University "Alexandru Ioan Cuza" of Iaşi
Faculty of Computer Science

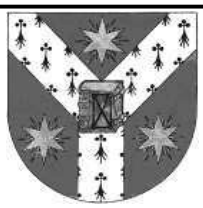
TECHNICAL REPORT

Support Vector Machines for MicroRNA Identification

Liviu Ciortuz

TR 08-01, June 2008

ISSN 1224-9327



Universitatea "Alexandru Ioan Cuza" Iaşi
Facultatea de Informatică
Str. Berthelot 16, 6600-Iaşi, Romania
Tel. +40-32-201090, email: bibl@infoiasi.ro

Support Vector Machines for MicroRNA Identification

Liviu Ciortuz

Department of Computer Science
"Al.I. Cuza" University, Iaşi, Romania,
ciortuz@info.uaic.ro

Abstract. This technical report provides an overview on the support vector machines that have been designed during the recent years for the identification of microRNA sequences. Our aim is to arrive in the end to identify potential ways in which the quality of such machine learning methods could be further improved, knowing that the rate of false positves they produce is still too high.

Introduction

Several attempts have been made in the past to provide a survey of existing systems for microRNAs identification [7] [37] [29] but due to the significant number of recent such systems, these surveys are already outdated. None of the cited papers mentions any single Support Vector Machine (SVM) system, altough there are to date at least a dozen papers describing such systems, and they give now the best results for microRNA identification. It seems to us that the present technical report is the first one to offer a review of existing SVMs for miRNAs identification.

Section 1 constitutes an introduction to the microRNA world. It firstly presents the phenomenon of RNA-mediated interference, to which microRNA play a central role. Secondly, it introduces two of the most important categories of features that describe RNAs in general and microRNAs in particular: thermodynamical and topological features. Section 2 makes a gentle introduction to SVMs. Both linear (with hard and respectively soft margin) and non-linear cases are treated. Section 3 presents most of the existing SVMs that have been devised for microRNA identification. Other features than those presented in Section 1 are introduced "on fly". Section 4 sketches further work.

1 MicroRNAs

1.1 Background: RNA interference

MicroRNA sequences, henceforth abbreviated miRNAs, are an important subclass of the so-called *non-codant RNAs*, i.e. RNAs that do not serve for protein synthesis. MiRNAs serve as main players into a very important process called *RNA interference* (RNAi), also known as RNA-mediated interference [17].

RNAi constitutes a major exception to the *central dogma of molecular biology* that is usually simply stated as athree stage molecular process:

$$\text{DNA} \overset{transcription}{\longrightarrow} \text{RNA} \overset{translation}{\longrightarrow} \text{proteins.}$$

RNAi blocks or reduces the DNA expression at post transcriptional level, namely by degradation of messenger RNA (mRNA) or the inhibition of mRNA translation into proteins.

The molecular genesis of miRNA comprises the following steps:

– miRNA is produced in the nucleus as a sequence of around thousand nucleotides long called *primary miRNA* (pri-miRNA);
– a *microprocessor* (Drosha and its Pasha companion) protein complex cuts away from the primary miRNA a 70–130 nucleotide sequence having a hairpin (stem-loop) structure, which is called the *precursory* miRNA (pre-miRNA). Then a transporter protein complex (Exportin 5) carries pre-miRNA into the cytoplasm;
– a Dicer protein complex cuts away the pre-miRNA's loop. Afterwards, the resulting double strand is un-wound. One strand of 19–23 nucleotides, the *mature miRNA*, is further on incorporated into the RISC protein complex (RNA-Induced Silencing Complex);
– the RISC complex attaches the mature miRNA to a target mRNA sequence, subsequently provoking either the mRNA's cleavage or the obstruction of mRNA's translation into protein.

The first miRNA discovered was lin-4 [22]. Larval development of C. elegans depends on lin-14, a nuclear protein. In later development, the lin-4 miRNA completely suppresses the translation of lin-14 mRNA. An example of human miRNA is hsa-mir-16-1 from chromosome 13. This miRNA together with its companion (mir-15a) have been proved to be deleted or downregulated in more than two thirds of cases of chronic lymphocytic leukemia [37].

RNAi, the process in which miRNAs are involved, is a very important cellular mechanism. It was first explained by Andrew Fire and Craig Mello [12], a discovery for which they were awarded the Nobel prize in medecine and physiology in 2006. Here we list some of RNAi's main implications:

– transcription regulation: RNAi participates in the control of the amount of mRNA existent in the cell;
– protection from viruses: RNAi blocks the multiplication of viral RNA, and as such plays an import part in the organism's immune system;
– RNAi may serve to identify the function of virtually any gene, by knocking down/out the corresponding mRNA. In recent projects, entire libraries of short interfering RNAs (siRNAs) are created, aiming to silence each gene of a chosen model organism;
– therapeutically, RNAi may help researchers design drugs for cancer, tumors, HIV, and other diseases.

Here follow several *specific facts* on miRNAs (for more details see [37] [3] [8]) that can help researchers to design miRNA identification systems based on machine learning (ML):

– Primary miRNAs can be located in introns of protein-coding regions, exons and introns of non-coding regions, or intergenic regions.
– The stem-loop structure of a pre-miRNA should have a low free energy level in order to be stable [4].
– Many miRNAs are conserved across closely related species (but there are only few universal miRNAs), therefore many prediction methods for miRNAs use genome comparisons. The degree of conservation between orthologuos miRNAs is higher on the mature miRNA subsequence than on the flanking regions; loops are even less conserved.
– Conservation of miRNA sequences (also its length and structure) is lower for plants than it is for animals. In viruses, miRNA conservation is very low. Therefore miRNA prediction methods usually are applied (or, are tuned) to one of these three classes of organisms.
– MiRNAs tend to be situated in clusters, within a few kilobases. The miRNAs situated in a same cluster can be transcribed together.
– A highly conserved motif (with consensus CTCCGCCC for *C. elegans* and *C. briggsae*) may be present within 200 base-pairs upstream the miRNA clusters.
– Identification of MiRNA *target sites* is easy to be done for plants (once miRNA genes and their mature subsequence are known) but is more complicated for animals due to the fact that usually there is an imperfect complementarity between mature miRNAs and their targets.

The identification of new miRNAs is complicated by the fact that usually there are millions of hairpin RNAs expressed by a genome, while the number of miRNAs is only slightly above one thousand. This fact usually results in a quite high number of false positives (therefore low specificity) for the ML systems that were designed to identify miRNAs.

