vector x

Figure 4.1 SVM methodology

Relevant informative features should be selected from the large collection of database features, as machine learning faces tasks with high dimensional data. A correlation-based feature selection method is used for feature selection based on the concept of entropy. For designing learning algorithms, the need to define a class of functions arises, where capacity has to be computed accurately. SVM classifiers are based on the class of hyperplanes (Marti A.Hearst, IEEE intelligent systems).

$$(w.x) + b = 0 \quad w \in R^N, \; b \in R. \tag{1}$$

corresponds to decision functions

$$f(x) = \text{sign}\,((w.x) + b) \qquad (2)$$

Training a support vector machine (SVM) leads to a quadratic optimization problem with bound constraints and one linear equality constraint. The "Fisher score" vector introduced by Tommi Jaakola is a pretty good feature vector when used with a properly scaled radial-basis kernel, though it is large vector and many of the features are just adding noise.

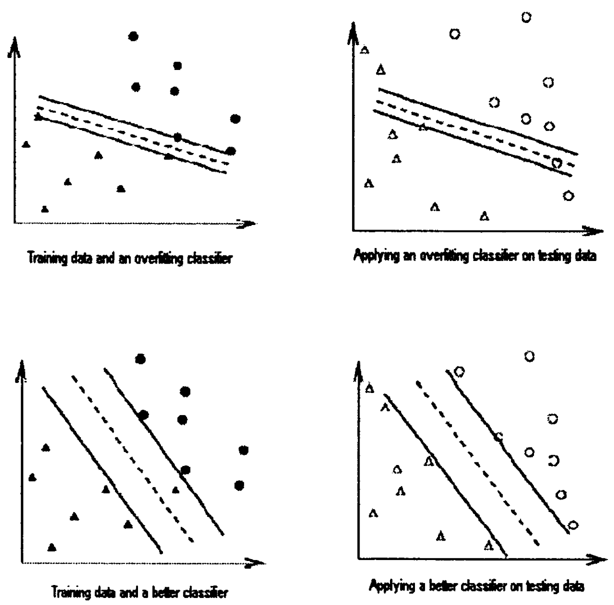

Training data and an overfitting classifier
Applying an overfitting classifier on testing data
Training data and a better classifier
Applying a better classifier on testing data

Figure 4.2 An overfitting classifier and a better classifier for training and testing data.

Large enormity of feature dimensionality can decrease the scalability and learning performance of the machine learning algorithms. High dimensional data can contain high degree of irrelevant and redundant information which may greatly degrade the performance of the learning algorithms. This is the reason why, these algorithms need an exponential increase in the number of training samples with respect to an increase in the dimensionality of the samples in order to uncover and learn the various dimensions to the nature of the samples. The selection of relevant informative features among the large collection of feasible features is necessary for machine learning tasks with high dimensional data (Fig 4.2).

In general, the optimum margin hyperplane will be a linear combination of the input vectors; support vectors are those training examples which obtain a non-zero coefficient, i.e., the ones that lie closest to the separating hyperplane. The learning problem is converted into an optimization problem, without local optima, to give a very good generalization performance. The main idea of a Support Vector Machine (SVM) is to map the data into some other dot product space (called feature space) F via a nonlinear map.

$$\Phi: R^N \rightarrow F, \tag{3}$$

and perform the above linear algorithm in F.

This only requires the evaluation of dot products,

$$k(x,y) := (\Phi(x) . \Phi(y)) \tag{4}$$

where $\Phi(x_i)$ should be substituted for each training example x_i, and perform the optimal hyperplane algorithm in F. In input space, the hyperplane corresponds to a nonlinear decision function whose form is determined by the kernel.

The choice of kernel functions and hence of the feature space to work in, determines both the functional form of the estimate and the objective function. The algorithm used in the SVM learning describes the following properties, that it is based on the statistical learning theory, it is more practical as it can be reduced to a quadratic equation which can be solved for a unique solution, and it contains many heuristic algorithms as special cases, where a choice of kernels can be applied for different applications to obtain different architectures such as polynomial classifiers, RBF classifiers or three–layer neural nets. In this model, the similarity features have been extracted from the scores of the RSMatch multiple sequence alignment results, using different parameters for feature selection and provided train.dat and test.dat files to train the SVM classifier.

4.2 Data Generation

Data is first generated from the RNa databases, RNA sequences randomly from different family classes and run through the RSMatch, which invokes automatically the Vienna RNA package to fold all those into secondary structures and do the multiple structure alignment with similarity scores. A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one "target value" (class labels) and several "attributes" (features). The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

A set of positive examples should also be generated for training the Support Vector Machines, so 100 positive examples are taken and also 100 negative example set is generated by reshuffling the same positive data set randomly. This negative set is generated, in order to train the machine, so that it can differentiate between the random structures – which are already known and which do not belong to those families.

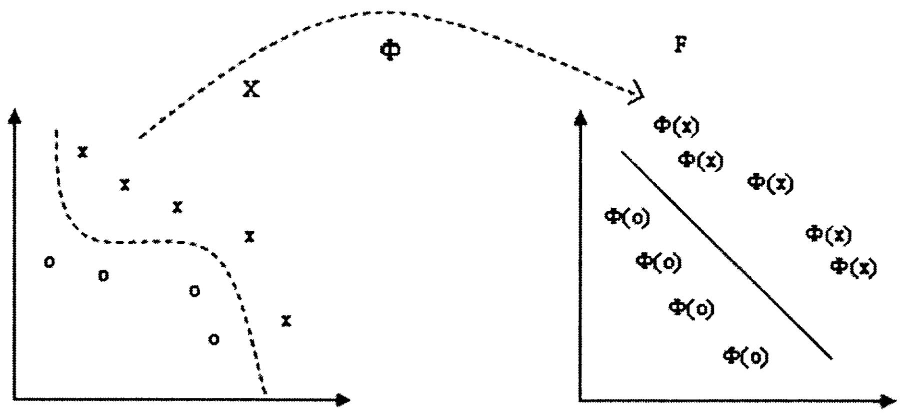


Figure4.3 Projections of "features" into feature space F where data points are discriminated from each other. Φ(x) and Φ(o) are feature vectors of the original data points x and o.

Functional RNA secondary structures can be identified in multiple sequence alignments with high sensitivity and high specificity. These are some of the ways, to choose informative features to train our machine: The variances can be computed for differences in the feature value between positive examples and negative examples. A good feature has a low variance between positives and a high variance between negatives. The distribution of the signs of the feature differences between positive and negative signs should be carefully featured, a good feature will have mostly positive or mostly negative signs. The distribution of the feature differences should also be considered, and can be optimized by multiplying or dividing by the mean or standard deviation for those features.

4.3 Kernels

Support Vector machines can be combined with various feature selection strategies for better performance. Some of them are known as filters, which are general feature selection methods independent of SVM, and the other type are wrapper type methods, which are modifications of SVM which choose important features as well as conduct training/testing.