1.2 RNA thermodynamical and topological features

1.2.1 Basics: Computing RNA minimum free energy

Unlike DNA sequences which form the well known, simple double stranded helix, RNA form quite sophisticated structures. The main categories of the secondary (2D) structure elements of RNAs are

– stem: a double strand formed by a succession of complementary base-pairs (A-U, C-G)
– loop: short sequences of unpaired nucleotides that usually end a stem
– multi-loop: sequences of unpaired nucleotides connecting three or more stems
– bulges and wobbles: irregularities that occur on stems, due to a succession of non-complementary nucleotides on the two opposite strands; the nucleotides that form a bulge appear on only one strand of a stem.

We will not work here with pseudo-knots (a complicated sort of RNA secondary structure elements), therefore given an RNA sequence $x = x_1x_2 \dots x_n$, for any two base pairs (x_{i_1}, x_{j_1}) and (x_{i_2}, x_{j_2}) in the secondary structure of x such that $i_1 < j_1$ and $x_2 < j_2$, the following relation holds: $x_{i_1} < x_{i_2} < x_{j_2} < x_{j_1}$ or $x_{i_2} < x_{i_1} < x_{j_1} < x_{j_2}$.

Predicting the secondary structure of an RNA sequence is a long-time studied problem, and still it is not entirely satisfactory solved. The most widely used approach is based on molecular thermodynamics and it employs a simple assumption: given an RNA sequence x, among all its possible secondary structure the most stable one is that for which the minimum free energy (MFE) is reached. This approach was proposed by Zuker and his colleagues in the beginning of 1980s. Prior to this, a simple approach was proposed by Nussinov and his colleagues (1978). They determined the secondary structure by computing the maximum number of complementary base-pairs.

A simple algorithm for RNA folding: Nussinov

Initialisation: $S(i, i-1) = 0$ for $i = 2$ to L and $S(i, i) = 0$ for $i = 1$ to L

Recurrence: $$S(i,j) = \max \begin{cases} S(i+1, j-1) + 1 \text{ if } [\ i, j \text{ base pair }] \\ S(i+1, j) \\ S(i, j-1) \\ \max_{i<k<j}\{S(i,k) + S(k+1, j)\} \end{cases}$$

Output: $S(1, L)$

It is a dynamic programming algorithm. Given an RNA sequence $x_1x_2 \dots x_n$, the (i, j) element in the $n \times n$ matrix S will store the maximum number of complementary base-pairs that can be formed by the nucleotides in the sequence, allowing no pseudo-knots. The attentive reader will see that the initialisation relation $S(i, i-1) = 0$ for $i = 2$ to L is required for the smooth formulation of the first case of the recurrence relation.[1] The recurrence relation itself can be easily understood. The filling of the matrix S can be done by computing the elements on the successive parallels to the main diagonal. The maximum number of base pairs will be retrieved in $S(1, n)$, and the optimal structure itself can be constructed via backtracing. The time complexity of the Nussinov algorithm is $\mathcal{O}(n^3)$.

The main idea behind Zuker's algorithm is that each base pair added to the secondary structure of a given RNA lowers its free energy, while irregularities (loops, bulges, wobbles and multi-loops) increase the free energy. The specific amounts of free energies corresponding to these cases are the estimated by very elaborate laboratory measurements.

Zuker's algorithm recursively computes

- $W(i, j)$: MFE of all non-empty foldings of the subsequence $x_i, \dots, x_j$

[1] Take for instance the case of $i = 2, j = 3$.

– $V(i,j)$: MFE of all non-empty foldings of the subsequence $x_i, \ldots, x_j$, containing the base pair (i,j)

The matrices V and W will be (simultaneously) filled in the same manner as indicated for the Nussinov algorithm, namely by successively computing the parallels to the main diagonal..

Zuker algorithm

Initialization: $W(i,j) = V(i,j) = \infty$ for all i,j with $j-4 < i < j$.

Recurrence: for all i,j with $1 \leq i < j \leq n$

$$V(i,j) = \min \begin{cases} eh(i,j) \\ es(i,j) + V(i+1,j-1) \\ VBI(i,j) \\ VM(i,j) \end{cases}$$

$$W(i,j) = \min \begin{cases} V(i,j) \\ W(i+1,j) \\ W(i,j-1) \\ \min_{i<k<j}\{W(i,k) + W(k+1,j)\} \end{cases}$$

where

$$VBI(i,j) = \min_{\substack{i < i' < j' < j \\ i' - i + j - j' > 2}} \{ebi(i,j,i',j') + V(i',j')\}$$

$$VM(i,j) = \min_{i<k<j-1}\{W(i+1,k) + W(k+1,j-1)\} + a$$

with a being a constant energy contribution to close the multi-loop.

The values of the parameters

– $eh(i,j)$ – the energy of the hairpin closed by the pair (i,j)
– $es(i,j)$ – the energy of the stacked pair (i,j) and $(i+1,j-1)$
– $ebi(i,j,i',j')$ – the energy of the bulge or interior loop that is closed by (i,j), with the pair (i',j') *accessible* from (i,j) (i.e., there is no base pair (k,l) such that $i < k < i' < l < j$ or $i < k < j' < l < j$)

are taken from tables.

1.2.2 Other thermodynamical features

The following notions are from [Freyhult et al., 2005]. Given an RNA sequence x whose length is L, we define

• The *adjusted MFE* of x:

$$dG(x) = \frac{MFE(x)}{L}$$

Note that $dG(x)$ removes the bias that a long sequence tends to have a lower MFE.

• The *MFE Index 1:* the ratio between $dG(x)$ and the percentage of the $G+C$ content in the sequence x.
• The *MFE Index 2:*

$$\frac{dG(x)}{S}$$

where S is the number of stems in x that have more than three contiguous base-pairs.
• *Z-score*: the number of standard deviations by which $MFE(x)$ differs from the mean MFE of $X_{shuffled}\,(x)$, a set of shuffled sequences having the same dinucleotide composition as x:

$$Z(x) = \frac{MFE(x) - E(MFE(x') \;:\; x' \in X_{shuffled}\,(x))}{\sigma(MFE(x') \;:\; x' \in X_{shuffled}\,(x))}$$

Note: See the *Altschul-Erikson* algorithm (1985) for sequence shuffling.
• *P-value:* the number of sequences in $X_{shuffled}\,(x)$ having a MFE level lower than the MFE of x.

$$\frac{\mid \{x' \in X_{shuffled}\,(x) \;:\; MFE(x') < MFE(x)\} \mid}{\mid X_{shuffled}\,(x) \mid}$$