Support Vector machines are a great promising approach for data classification, and its basic idea is to map data into a high dimensional space and find a separating hyperplane with the maximal margin. If the training vectors in two classes and a vector of labels are given, SVM can solve a quadratic optimization problem, but there is a need for one kernel definition here. There is a large choice of kernel functions, for various applications, based on the biophysical features selected as the similarity measure.

The usage of a kernel function is experimentally studied and proved for high accuracy, Kernel design for RNA structures (Taishin Kin et al.). It is hard to define kernel functions to reflect the secondary structures of RNA, and for defining the similarity (kernel) between two RNAs, simple sequence comparison algorithms are not enough, because the sequences can form stems, hairpins, bulges, etc. Even string kernels including spectrum kernels cannot define RNA similarities, because they cannot take into account the secondary structures of the RNA molecule.

A kernel is a definition of a similarity measure, and can be taken for two secondary structures (in this case).

If a non-linear function can be defined,

$\Phi: X \rightarrow F$, which maps the data points from X to F , where

$F = \{\Phi (z): z \in X\}$, and we can use a linear discriminator, f(z).

F is a feature space and Φ (z) is called a feature vector.

f(z) with respect to Φ (z):

$$f(z) = \sum_{i=1}^{N} \omega_i \Phi_i (z) + b \quad = \quad (\omega.\Phi (z)) + b \ , \quad f(x) \geq 0, f(0) < 0, \qquad (5)$$

where ω is parameter, which is a linear combination obtained from training data x (positive training data) and o (negative training data).

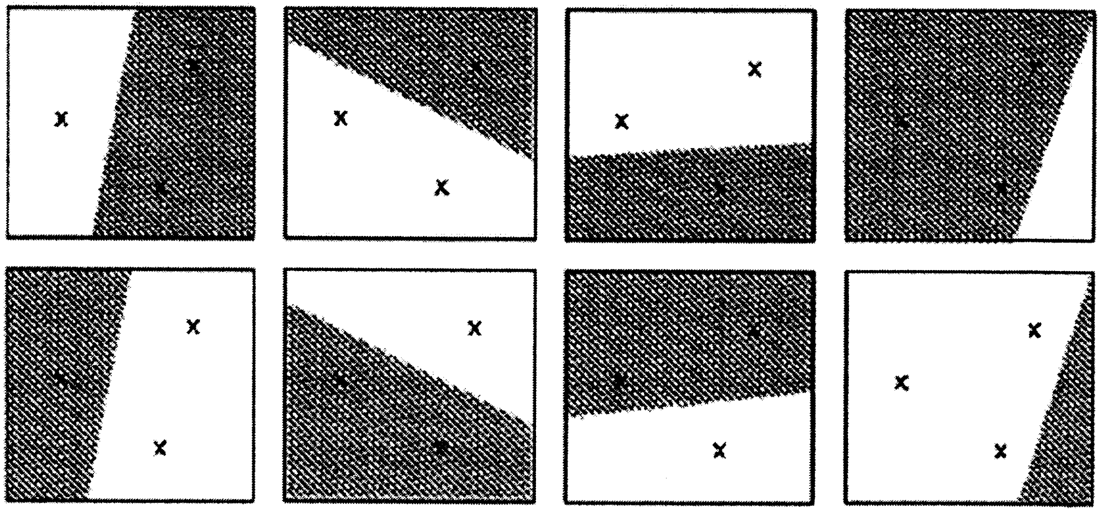


Figure 4.4 A simple VC dimension example for assigning 3 points in two classes using separating hyperplanes.

Generally, finding Φ is not that easy, but if the feature vector $\Phi(z)$ is computed directly, several complications of non-linear discriminant analysis can be bypassed. Such a method involving the direct computation is called a kernel function. When counting the occurrence of each nucleotide in a given sequence based on the assumption that counting each composition in a sequence reflects a basic feature of the sequence, representing the result of the counting as a vector and the dot product between two of such vectors is called a count kernel (Taishin et al.). Count kernels for labeled biological sequences indicating the secondary structure exploits the context information of the sequence data if it is available – whether the context means coding/non-coding.

If sequence of labels: $h = (h_1, \ldots h_m)$ in a sequence x, z represents a combined sequence of symbols and the labels, $z=(x, h)$. The first order and second order count kernels are defined.

4.4 Kernel Design for RNA Structures

RNA sequences form peculiar secondary structures and so a stochastic context free grammar (SCFG) is used for representing the RNA secondary structures (Tsuda et al.). But we're using kernel design for RNA structures using context free grammar (CFG) with the following generative rules:

$S \rightarrow R_1$, $R_1 \rightarrow P_1a$, $P_1 \rightarrow gP_2c$, $P_2 \rightarrow g\,P_3c$, $P_3 \rightarrow gL_1c$, $L_1 \rightarrow cL_2$, $L_2 \rightarrow aL_3$, $L_3 \rightarrow uE$,

where P represents the state which emits a canonical base pair. R and L correspond to the states which emit single base to right and left, respectively. S and E are special states which do not associate any symbols but represent start and end respectively.

The generative rules are interpreted as R_1 emits “a” to right and makes transition to P_1, then P_1 emits “g” to left and “c” to right simultaneously then move to P_2 and so on (Taishin et al.).

In general, there are six types of states in order to represent an RNA secondary structure:

S: start $S \rightarrow W$ - a special state which means the beginning of the secondary structure

B: bifurcation $B \rightarrow WW$ – makes transition to two states

P: pair-wise $P \rightarrow a\,Wb$ – emits two symbols a to left and b to right.

$ab \in \{ AU, UA, CG, GC\}$

L: left $L \rightarrow a\,W$ – emits single symbol a to left

R: right $R \rightarrow W\,a$ – emits single symbol a to right

E: end $E \rightarrow \epsilon$ - a special state which means the end of the secondary structure.