• The *adjusted bas-pairing propensity* $dP(x)$: the average number of base pairs in the secondary structure of x. It removes the bias that longer sequences tend to have more base-pairs.
• The *adjusted Shannon entropy:*

$$dQ(x) = \frac{-\sum_{i<j}\; p_{ij} \log_2(p_{ij})}{L}$$

where p_{ij} is the probability that x_i and x_j is a base-pair in x:

$$p_{ij} = \sum_{S_\alpha \in \mathcal{S}(x)} P(S_\alpha)\, \delta_{ij}^{\alpha}$$

and in turn $\mathcal{S}(x)$ is the set of all secondary structures corresponding to x, while the value of δ_{ij}^{α} is 1 if x_i and x_j is a base-pair in the structure S_α and 0 otherwise. The probability of $S_\alpha \in \mathcal{S}(x)$ follows a Boltzmann distribution:

$$P(S_\alpha) = \frac{e^{-MFE_\alpha/RT}}{Z}$$

with

$Z = \sum_{S_\alpha \in \mathcal{S}(x)} e^{-MFE_\alpha/RT}$,
$R = 8.31451$ Jmol$^{-1}K^{-1}$ (a molar gas constant), and
$T = 310.15$K (37° C)

Note: Low values of dQ indicate that *i.* one or a few base-pairs are dominant in the RNA's structure(s), or *ii.* there are no base-pairs at all.

- The *adjusted base-pair distance* (or *ensemble diversity*):

$$dD(x) = \frac{\frac{1}{2}\sum_{S_\alpha, S_\beta \in \mathcal{S}(x)} P(S_\alpha)P(S_\beta)d_{BP}(S_\alpha, S_\beta)}{L}$$

where $d_{BP}(S_\alpha, S_\beta)$, the base-pair distance between two structures S_α and S_β of the sequence x, is defined as the number of base-pairs not shared by the structures S_α and S_β:

$$d_{BP}(S_\alpha, S_\beta) = \mid S_\alpha \cup S_\beta \mid - \mid S_\alpha \cap S_\beta \mid = \mid S_\alpha \mid + \mid S_\beta \mid -2 \mid S_\alpha \cap S_\beta \mid .$$

Because $\mid S_\alpha \mid = \sum_{i<j} \delta_{ij}^{\alpha}$, we get $d_{BP}(S_\alpha, S_\beta) = \sum_{i<j}(\delta_{ij}^{\alpha} + \delta_{ij}^{\beta} - 2\delta_{ij}^{\alpha}\delta_{ij}^{\beta})$, and following a quite straightforward calculus (See [Freyhult et al., 2005]) we arrive at a simpler form for dD:

$$dD(x) = \frac{\sum_{i<j}(p_{ij} - p_{ij}^2)}{L}$$

Note: The probabilities p_{ij} are efficiently computed using the algorithm presented in [McCaskill, 1990].

1.2.3 RNA topological features

Following [14], we will present tree graphs and dual graphs that give a coarse but useful approximation of the RNA secondary structure. Each RNA sequence that has no pseudo-knots can be assigned a *tree graph* that is constructed using the following rules:

- each bulge, hairpin loop or wobble ("internal loop") having at least two unmatched nucleotides or one non-complementary base pair constitutes a vertex in the tree graph;[2]
- the 3' and 5' ends of a stem are assigned (together) a vertex;
- a multi-loop ("junction"), which is the location at which at least three stems meet, is a vertex;

RNAs that have pseudo-knots (and also those that do not have pseudo-knots) can be assigned *dual graphs*:

- a vertex is a double stranded stem that has at least two complementary base pairs;
- an edge is a single strand that connects secondary structure elements (bulges, wobbles, loops, multi-loops and stems).[3]

[2] Special case: for wobbles, GU is considered a complementary base pair.

[3] For bulges, there should be at least two unmatched nucleotides or one non-complementary base pair, as in the definition of the tree graph.

The reader should notice that it is possible that two distinct RNAs map onto the same (tree or dual) graph.

Using the spectral techniques in graph theory [25] one can quantitatively characterize the tree graphs and dual graphs assigned to RNAs. Let G be an unoriented graph, possibly having loops and multiple edges. We will use the following notations:

- $A(G)$ is the *adjiacency matrix* of the graph G: a_{uv} is the number of edges between vertices u and v;
- $D(G)$ is the *degree matrix* of G: $d_{uv} = 0$ for $u \neq v$ and $d_{uu} = \sum_v a_{uv}$;
- $L(G) = D(G) - A(G)$ is called the *Laplacian matrix* of the graph G;
- $L(G)X - \lambda X$ is named the *characteristic polynomial* of the matrix $L(G)$. Its roots $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ are called the Laplacian eigenvalues of G, where $n =\mid V(G) \mid$ denotes the number of vertices in G. The tuple $(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is called the *spectrum* of G and it is independent of the labelings of graph vertices.

It can be proved that $\lambda_1 = 0$ and $\lambda_2 > 0$ if and only if the graph G is connected, and also graphs with resambling topologies have closed λ_2 values. Thus λ_2 can be used as a measure of similarity between graphs, and some authors call it *graph connectivity*. The miPred SVM [27] used this similarity measure on tree graphs and dual graphs, together with thermodynamical and mono- and di-nucleotide frequency features, to distinguesh between miRNAs and pseudo-miRNAs.

2 Support Vector Machines: An introduction

SVM is a tool for solving classification problems which was designed by Vladimir Vapnik and his collaborators [9]. Here we view classification as a machine learning technique.[4] The following subsection gives the reader the possibility to be acquainted with the basic notions in machine learning based classification. For a detailed introduction to the field of machine learning, the reader should refer to Tom Mitchell's book [24].

2.1 Basic machine learning notions

The *classification problem*can be simply formulated as follows:

A *concept* c is a subset of R^d. Automatically learning to classify a given *instance* $x \in R^d$ with respect to the concept c requires two steps: the first one is *training*, while the second one is named *generalisation*, *query*, or *test*. For both tasks we assume that a number of instances are selected from R^d according to a certain probabilistic distribution $\mathcal{D}$.

Let us assume that the *training instances* (also called *examples*) form a countable set $S \subseteq R^d$. Each $x_i \in S$ is associated a label $y_i \in \{-1, +1\}$ such that

[4] Classification is supervised learning. Unsupervised learning is known as *clusterization*.

$y_i = +1$ iff $x_i \in c$.[5] The objective of the training phase is to produce a *hypothesis* $h : R^d \to \{-1, +1\}$ that approximates the concept c.

According to the Probabil Approximative Correct (PAC) model [33] that constitutes an important part of the mathematical set-up for SVMs, an *approximately correct hypothesis* h for the concept c is one for which

$$P(\{x \mid h(x) = -1 \text{ and } x \in c, \text{ or } h(x) = +1 \text{ and x } \notin c\}) < \epsilon$$

where ϵ is a certain (chosen) positive value, and $P : 2^{\Omega} \to [0; 1]$ is the probability function used in the definition of the distribution $\mathcal{D}$.[6]

We say that the *hypothesis* h is *consistent* with the concept c on the training set S iff $h(x_i) = y_i$ for any training instance $x_i \in S$ whose associated label was y_i. The set made of all such consistent hypothesis is called the *space version* of the concept c. More precisely, as the search is done in a certain hypothesis space H, assuming that the set of training instances is S, the *space version* is denoted $VS_{H,S}$.

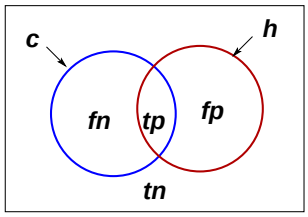
It is common practice to evaluate the quality of a learned hypothesis h by checking it against a subset $V \subseteq S$, assuming that the training was performed on the set $S \backslash V$ instead of S. The set V is called *validation set*.

The following *statistic measures* are widely used for evaluation:

$$\textit{accuracy:} \quad Acc = \frac{tp + tn}{tp + tn + fp + fn}$$

$$\textit{precision:} \quad P = \frac{tp}{tp + fp}$$

$$\textit{recall:} \quad R = \frac{tp}{tp + fn}$$



where tp is the number of true positives, fp the number of false positives, tn the number of true negatives, fn the number of false negatives produced by the hypothesis h on the validation set V. Formally, true positives are those x for which $h(x) = 1$ and $x \in c$, false positives are characterized by $h(x) = 1$ and $x \notin c$, for true negatives $h(x) = 0$ and $x \notin c$, while for false negatives $h(x) = 0$ and $x \in c$.

A measure that combines precision and recall in a single quantity is the so-called *F-measure*:

$$\textit{F-measure:} \quad F = \frac{2PR}{P + R}$$

[5] This is the case of noise-free learing. In a simple acception, the notion of *noise* refers to the possiblity of associating wrong labels y_i to instances x_i. While in practice it is generally rare the case that the labeling of training instance is error-free because of human and/or machine errors, for the sake of clarity here we restrict ourselves to noise-free automate learning.

[6] $\mathcal{D} : R^n \to [0; 1]$ and $\mathcal{D}(x) = P(\{\omega \mid X(\omega) = x\})$, where $X : \Omega \to R^n$ is a random variable, and $P : 2^{\Omega} \to [0; 1]$. In the above set-up, Ω is R^d. If $\mathcal{D}$ is the normal (Gaussean) distribution, then $n = 1$.

In bioinformatics, the recall measure is usually called *sensitivity* (Se), while instead of precision the *specificity* measure is preferred:

$$Sp = \frac{tn}{tn + fp}$$

Machine learning methods always incorporate certain specific biases that limit or guide the search for good hypotheses h among the whole set of plausible hypotheses H. The learning biases for SVMs is the subject of the next subsection.

2.2 The learning biases for SVMs

SVMs were designed with two main *biasis* in mind. The *first bias* consists in searching (only) for linear hypotheses $h(x) = w \cdot x + w_0$, where the $\cdot$ operator represents the dot (scalar) product of the vectors w and $x \in R^n$. We will show that searching for linear hypotheses is appropriate not only in the case when the training set S is linearly separable (i.e. there is indeed a hyperplane in R^n that separates positive instances from negative ones), but also when S is not. In the latter case, instances $x \in R^n$ will be mapped into a larger space R^m, with the hope that a separating hyperplane will be found there. That hyperplane would correspond to a non-linear separating curve in the origin space R^n, and that curve will represent the *learned form* of the concept c.

The *second bias* in the design of SVMs is that in order to minimize the risk that test instances x would be misclassified, among all existing linear hyperplines that separate positive from negative instances, an *optimal separating hyperplane* $h(x) = 0$ must be chosen, so to maximize the distance to (each of) the training instances that are the closest ones with respect to the separating hyperplane.

A more detailed discussion of these points will follow in the next subsections.

2.3 Linear Support Vector Machines

In the above set-up, assuming a finite training set $S = \{x_1, x_2, \ldots, x_m\}$ that is linear separable, a hyperplane $(w, w_0) \in R^d \times R$ that separates the positive from the negative examples has the property $y_i(w \cdot x_i + w_0) > 0$ for $i = 1, \ldots, m$ Obviously, if (w, w_0) is a separating hyperplane for S, then $(\alpha w, \alpha w_0)$, with $\alpha \in R^+$ has the same property. Therefore, after eventually re-naming w and w_0 we can assume

$$y_i(w \cdot x_i + w_0) \geq 1 \text{ for } i = 1, \ldots, m$$

The signed distance from x_i to the hypeplane (w, w_0) is $d_i = (w \cdot x + w_0)/ \mid\mid w \mid\mid$, and consequently $y_i d_i \geq 1/ \mid\mid w \mid\mid$ and $\mid d_i \mid \geq 1/ \mid\mid w \mid\mid$. Searching for an *optimal separating hyperplane*, i.e. one that is as distant as possible from the closest points $x_i \in S$ to it, will imply maximizing $1/ \mid\mid w \mid\mid$, which is equivalent to minimizing $\mid\mid w \mid\mid$.

To summarize, given S, the search for an optimal hyperplan separating its elements amounts to solving the following quadratic constraint problem:

$$\begin{array}{l} \textit{minimize } \frac{1}{2}||w||^2 \\ \textit{subject to } y_i(w \cdot x_i + w_0) \geq 1 \text{ for } i = 1, \ldots, m \end{array}$$

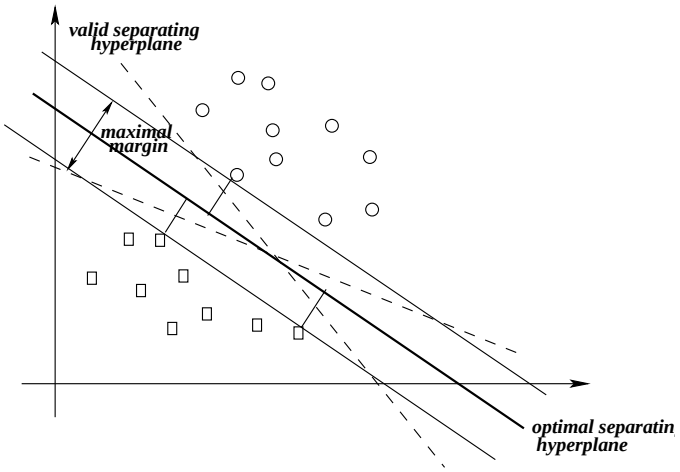


Fig. 1. Illustrating the biasis behind the design of SVMs. Circles represent positive examples, while squares represent negative examples.