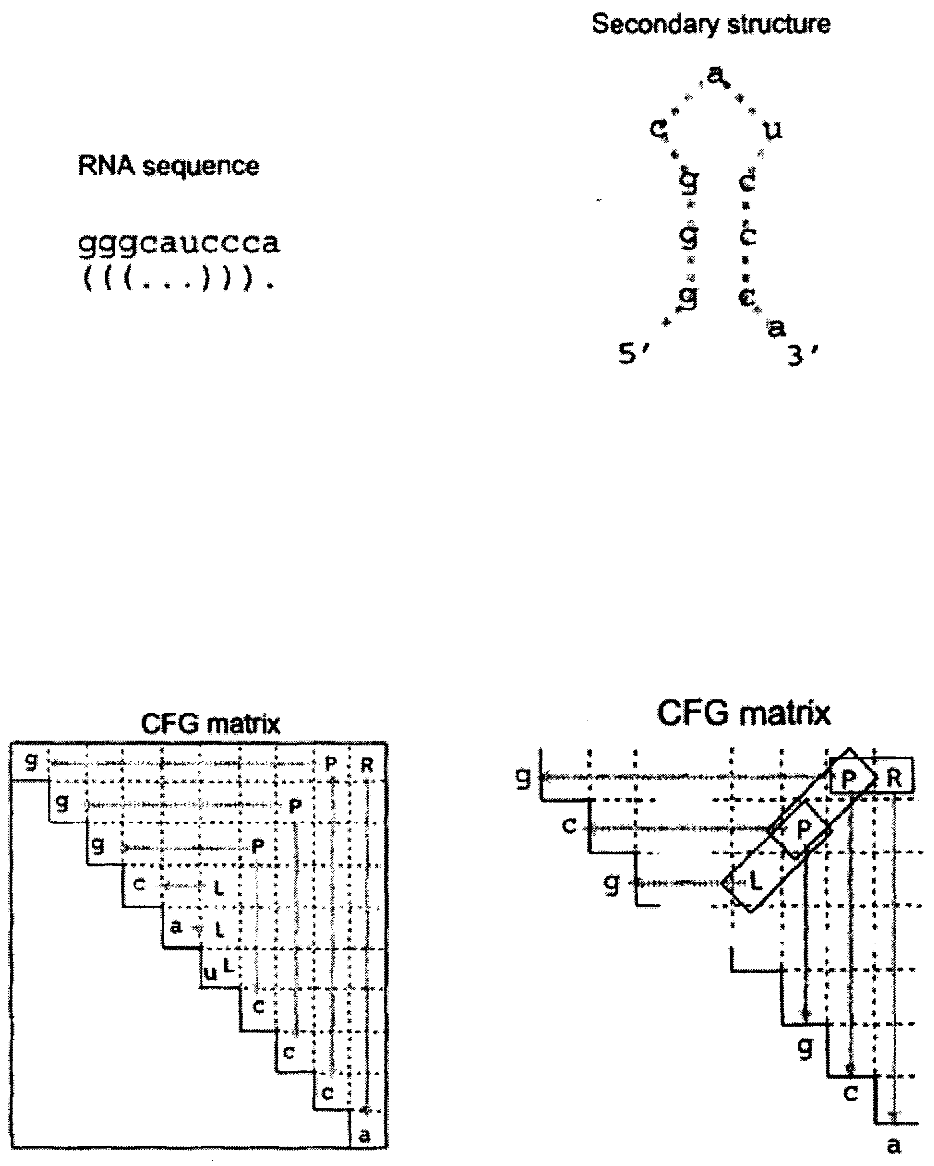


Figure 4.5 Counting scheme of the 2nd order feature vector from a CFG matrix.

Simplest form of a kernel is a function that calculates the dot product of two training vectors. Kernels calculate these dot products in feature space, often without explicitly calculating the feature vectors, operating directly on the input vectors instead.
A secondary structure of two RNAs is known and represented using context-free grammar (CFG) where a state (or a non-terminal) is associated with one or two bases in an RNA sequence. A feature vector is constructed by counting the base-state combinations and the kernel is defined as the dot product between the two vectors. This is known as "count kernel" for RNAs. And when we generalize this taking into account the consecutive two base state combinations, the resulting kernel is the second order count kernel for RNAs. When the RNA secondary structure is not known, as in many cases, a probabilistic model is estimated such as stochastic context-free grammar (SCFG) and nowadays, marginalized count kernels for unknown RNA structures are being used.

CHAPTER 5

IMPLEMENTATION

5.1 Theory

With the theory of RSMatch, the similar features from the secondary structures have been extracted, defining a similarity measure, and choosing a CFG count kernel in many different kernels available, and theory of Support Vector Machines, how they get trained with the training input and classify a test data. In this experimental study, the state of art SVM ^{struct} (Joachims.T), which is a Support Vector Machine (SVM) algorithm for predicting multivariate outputs are taken as the base program, from which some modifications were applied and some required particular options were obtained. It performs supervised learning by approximating a mapping h: X --> Y using labeled training examples (x_1, y_1), ..., (x_n, y_n). Unlike regular SVMs, however, which consider only univariate predictions like in classification and regression, SVM^{struct} can predict complex objects **y** like trees, sequences, or sets. A sparse approximation algorithm is implemented in SVM^{struct} and this implementation is based on the SVM^{light} quadratic optimizer (SVM^{light} is an implementation of Support Vector Machines in C (Joachims.T)).

SVM^{struct} can be thought of as an API for implementing different kinds of complex prediction algorithms which is currently implementing many learning tasks such as SVM^{multiclass} , which is a multi-class classification program that learns to predict one of **k**-mutually exclusive classes and SVM^{cfg} which learns a weighted context free grammar from example data. Training examples (e.g. for natural language parsing) specify the sentences along with the correct parse tree and the goal is to predict the parse tree of new sentences.

Based on this API's existing instantiation, the general sparse approximation training algorithm for this particular application has been specialized, by making minor changes to the existing code, referring to the algorithm and the code is changed for:

1) A function for computing the feature vector Psi.
2) A function for computing the argmax over the (kernelized) linear discriminant function.
3) A loss function.

The feature vectors have been taken from the RSMatch results and the kernel function is taken as the CFG count kernels, and implemented referring to this algorithm in this case study and results were analyzed.

SVM^{multiclass} is an implementation of the multi-class Support Vector Machine (SVM). While the optimization problem is the same as in SVM^{light} , this implementation uses a different algorithm which is described in SVM^{struct}. This implementation is an instance of SVM^{struct}. It was not optimized for speed by exploiting special properties of the multi-class optimization problem, nevertheless, this implementation can handle fairly large problems.

The implementation was developed on Solaris 2.7 with gcc, but compiles also on Linux, IRIX, Cygwin, and Powermac.

The source code is available at the following location:

http://download.joachims.org/svm_multiclass/current/svm_multiclass.tar.gz

The archive should be expanded into the current directory, with all the relevant files and it can be compiled using the command:

```
make
```

This will produce the executables svm_multiclass_learn and svm_multiclass_classify in the current directory, from which we can start working with the program.

5.2 Using the software

SVM^{multiclass} consists of:

- a learning module (svm_multiclass_learn) and
- a classification module (svm_multiclass_classify).

The classification module is used to apply the learned model to new examples (usage is much like SVM^{light}).

You call the function:

```
svm_multiclass_learn -c 1.0 train.dat model.dat
```

which trains an SVM on the training set train.dat and outputs the learned rule to model.dat using the regularization parameter C set to 1.0.