This is called the *primal form* of linear SVM. Such a quadratic constraint problem can be solved efficiently by numeric methods if d is in the order of thousands (remember that $S \subseteq R^d$).

In practice it is often the case that d is much larger, therefore we should proceed to a further step: since both the optimisation criterion and the constraints in the primal form of the linear SVM problem are convex, we can apply the Kuhn-Tucker theorem [13] and transform the above problem into an equivalent, *dual form*:

$$\begin{aligned} &\textit{maximize } \textstyle\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \; x_i \cdot x_j \\ &\textit{subject to } \textstyle\sum_{i=1}^{m} y_i \alpha_i = 0 \\ &\qquad\qquad \alpha_i \geq 0, i = 1, \ldots, m \end{aligned}$$

The newly introduced parameters α_i for $i = 1, \ldots, m$ are called *Lagrangean multipliers.*[7]

[7] For those readers who are interested in mathematical details, we sketch the link between the primal and the dual forms of the linear SVM problem:

Instead of minimizing $\frac{1}{2} \mid\mid w \mid\mid$, the dual problem aims to minimizing the *Lagrangean function*

$$L_P(w, w_0, \alpha) = \frac{1}{2}||w^2|| - \sum_{i=1}^{m} \alpha_i (y_i(w \cdot x_i + w_0) - 1)$$

The link between the *solutions* of the two forms of the linear SVM problem is given by the following relations:

$$\overline{w} = \sum_{i=1}^{m} \overline{\alpha}_i y_i x_i$$

$$\overline{\alpha}_i(y_i(\overline{w} \cdot x_i + \overline{w}_0) - 1) = 0 \text{ for any } i = 1, \ldots, m$$

where $(\overline{w}, \overline{w}_0)$ is the optimal solution of the primal problem and $\overline{\alpha} = (\overline{\alpha}_1, \ldots, \overline{\alpha}_m$ is the optimal solution of the dual problem.

The reader should note that $\overline{\alpha}_i$ can be nonzero only in if the constraint $y_i(w \cdot x_i + w_0) \geq 1$ is satisfied with the equality sign. The points x_i for which $\alpha_i \neq 0$ are the closest points to the optimal separating hyperplane (OSH). They are called *support vectors* because they are the only elements of S needed to determine the OSH. Indeed, as shown by the above relations, the vector $\overline{w}$ is a linear combination of these x_i.

The problem of *calssifying* a new data point x amounts to simply looking at $sign(\overline{w} \cdot x + \overline{w}_0)$.

2.4 Linear SVMs with soft margin

When the training set $S = \{x_1, \ldots, x_m\}$ is not linearly separable or we simply ignore whether or not it is linearly separable, we introduce m non-negative variables ξ_i, for $i = 1, \ldots, m$, subject to the inequalities

$$y_i(w \cdot x_i + w_0) \geq 1 - \xi_i, \text{ for } i = 1, \ldots, m$$

The *generalised OSH* is then viewed as the solution to the (primal) problem:

$$\begin{aligned} &\textit{minimize } \tfrac{1}{2}||w||^2 + C \textstyle\sum_{i=1}^{m} \xi_i \\ &\textit{subject to } y_i(w \cdot x_i + w_0) \geq 1 - \xi_i \text{ for } i = 1, \ldots, m \\ &\qquad \xi_i \geq 0 \text{ for } i = 1, \ldots, m. \end{aligned}$$

Finding its minimum implies the following constraints on its partial derivatives:

$$\frac{\partial L}{\partial w_0} = -\sum_{i=1}^{m} y_i \alpha_i = 0$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} y_i \alpha_i x_i = 0 \Rightarrow w = \sum_{i=1}^{m} y_i \alpha_i x_i$$

$$\text{where } \frac{\partial L}{\partial w} = (\frac{\partial L}{\partial w_1}, \ldots, \frac{\partial L}{\partial w_d}).$$

By substituting these constraints into L_P we get its dual form

$$L_D(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \; x_i \cdot x_j$$

L_D will constitute the optimisation criterion in the dual form of the linear SVM problem.

The associated dual form is:

$$\begin{aligned} &\textit{maximize } \sum_{i=1}^{m} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \; x_i \cdot x_j \\ &\textit{subject to } \sum_{i=1}^{m} y_i \alpha_i = 0 \\ &\qquad 0 \leq \alpha_i \leq C, i = 1, \ldots, m \end{aligned}$$

As before, there is a link between the solutions of the two forms of our classification problem:

$$\begin{aligned} \overline{w} &= \sum_{i=1}^{m} \overline{\alpha}_i y_i x_i \\ \overline{\alpha}_i (y_i (\overline{w} \cdot x_i + \overline{w}_0) - 1 + \overline{\xi}_i) &= 0 \\ (C - \overline{\alpha}_i) \, \overline{\xi}_i &= 0 \end{aligned}$$

Here above C acts as a *regularizing parameter*: large C leads to minimizing the number of misclassified points, while small C maximizes the minimum distance $1/||w||$ to correctly classified instances.[8]

If $\bar{\alpha}_i < C$, then $\bar{\epsilon}_i = 0$, therefore x_i is correctly classified, and the distance from x_i to the *OSH* is $\geq 1/||w||$. If $\bar{\alpha}_i = C$, then

- $\bar{\epsilon}_i > 1$, meaning that x_i is misclassified;
- $0 < \bar{\epsilon}_i \leq 1$, therefore x_i is correctly classified but it is closer than $\geq 1/||w||$ from the *OSH*;
- $\bar{\epsilon}_i = 0$, which is a rare case.

2.5 Non-linear SVMs and kernel functions

When S, the set of training examples is non-lineraly separable, instead of searching for higher-order separators (e.g. polynomials or hyperbolic functions), the SVM's main inventor, Vladimir Vapnik, chosed a surprisingly simple and successful idea: search for a function $\Phi : R^d \rightarrow R^n$ that maps the examples $x_1, x_2, \ldots, x_m$ into a higher-order space ($n > d$) in which the quest for a linear hyperplane that optimally separates $\Phi(x_1), \Phi(x_2), \ldots, \Phi(x_m)$ is successful. The linear separator in R^n will usually correspond to a non-linear separator of the training examples in R^d. (See the example given below, and illustrated in Figure 2.5.)

In this new set-up, the same formulation for the SVM problem in used as in subsection 2.4, except for replacing x_i with $\Phi(x_i)$. The only one drawback that appears is that the dot product $\Phi(x_i) \cdot \Phi(x_j)$ is computationally expensive. This inconvenient may be solved if we could find a function $K : R^d \times R^d \rightarrow R$ such that $K(x, y) = \Phi(x) \cdot \Phi(y)$ for all $(x, y) \in R^d$. In such a case, the expensive dot product in R^n is replaced by a calculus in the lower dimensional space R^d. Such functions K are called *kernel functions*.

[8] Alternatively, one may specify the fraction of training examples that can be misclassified by the SVM, and the corresponding value of C can be calculated subsequently. This is the approach followed by Thorsten Joachims in one of the implementation SVM^{light} [21].

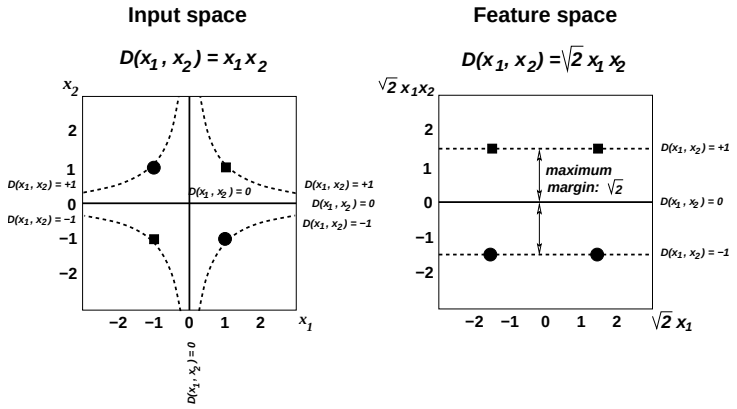


Fig. 2. A non-linearly separable training set (that corresponds to the negation of the boolean operation XOR), and the optimal hyperplane found by an SVM.

Example: Consider XOR, the binary *exclussive or* operation, and the set of examples $S = \{(1,1),(1,-1),(-1,1),(-1,-1)\}$ with the labeling $y_i = x_i^1 \neg\text{XOR } x_i^2$ for each $x = (x_i^1, x_i^2) \in S$. S is not linearly separable in R^2. Let $\Phi(x)$ be a mapping of each $x = (x_i^1, x_i^2)$ to an array $\Phi(x) = (z^1, z^2, \ldots, z^n) \in R^n$. If the definition of Φ is such that, for j be a certain index in $\{1, \ldots, n\}$ and a constant $c \neq 0 \in R$, the following simple restriction is satisfied

$$z^j = cx_i^1 x_i^2 \text{ for every } x = (x^1, x^2)$$

then in the space R^n the hyperplane D defined by the equation $z^j = 0$ is a linear separator for the set $\Phi(S) = \{\Phi(x) \mid x \in S\} \subseteq R^n$, and it corresponds to the non-linear separator $x^1 x^2 = 0$ in the original space (R^2).

For instance, the function $\Phi(x) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]$ where $x = (x_1, x_2)$, satisfies the condition mentioned above. The interested reader can verify that for every x and y in R^2, $\Phi(x) \cdot \Phi(y) = K(x,y)$, where $K(x,y) = (x \cdot y + 1)^2$, a simple function that plays the role of a kernel.

Moreover, it can also be shown in this case that D is an optimal linear separator for $\Phi(S)$, and the four given training instances in R^2 are all support vectors.

Several simple classes of kernel functions exist: polynomials $K(x,x') = (x \cdot x' + c)^q$, radial basis functions (RBF) $K(x,x') = e^{-\frac{||x-x'||^2}{2\sigma^2}}$ and sigmoid functions like $K(x,x') = tanh(\alpha x \cdot x' - b)$. The interested reader can easily find the appropriate Φ functions in each case of interest.

In practice, given a certain classification problem, instead of trying different Φ functions, the SVM user will try different kernel functions as a parameter (along with the regularisation parameter C), leaving Φ unspecified.

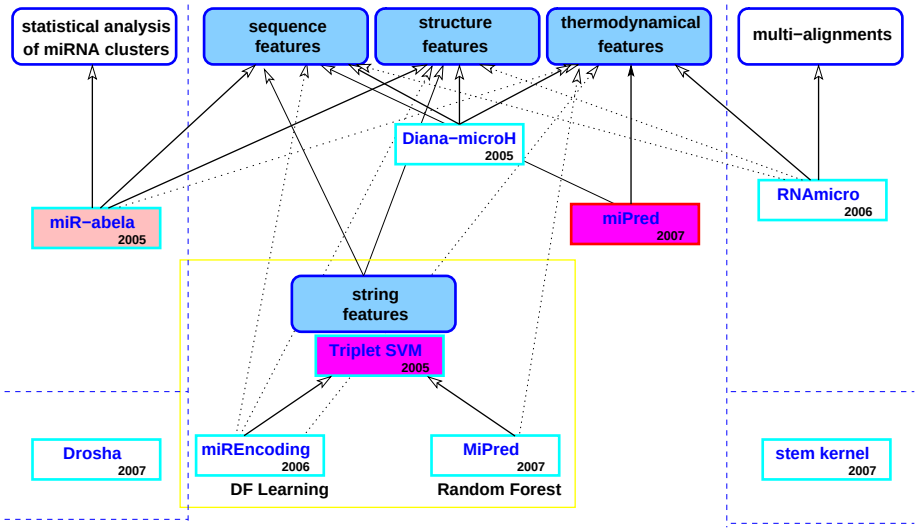


Fig. 3. An overview of SVMs for miRNA identification.

Several free implementations of SVM exist, among which the most widely used are SVM^{light} by Thorsten Joachims [21], and LIBSVM by Chang and Lin [23].

We mention that kernel functions are used not only in conjunction with SVMs but also with other algorithms like PCA (principal component analysis) from the field of data mining.

3 SVMs for microRNA identification

Here we will present the main existing systems for the identification of miRNAs which are based on the SVM classifier [9] [11]. Since not all of these systems are trained on the same data sets, comparing their classification/generalisation performances is not obvious. However, what basically makes the difference among such systems is the set of features that they employ. This is what decides their individual performances, assuming that the same training dataset is used and that search space for the best values of the generic SVM parameters are well explored. This is why here our aim will be to present the main alternatives/ways in which the feature sets of the SVM-based miRNA classification systems were designed.

Triplet-SVM by Xue et al. [36] basically uses a (enhanced spectrum) string kernel to identify *ab initio* miRNAs. The defined string features combine first and second level structure informations on 3-mers. Trained on human miRNAs from the miRNA registry database [1] [15], the triplet-SVM system is reported

to achieve around 90% accuracy in distinguishing real from pseudo[9] miRNA hairpins in the human genome and other 11 species, including *C. briggsae*, *C. elegans*, *D. pseudoobscura*, *D. melanogaster*, *Oryza sativa*, *A. thaliana* and the *Epstein Barr* virus.