Other options are:

1) General options:

```
-?          -> this help
-v [0..3]   -> verbosity level (default 1)
```

```
-y [0..3]   -> verbosity level for svm_light
               (default 0)
```

2) Learning options:

```
-c float   -> C: trade-off between training error and margin
               (default 0.01)
-p [1,2]   -> L-norm to use for slack variables. Use 1 for L1-norm,
              use 2 for squared slacks.
              (default 1)


-o [1,2]   -> Slack rescaling method to use for loss.
              1: slack rescaling
              2: margin rescaling
              (default 1)


-l [0..]   -> Loss function to use.
              0: zero/one loss
              (default 0)
```

3) Kernel options:

-t int -> type of kernel function:

0: linear (default)

1: polynomial (s a*b+c)^d

2: radial basis function exp(-gamma ||a-b||^2)

3: sigmoid tanh(s a*b + c)

4: user defined kernel from kernel.h

-d int -> parameter d in polynomial kernel

-g float -> parameter gamma in rbf kernel

-s float -> parameter s in sigmoid/poly kernel

-r float -> parameter c in sigmoid/poly kernel

-u string -> parameter of user defined kernel

4) Optimization options:

-q [2..] -> maximum size of QP-subproblems (default 10)

-n [2..q] -> number of new variables entering the working set in each iteration

(default n = q). Set n size of cache for kernel evaluations in MB (default 40)

The larger the faster...

-e float -> eps: Allow that error for termination criterion

(default 0.01)

-h [5..] -> number of iterations a variable needs to be

optimal before considered for shrinking (default 100)

-k [1..] -> number of new constraints to accumulate before recomputing the QP

solution (default 100)

-# int -> terminate optimization, if no progress after this number of iterations.

(default 10000)

5) Output options:

-a string -> write all alphas to this file after learning

(in the same order as in the training set)

6) Structure learning options:

none

The input file, here, used to collect the features into the `training_file` contains the training example set, the target value is a positive integer that indicates the class. The first lines may contain comments and are ignored if they start with #. Each of the following lines represents one training example and is of the following format:

```
<line>    .=.    <target>    <feature>:<value>    <feature>:<value>    ...
<feature>:<value>                    #                    <info>
<target>.=.<integer>
```

```
<feature>.=.<integer>
<value>.=.<float>
<info> .=. <string>
```

The target value and each of the feature/value pairs are separated by a space character. Feature/value pairs will be ordered by increasing feature number. Features with value zero can be skipped. The target value denotes the class of the example via a positive integer.

For example, the line

```
3 1:0.43 3:0.12 9284:0.2 # abcdef
```

specifies an example of class 3 for which feature number 1 has the value 0.43, feature number 3 has the value 0.12, feature number 9284 has the value 0.2, and all the other features have value 0. In addition, the string `abcdef` is stored with the vector, which can serve as a way of providing additional information when adding user defined kernels.

The result of `svm_multiclass_learn` is the model which is learned from the training data in `training_file`. The model is written to `model_file`. To make predictions on test examples, `svm_multiclass_classify` reads this file. `svm_multiclass_classify` is called as follows:

svm_multiclass_classify [options] training_file model_file output_file. And for all the test examples in `training_file`, the predicted classes are written to `output_file`. There is one line per test example in `output_file` in the same order as in `training_file`.

This is the implementation used in this experimental study to train the Support Vector machine, by training with a training set of data in training_file, where the SVM learns and the model is written to the model_file. And after reading the predictions file, how the SVM classifier classifies the training file's model into the output file can be defined.

Support Vector Machine (Boser, Guyon, Vapnik, 1992) for classification involves training and testing data which consists of some data instances. Each instance in the training set contains one "target value" (class labels) and several "attributes" (families). To classify and compare the randomly taken RNA sequences from different RNA families, in this experimental analysis, the Hepatitis virus family is taken as an example data set. Hepatitis infects mammals and belongs to the family of the Hepad naviridae. SVM is given the training data set with 690 sequences taken randomly into training/testing data, where 6 features were calculated as descriptors for the required properties taken into account.

RSMatch is applied for feature extraction based on the scores when multiple structure alignment is done, and these features along with the count kernel definitions, trained to an edited API, $SVM^{multiclass}$ learns a training set of data to model it into model_file, which is hence classified and the result is printed out to output_file. With training, validation, and testing data together, these experimental results are accessed according to the classifier performance with sensitivity and specificity by computing area under ROC (Receiver Operating Characteristic curve), which is a general measure of the discrimination ability.

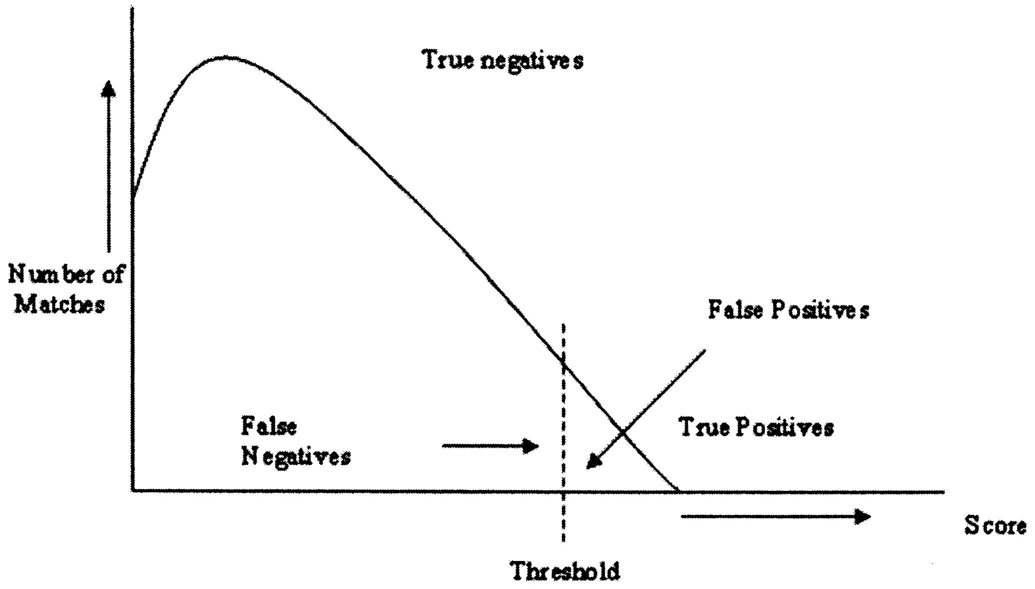


Figure 5.1 True negatives, True positives, False negatives, False positives.

These experiments have been done by supervised learning – which is a task of determining the function, given a sample of input/output pairs (training sample), that maps any input to an output such that disagreement with future input/output pairs is minimized. ROC curves are plots of the function of false-positives (FFP) f_p / t_n + f_p versus the fraction of true-positives (FTP) t_p / t_p + f_n (Fig 5.1).

CHAPTER 6

CONCLUSION

Classifying RNA secondary structures is a big challenge in the field of computational biology today, where computer tools and automated analysis play a very important role. This thesis intends to study and combine the state of art programs, to extract features, using RSMatch (Liu, J. et al.) – from secondary structures of an RNA molecule and then defining a kernel function based on count kernels on RNA secondary structures (Taishin et al.) and then training these sample features, into a Support Vector Machine referring to the algorithm implemented in SVM^{multiclass} software (Joachims. T) or the multi-class optimization.