MiREncoding [38] implemented an SVM which in addition to triplet-SVM's "local" features added a number of "global" features:

- sequence length,
- GC content,
- number of paired bases,
- length basepair ratio (length of the sequence / the number of basepairs),
- central loop length,
- symmetric difference (the difference of length of the two arms),
- number of bulges,
- average bulge size,
- number of tails,
- average tail size,
- free energy per nucleotide.

Tested on 15 data sets, miREncoding obtained an overall 4% accuracy gain over triplet-SVM, and reported a specificity of 93.3% at 92% sensitivity. Using a feature selection algorithm called Discrete Function Learning [40] (after having discretised the continuous attributes), the authors identified the "essential" (i.e. most discriminative) features for the miRNAs classification among all the used ones. DFL finds that using only four such essential features — A(((, G.((, length basepair ration, and energy per nucleotide — greatly improves the classification done by the C4.5, kNN and RIPPER algorithms. However, in general MiREncoding SVM performed better when using all attributes. Remarkably, on several test data sets used by miREncoding's authors, the classification performances of C4.5 [ ], kNN [ ] and RIPPER [ ] on the essential (DFL-selected) feature set are better than those obtained by the SVM on the full feature set.

The authors of the MiPred system [20] also improved on triplet-SVM but they added two global features: minimum free energy (MFE) and the so-called P-value and then they replaced the SVM with the random forest (RF) classifier [6], a machine learning technique which is an extension of the bagging algorithm [5]. Quite surprisingly, RF behaves better than SVM on classifying miRNAs. Trained on the same data set as triplet-SVM, the MiPred system reported 98.21% specificity at 95.09% sensitivity, and an improvement on triplet-SVM's accuracy from 83.9% to 91.29%.

DIANA-microH [32] uses as features the minimum free energy, three of the global features employed by miREncoding — the number of based pairs, central

[9] *Pseudo miRNA* hairpins are defined as RNA hairpins whose stem length and minimum free energy are in the range of those exhibited by the genuine, *real miRNAs*.

Features over the *entire hairpin* structure:
1 free energy of folding
2 length of the longest simple stem
3 length of the hairpin loop
4 length of the longest perfect stem
5 number of nucleotides in symmetrical loops
6 number of nucleotides in asymmetrical loops
7 average distance between internal loops
8 average size of symmetrical loops
9 average size of asymmetrical loops
10-13 proportion of A/C/G/U nucleotides in the stem
14-16 proportion of A-U/C-G/G-U base pairs in the stem

Features over the *longest "symmetrical" region* of the stem:
17 length
18 distance from the hairpin loop
19 number of nucleotides in internal loops
20-23 proportion of A/C/G/U nucleotides
24-26 proportion of A-U/C-G/G-U base pairs

Features over the *relaxed symmetrical region*:
27 length
28 distance from the hairpin loop
29 number of nucleotides in symmetrical internal loops
30 number of nucleotides in asymmetrical internal loops
31-34 proportion of A/C/G/U nucleotides
35-37 proportion of A-U/C-G/G-U base pairs

Features over all *windows of lengths* l_m, the (assumed) length of mature miRNA:
38 maximum number of base pairs
39 minimum number of nucleotides in asymmetrical loops
40 minimum asymmetry over the internal loops in this region

Fig. 4. miR-*abela* features.

loop length, GC content — plus one new global feature, *stem linearity*[10] and one evolutionary based feature, *arm conservation* (computed using human vs. rat or human vs. mouse sequence comparisons). Trained on the miRNAs from the human miRNA repository, the authors claim a 98.6% accuracy following 5-fold cross-validation.

[30] uses a SVM (identified as miR-*abela* in [20]) based on a bunch of non-string features (see Figure 4), that the authors have grouped into four categories:

– features over the entire hairpin structure (16 features),

[10] It is defined as "the largest possible section of the stem subregion that is likely to form a mostly double-stranded conformation".

– features over the longest "symmetrical" region of the stem (10 features),[11]
– features over the relaxed symmetrical region (11 features),[12]
– features over all windows of lengths equal to l_m, the (assumed) length of mature miRNA (3 features).[13]

miR-*abela* was trained on 178 human pre-miRNAs as positive examples and 5395 randomly chosen sequences (from genomic regions, tRNA, rRNA and mRNA) as negative examples. The scores produced by this SVM for robust pre-miRNA candidates[14] in a given genomic region are further fed into a statistical model that estimates the number of genuine pre-miRNAs in that region. miR-*abela*'s output was experimentally validated in the laboratory. Out of 32 pre-miRNA predictions made by miR-*abela* on the genomes of eight human pathogenic viruses, 13 were confirmed by the cloning study. Similarly, 68 out of 260 predictions of new pre-miRNAs for human, mouse and rat were experimentally confirmed.

[18] proposed two SVMs. The first one, called Microprocessor SVM, is intended for the recognition of Drosha cutting sites on sequences that are presumed to extend pre-miRNA sequences (and at the same time, are substrings of primary miRNAs). For a given hairpin, the system proposes a bunch of *processing site candidates* for the Drosha complex. Such a site is the 5' end of a 50-80nt sequence centered around a stem loop. (The 3' end is determined by a 2nt overhang w.r.t. the 5' end.) For each candidate processing site, a very high number of features (686) are computed.[15] These features register local, (including very low-level) detailed informations on the regions up (24nt) and down (50nt) the candidate site. Trained on miRNAs from miRbase 8.0, and tested via 10-fold cross validation, Microprocessor SVM successfully identified 50% of the Drosha processing sites. Moreover, in 90% of the cases, the positions predicted by Microprocessor SVM are within 2nt of the true site.

The second SVM in [18], dubbed miRNA SVM, is a miRNA classifier whose features are:

– the features of the best predicted Drosha cutting site among those computed by Microprocessor SVM for that (extended) pre-miRNA, and
– seven other features that gather statistics on all Drosha candidate sites considered by Microprocessor SVM for that pre-miRNA.

Tests done with this miRNA SVM made the authors conclude that

[11] I.e., the longest region without any asymmetrical loops.
[12] I.e., the longest region in which the difference between the 3' and 5' component of assymetrical loops is not larger than Δl, a parameter.
[13] l_m is the second parameter used for tuning the miR-*abela* classifier.
[14] A "robust" pre-miRNA candidate has the same stem loop secondary structure independent of the size of the larger region wich contains it. See the Methods section in [30] for details on how these robust candidates are computationally defined and identified.
[15] 686 is the number indicated by Helvik and his colleagues in [18]. In fact, if we one notices that two of those feature sets are categorial, then the real number is 242.