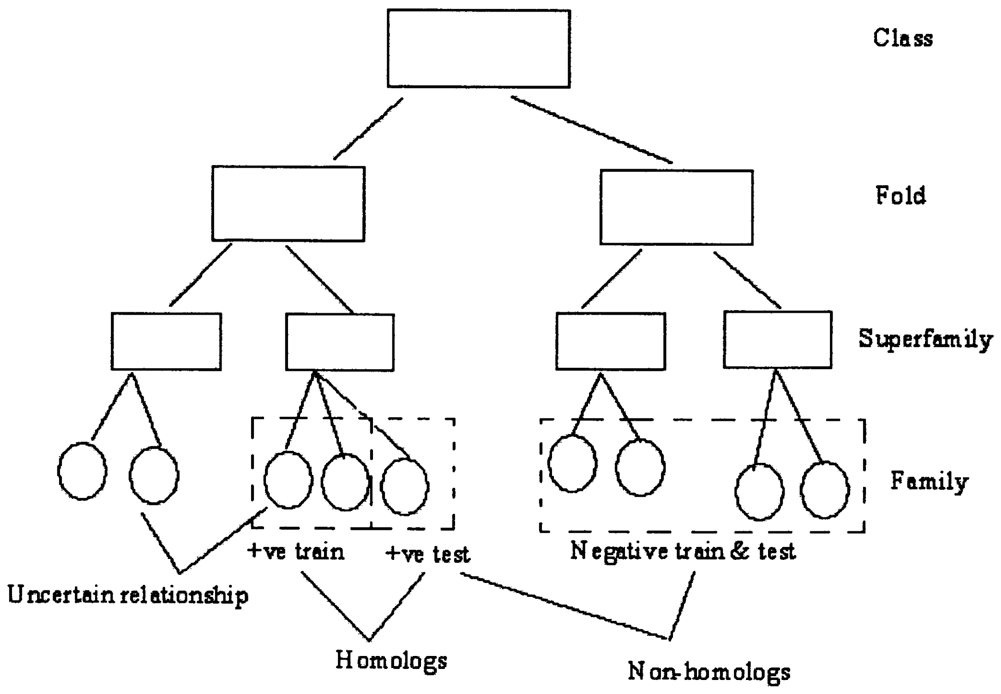


Figure 6.1 Classification model

This is an experimental study based and extended study of new methods that were mentioned in two of the recent papers, classifying RNA secondary structures using the traditional clustering algorithms and classifying based on labeled dual graphs. Using the combination of state of the art programs and software, this is a trial to extend it into a stand–alone model having all the components in the future with more accuracy and better performance and less time-consuming input–output results.

This is an initiative to learn more and apply different combinations of the state of the art tools available now, optimizing the parameters and training different algorithms in order to be able to learn the training examples and predict different queries on the knowledge base which has unseen data points and classify them into nearly same conditions specified for the training set. The future work may be extrapolated to develop along these steps and plan for developing a tool which has all the underlying integrations of SVMs, kernels, and different state of the art algorithms, with an easy and good user interface, which classifies different new structures (rising rapidly daily in new databases) easily and readily, even by a person who need not learn all the underlying principles of all algorithms used. This can lead to development of a good tool which has efficient classification performance rather than tedious, time-taking manual curation.

APPENDIX

SAMPLE INPUT AND OUTPUT FILES

The Appendix contains the samples and results.

```
2,30,2,1,2,2,2,2,1,2,2,2,2,2,1.00,85,18,4.0,?,1
2,50,1,1,2,1,2,2,1,2,2,2,2,2,0.90,135,42,3.5,?,1
2,78,1,2,2,1,2,2,2,2,2,2,2,2,0.70,96,32,4.0,?,1
2,31,1,?,1,2,2,2,2,2,2,2,2,2,0.70,46,52,4.0,80,1
2,34,1,2,2,2,2,2,2,2,2,2,2,2,1.00,?,200,4.0,?,1
2,34,1,2,2,2,2,2,2,2,2,2,2,2,0.90,95,28,4.0,75,1
1,51,1,1,2,1,2,1,2,2,1,1,2,2,?,?,?,?,?,1
2,23,1,2,2,2,2,2,2,2,2,2,2,2,1.00,?,?,?,?,1
2,39,1,2,2,1,2,2,2,1,2,2,2,2,0.70,?,48,4.4,?,1
2,30,1,2,2,2,2,2,2,2,2,2,2,2,1.00,?,120,3.9,?,1
2,39,1,1,1,2,2,2,1,1,2,2,2,2,1.30,78,30,4.4,85,1
2,32,1,2,1,1,2,2,2,1,2,1,2,2,1.00,59,249,3.7,54,1
2,41,1,2,1,1,2,2,2,1,2,2,2,2,0.90,81,60,3.9,52,1
2,30,1,2,2,1,2,2,2,1,2,2,2,2,2.20,57,144,4.9,78,1
2,47,1,1,1,2,2,2,2,2,2,2,2,2,?,?,60,?,?,1
2,38,1,1,2,1,1,1,2,2,2,2,1,2,2.00,72,89,2.9,46,1
2,66,1,2,2,1,2,2,2,2,2,2,2,2,1.20,102,53,4.3,?,1
2,40,1,1,2,1,2,2,2,1,2,2,2,2,0.60,62,166,4.0,63,1
2,38,1,2,2,2,2,2,2,2,2,2,2,2,0.70,53,42,4.1,85,2
2,38,1,1,1,2,2,2,1,1,2,2,2,2,0.70,70,28,4.2,62,1
2,22,2,2,1,1,2,2,2,2,2,2,2,2,0.90,48,20,4.2,64,1
2,27,1,2,2,1,1,1,1,1,1,1,2,2,1.20,133,98,4.1,39,1
2,31,1,2,2,2,2,2,2,2,2,2,2,2,1.00,85,20,4.0,100,1
2,42,1,2,2,2,2,2,2,2,2,2,2,2,0.90,60,63,4.7,47,1
2,25,2,1,1,2,2,2,2,2,2,2,2,2,0.40,45,18,4.3,70,1
2,27,1,1,2,1,1,2,2,2,2,2,2,2,0.80,95,46,3.8,100,1
2,49,1,1,1,1,1,1,2,1,2,1,2,2,0.60,85,48,3.7,?,1
2,58,2,2,2,1,2,2,2,1,2,1,2,2,1.40,175,55,2.7,36,1
2,61,1,1,2,1,2,2,1,1,2,2,2,2,1.30,78,25,3.8,100,1
2,51,1,1,1,1,1,2,2,2,2,2,2,2,1.00,78,58,4.6,52,1
1,39,1,1,1,1,1,2,2,1,2,2,2,2,2.30,280,98,3.8,40,1
1,62,1,1,2,1,1,2,?,?,2,2,2,2,1.00,?,60,?,?,1
2,41,2,2,1,1,1,1,2,2,2,2,2,2,0.70,81,53,5.0,74,1
```

Figure A1 Hepatitis features.

```
Hepatitis Domain (Sources unknown)
Past Usage:
       Diaconis,P. & Efron,B. (1983).  Computer-Intensive Methods in
       Statistics.  Scientific American, Volume 248.
       -- Gail Gong reported a 80% classfication accuracy

Number of Instances: 155
Number of Attributes: 20 (including the class attribute)

Attribute information:
      1. Class: DIE, LIVE
      2. AGE: 10, 20, 30, 40, 50, 60, 70, 80
      3. SEX: male, female
      4. STEROID: no, yes
      5. ANTIVIRALS: no, yes
      6. FATIGUE: no, yes
      7. MALAISE: no, yes
      8. ANOREXIA: no, yes
      9. LIVER BIG: no, yes
     10. LIVER FIRM: no, yes
     11. SPLEEN PALPABLE: no, yes
     12. SPIDERS: no, yes
     13. ASCITES: no, yes
     14. VARICES: no, yes
     15. BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00
         -- see the note below
     16. ALK PHOSPHATE: 33, 80, 120, 160, 200, 250
     17. SGOT: 13, 100, 200, 300, 400, 500,
     18. ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0, 6.0
     19. PROTIME: 10, 20, 30, 40, 50, 60, 70, 80, 90
     20. HISTOLOGY: no, yes
```

Figure A2 Data file.

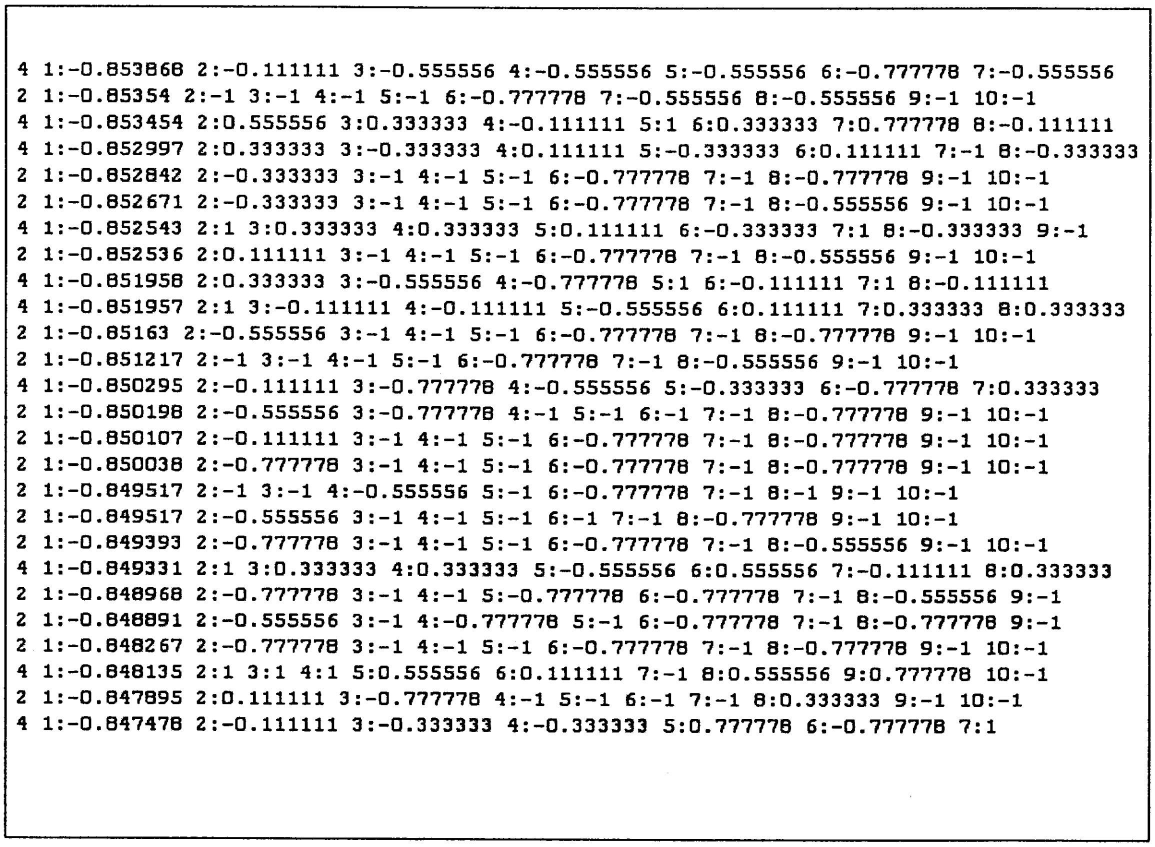
```
Missing Attribute Values: (indicated by "?")
    Attribute Number:    Number of Missing Values:
                  1:     0
                  2:     0
                  3:     0
                  4:     1
                  5:     0
                  6:     1
                  7:     1
                  8:     1
                  9:     10
              10:     11
              11:     5
              12:     5
              13:     5
              14:     5
              15:     6
              16:     29
              17:     4
              18:     16
              19:     67
              20:     0

Class Distribution:
    DIE: 32
    LIVE: 123
```

Figure A3 Data file.

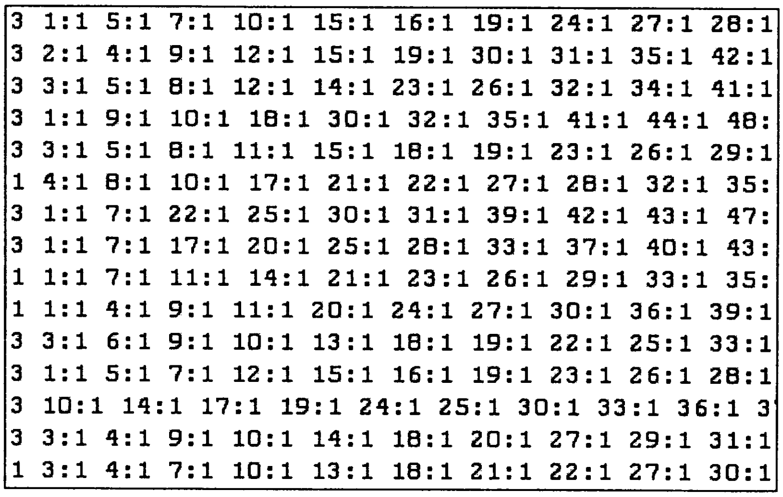
```
2.000000  1:1000025.000000 2:5.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:3.000000
2.000000  1:1002945.000000 2:5.000000 3:4.000000 4:4.000000 5:5.000000 6:7.000000 7:10.000000 8:3.00000
2.000000  1:1015425.000000 2:3.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:2.000000 8:3.000000
2.000000  1:1016277.000000 2:6.000000 3:8.000000 4:8.000000 5:1.000000 6:3.000000 7:4.000000 8:3.000000
2.000000  1:1017023.000000 2:4.000000 3:1.000000 4:1.000000 5:3.000000 6:2.000000 7:1.000000 8:3.000000
4.000000  1:1017122.000000 2:8.000000 3:10.000000 4:10.000000 5:8.000000 6:7.000000 7:10.000000 8:9.000
2.000000  1:1018099.000000 2:1.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:10.000000 8:3.00000
2.000000  1:1018561.000000 2:2.000000 3:1.000000 4:2.000000 5:1.000000 6:2.000000 7:1.000000 8:3.000000
2.000000  1:1033078.000000 2:2.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:1.000000
2.000000  1:1033078.000000 2:4.000000 3:2.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:2.000000
2.000000  1:1035283.000000 2:1.000000 3:1.000000 4:1.000000 5:1.000000 6:1.000000 7:1.000000 8:3.000000
2.000000  1:1036172.000000 2:2.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:2.000000
4.000000  1:1041801.000000 2:5.000000 3:3.000000 4:3.000000 5:3.000000 6:2.000000 7:3.000000 8:4.000000
2.000000  1:1043999.000000 2:1.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:3.000000 8:3.000000
4.000000  1:1044572.000000 2:8.000000 3:7.000000 4:5.000000 5:10.000000 6:7.000000 7:9.000000 8:5.00000
4.000000  1:1047630.000000 2:7.000000 3:4.000000 4:6.000000 5:4.000000 6:6.000000 7:1.000000 8:4.000000
2.000000  1:1048672.000000 2:4.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:2.000000
2.000000  1:1049815.000000 2:4.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:3.000000
4.000000  1:1050670.000000 2:10.000000 3:7.000000 4:7.000000 5:6.000000 6:4.000000 7:10.000000 8:4.0000
2.000000  1:1050718.000000 2:6.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:3.000000
4.000000  1:1054590.000000 2:7.000000 3:3.000000 4:2.000000 5:10.000000 6:5.000000 7:10.000000 8:5.0000
4.000000  1:1054593.000000 2:10.000000 3:5.000000 4:5.000000 5:3.000000 6:6.000000 7:7.000000 8:7.00000
2.000000  1:1056784.000000 2:3.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:2.000000
2.000000  1:1059552.000000 2:1.000000 3:1.000000 4:1.000000 5:1.000000 6:2.000000 7:1.000000 8:3.000000
4.000000  1:1065726.000000 2:5.000000 3:2.000000 4:3.000000 5:4.000000 6:2.000000 7:7.000000 8:3.000000
```

Figure A4 Training data.



```
4 1:-0.853868 2:-0.111111 3:-0.555556 4:-0.555556 5:-0.555556 6:-0.777778 7:-0.555556
2 1:-0.85354 2:-1 3:-1 4:-1 5:-1 6:-0.777778 7:-0.555556 8:-0.555556 9:-1 10:-1
4 1:-0.853454 2:0.555556 3:0.333333 4:-0.111111 5:1 6:0.333333 7:0.777778 8:-0.111111
4 1:-0.852997 2:0.333333 3:-0.333333 4:0.111111 5:-0.333333 6:0.111111 7:-1 8:-0.333333
2 1:-0.852842 2:-0.333333 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.777778 9:-1 10:-1
2 1:-0.852671 2:-0.333333 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.555556 9:-1 10:-1
4 1:-0.852543 2:1 3:0.333333 4:0.333333 5:0.111111 6:-0.333333 7:1 8:-0.333333 9:-1
2 1:-0.852536 2:0.111111 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.555556 9:-1 10:-1
4 1:-0.851958 2:0.333333 3:-0.555556 4:-0.777778 5:1 6:-0.111111 7:1 8:-0.111111
4 1:-0.851957 2:1 3:-0.111111 4:-0.111111 5:-0.555556 6:0.111111 7:0.333333 8:0.333333
2 1:-0.85163 2:-0.555556 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.777778 9:-1 10:-1
2 1:-0.851217 2:-1 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.555556 9:-1 10:-1
4 1:-0.850295 2:-0.111111 3:-0.777778 4:-0.555556 5:-0.333333 6:-0.777778 7:0.333333
2 1:-0.850198 2:-0.555556 3:-0.777778 4:-1 5:-1 6:-1 7:-1 8:-0.777778 9:-1 10:-1
2 1:-0.850107 2:-0.111111 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.777778 9:-1 10:-1
2 1:-0.850038 2:-0.777778 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.777778 9:-1 10:-1
2 1:-0.849517 2:-1 3:-1 4:-0.555556 5:-1 6:-0.777778 7:-1 8:-1 9:-1 10:-1
2 1:-0.849517 2:-0.555556 3:-1 4:-1 5:-1 6:-1 7:-1 8:-0.777778 9:-1 10:-1
2 1:-0.849393 2:-0.777778 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.555556 9:-1 10:-1
4 1:-0.849331 2:1 3:0.333333 4:0.333333 5:-0.555556 6:0.555556 7:-0.111111 8:0.333333
2 1:-0.848968 2:-0.777778 3:-1 4:-1 5:-0.777778 6:-0.777778 7:-1 8:-0.555556 9:-1
2 1:-0.848891 2:-0.555556 3:-1 4:-0.777778 5:-1 6:-0.777778 7:-1 8:-0.777778 9:-1
2 1:-0.848267 2:-0.777778 3:-1 4:-1 5:-1 6:-0.777778 7:-1 8:-0.777778 9:-1 10:-1
4 1:-0.848135 2:1 3:1 4:1 5:0.555556 6:0.111111 7:-1 8:0.555556 9:0.777778 10:-1
2 1:-0.847895 2:0.111111 3:-0.777778 4:-1 5:-1 6:-1 7:-1 8:0.333333 9:-1 10:-1
4 1:-0.847478 2:-0.111111 3:-0.333333 4:-0.333333 5:0.777778 6:-0.777778 7:1
```

Figure A5 Validation data.



```
3 1:1 5:1 7:1 10:1 15:1 16:1 19:1 24:1 27:1 28:1
3 2:1 4:1 9:1 12:1 15:1 19:1 30:1 31:1 35:1 42:1
3 3:1 5:1 8:1 12:1 14:1 23:1 26:1 32:1 34:1 41:1
3 1:1 9:1 10:1 18:1 30:1 32:1 35:1 41:1 44:1 48:
3 3:1 5:1 8:1 11:1 15:1 18:1 19:1 23:1 26:1 29:1
1 4:1 8:1 10:1 17:1 21:1 22:1 27:1 28:1 32:1 35:
3 1:1 7:1 22:1 25:1 30:1 31:1 39:1 42:1 43:1 47:
3 1:1 7:1 17:1 20:1 25:1 28:1 33:1 37:1 40:1 43:
1 1:1 7:1 11:1 14:1 21:1 23:1 26:1 29:1 33:1 35:
1 1:1 4:1 9:1 11:1 20:1 24:1 27:1 30:1 36:1 39:1
3 3:1 6:1 9:1 10:1 13:1 18:1 19:1 22:1 25:1 33:1
3 1:1 5:1 7:1 12:1 15:1 16:1 19:1 23:1 26:1 28:1
3 10:1 14:1 17:1 19:1 24:1 25:1 30:1 33:1 36:1 3
3 3:1 4:1 9:1 10:1 14:1 18:1 20:1 27:1 29:1 31:1
1 3:1 4:1 7:1 10:1 13:1 18:1 21:1 22:1 27:1 30:1
```

Figure A6 Scaling data.

Name	Type	Class	Training size	Testing size	Feature
Hepatitis	Classification	3	690		6

Figure A7 Classification based on the result file.

REFERENCES

1. Klosterman, R., Tamura, M., Holbrook, R., and Brenner, E., SCOR: a Structural Classification of RNA database, Nucleic Acids Research, 2002, Vol.30, No.1.

2. RNA databases and softwares, http://www.rnasociety.org/links

3. Durbin, R., Eddy, S., Krogh, A., and Mitchison, G., Biological Sequence Analysis, Probabilistic models of proteins and nucleic acids.

4. Chen, Y., and Lin, J., Combining SVMs with Various Feature Selection Strategies, Department of Computer Science, National Taiwan University, Taipei 106, Taiwan.

5. Karklin, Y., Meraz, F., and Holbrook, R., Classification of Non-Coding RNA Using Graph Representations of Secondary Structure, September 22, 2004.

6. Zuker, M., Calculating nucleic acid secondary structure, Curr Opin Struct Biol, 10(3):303-10, 2000.

7. Kin, T., Tsuda, K., and Asai, K., Marginalized Kernels for RNA Sequence Data Analysis, Genome Informatics 13: 112-122(2002).

8. Rivas, E. and Eddy, S., A dynamic programming algorithm for RNA structure prediction including pseudoknots, Journal of Molecular Biology, 283:1168-1171, 1999.

9. Rivas, E. and Eddy, S., Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs, Bioinformatics, 16:573-585, 2000.

10. Zuker, M., Computer prediction of RNA structure, Methods in Enzymology, 180:262-288, 1989.

11. Zuker, M., On folding all suboptimal foldings of an RNA molecule, Science, 244:48-52, 1989.

12. Vert, J., Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings, In Proc. Pacific Symposium on Biocomputing 2002, 649-660, 2002.

13. Joachims, T. <thorsten@joachims.org>, Cornell University, SVM light, Support Vector Machine, September, 2004.

14. Joachims, T., 11 in: Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, Scholkopf, B., and Burges, C., and Smola, A.(ed.), MIT Press, 1999.

15. Vapnik, N., The Nature of Statistical Learning Theory. Springer, 1995.

16. Yang, L., Hsu, W., Lee, M., Wong, L., SVM-Based Identification of MicroRNA Precursors, July,2005.

17. Flach, A., On the state of the art in Machine Learning: a personal review, Dec 2000.

18. Kin, T., Tsuda, K., and Asai, K., Computation and Application of Marginalized Kernels for biological sequence data, August 2002.

19. Noble, W., Support vector machine applications in computational biology.

20. Washietl, S., Hofacker, L., and Stadler, F., Fast and reliable prediction of noncoding RNAs.

21. Chang, C., & Lin, C., J. LIBSVM: a library for support vector machines(2001). Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

22. Stocsits, R., Hofacker, L., and Stadler, F., Conserved Secondary Structures in Hepatitis B virus RNA.

23. Thompson, J., Higgs, D., and Gibson, T. CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penal ties and weight matrix choice, Nucl. Acids Res., 22:4673-4680, 1994.

24. Marti, A., Trends and Controversies - Support Vector Machines, IEEE Intelligent Systems, Aug 1998.

25. Vapnik, V. Statistical Learning Theory. Wiley-Interscience, New York, 1998.

26. Nakaya, A., Yonezawa, A., and Yamamoto, A., Classification of RNA Secondary Structures Using the Techiques of Cluster Analysis, J.theor.Biol, 1996.

27. Condon, A., Problems on RNA Secondary Structure Prediction and Design.

28. Zuker, M., and Steigler, P., Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, Nucleic Acids Res 9, 1981, 133-148.

29. Westhof, E., and Fritsch, V., RNA folding: beyond Watson-Crick pairs, Structure 2000, 8:R55-R65, 2000.

30. Akutsu, T., Dynamic programming algorithms for RNA secondary prediction with pseudoknots, Discrete Applied Mathematics, 104, 2000, 45-62.

31. Wong, L., The Practical Bioinformatician, World Scientific Publishing Co., 2004.

32. Michael, S., Introduction to Computational Biology - Maps, sequences and genomes, Chapman & Hall, 1995.

33. Ding, Y., and Charles, E., A statistical sampling algorithm for RNA secondary structure prediction, 7280-7301 Nucleic Acids Research, 2003, Vol.31, No.24.

34. Rastegari, B., and Condon, A., Linear Time Algorithm for Parsing RNA Secondary Structure, Dept of Computer Science, University of British Columbia.

35. Jaime, E., Shigenobu, Y., Ichiishi, E., Carlos, A., RNA 3D Structure Prediction: (1) Assessing RNA 3D Structure Similarity from 2D Structure Similarity, Genome Informatics 15(2): 112-120(2004).

36. Deschenes, A., Kay, C., Jagdeep, P., Comparison of Dynamic Programming and Evolutionary Algorithms for RNA Secondary Structure Prediction.

37. Takakura, T., Asakawa, H., Seki, S., Kobayashi, S., Efficient Tree Grammatical Modeling of RNA Secondary Structures from Alignment Data.

38. Yang, H., Jossinet, F., Leontis, N., Chen, L., Westbrook, J., Berman, H. and Westhof, H., Tools for the automatic identification and classification of RNA base pairs, 3450-3460, Nucleic Acids Research, 2003, Vol.31, No.13.

39. Liu, J., Wang, J.T., Hu, J., and Tian, B. A method for aligning RNA secondary structures and its application to RNA motif detection. BMC Bioinformatics 2005, 6:89. Software at http://exon.umdnj.edu/software/RSmatch.

40. Rfam: annotating non-coding RNAs in complete genomes. Griffiths-Jones, S., Simon, M., Marshall, M., Khanna, A., Eddy, S., and Bateman, A., Nucleic Acids Res. 2005 33:D121-D124.

41. Rfam: an RNA family database. Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., and Eddy, A. Nucleic Acids Res. 2003 33(1):439-441.

42. Mathews, D.H., Sabina, J., Zuker., M. & Turner, D.H., Expanded Sequence Dependence of Thermodynamic Parameters Improves Prediction of RNA Secondary Structure, J. Mol. Biol. 288, 911-940 (1999)