1. precursor length
2. loop size
3. distance from the 5' processing site to the loop start
4. (48x4) nucleotide occurrences at each position in the 24nt regions of the precursor 5' and 3' arms
5. (24) base-pair information of each nucleotide for the 24nt at the precursor base
6. (4) nucleotide frequencies in the two regions in feature 4
7. number of base pairs in feature 5
8. (100x4) nucleotide occurrences at each position in the 50nt 5' and 3' flanking regions
.9 (48) base-pair information of each nucleotide for the 48nt in the flanking region outside the precursor
10. (4) nucleotide frequencies in the two regions in feature 8
11. number of base pairs for the 15nt immediately flanking the precursor
12. number of base pairs in the region in feature 9

---

13. number of potential processing sites
14. score of the best processing site
15. average score for all potential processing sites
16. standard deviation for all potential processing sites
17. difference between features and 15
18. distance between the three top scoring processing sites
19. number of local maximums in the processing site score distribution

Fig. 5. The features of Microprocessor SVM (those above the line) and miRNA SVM.

- its performance is close to those of other miRNA classification systems (triplet-SVM, miR-*abela*, and ProMiR [26]);
- in general, the validation of newly proposed (extended) pre-miRNAs should include a check on whether they exhibit or not Drosha cutting sites. Indeed, their work pointed to several entries that seem to have been mistakenly added to the miRbase repository.

The features of Microprocessor SVM and miRNA SVM are listed in Figure 5. The position specific *base-pair information* is 0, 0.5, or 1 if respectively none, one, or both of the nucleotides on the position x upstream of the 5' processing site and $x-2$ downstream of the 3' processing site are base-paired with a nucleotide in the opposite strand.

RNAmicro [19] is an SVM-based system that was constructed with the *aim* to find those miRNAs that have conserved sequence and secondary structures. Therefore it works on alignments instead of sequences (as the other SVMs here presented do). The features used for classification by RNAmicro are:

- the stem length for the miRNA candidate alignment
- the loop length
- the G+C content

- $\overline{MFE}$, the mean of the minimum folding energy *MFE*
- the mean of the z-scores,
- the mean of the *adjusted MFE*,
- the mean of *MFE* index 1,
- the *structure conservation index*, defined as the ratio of $\overline{MFE}$ and the energy of the consensus secondary structure.
- the *average column-wise entropy* for the 5' and 3' sides of the stem and also for the the loop; it is defined as

$$S(\xi) = -\frac{1}{len(\xi)} \Sigma_{i \in \xi} \; \Sigma_{\alpha} \; p_{i,\alpha} \; ln \; p_{i,\alpha}$$

  where $p_{i,\alpha}$ is the frequency of the nucleotide α (one of A,C,G,U) at the sequence position i
- S_{min}, the minimum of the column-wise entropy computed (as above) for 23nt windows on the stem

The positive examples on which RNAmicro was trained were 295 alignments that have been built starting from the miRNA registry 6.0, using homologous sequences. Negative examples were first generated from the positive alignments by doing shuffling until the consensus structure yielded a hairpin structure; 483 alignments of tRNAs were further added to the set of negative examples.

RNAz [34] is an SVM-based system that identifies non-codant RNAs using multiple alignments. RNAmicro was tested by applying it as a further filter to the output provided by RNAz for several genome-wide surveys, including *C. elegans*, *C. intestinalis*, and *H. sapiens*.

[28] defines a stem kernel function to measure the structure similarity between two RNA sequences. It generalizes a particular string kernel, namely the all-substrings kernel [31]. For the calculation of the values of this kernel, the authors use dynamic programming. After introducing the recurrence relations for a first, simple version of the stem kernel, different refinements are proposed, using respectively

- a gap weight
- base pairing probability
- a stacking weight (to reward lengthy stems)
- a minimal length for the stem loop
- a substitution score matrix for base pairs.

The authors' current implementation for the computation of the stem kernel has an $\mathcal{O}(n^2)$ complexity.[16] The stem kernel is used via SVM in a series of experiments with five classes of non-coding RNAs. For each class, 100 positive instances are considered, and other 100 negative instances are generated through sequence shuffling, preserving the dinucleotide frequency [35]. In these experiments, the stem kernel proves to be generally better than the all-substrings kernel. Our

[16] The paper mentions that improvements are on the way.

opinion is that experiments on (significantly) larger data sets are needed in order to assess the quality of this approach.

miPred [27] is an SVM based on 29 global and "intrinsic" hairpin folding features (no phylogenetic information was used):

- dinucleotide frequencies (16 features)
- G+C ratio
- folding features (6 features):
  dP – adjusted base pairing propensity,[17]
  dG – adjusted MFE,
  MFE_1 – MFE index 1,
  MFE_2 – MFE index 2,[18]
  dD – adjusted base pair distance,[19]
  dQ – adjusted Shannon entropy
- dF – a topological descriptor: the degree of compactness,[20]
- the normalised variants of dP, dD, dQ, dF, obtained from dinucleotide shuffling (5 features).

Trained on 200 human pre-miRNAs from miRbase 8.2 and 400 pseudo-hairpins from RefSeq genes,[21] the system was tested on

- the remaining 123 human miRNAs from miRbase 8.2
- 12387 ncRNAs from Rfam 7.0 [16] spanning 40 non-human species
- 31 mRNAs from GeneBank DNA databese (Benso et al., 2005).

and it iyelded 93.50% accuracy at five-fold cross-validation and 0.9833 area under the ROC curve. Tested on the remaining 123 human pre-miRNAs and 246 pseudo hairpins, the system achieved 84.55% sensitivity, 97.97% specificity and 93.50% accuracy. Further more, tested on 1,918 pre-miRNAs across 40 non-human species and 3,836 pseudo hairpins, it yields 87.65% (92.08%), 97.75% (97.42%), and 94.38% (95.64%) for the mean (overall) sensitivity, specificity, and accuracy. The miPred system was also evaluated on four complete viral genomes (*E.Barr virus*, *K.sarcoma-associated herpesvirus*, *M.γ-herpesvirus 68 strain WUMS* and *H.cytomegalovirus strain AD169*). On these viral genomes and seven other full genomes, the reported miPred's sensitivity is 100%(!) while the specificity is higher than 93.75%. Empirically it is shown that six features ensure most of miPred's discriminative power: MFE_1, zG, dP, zP, zQ, dG.

The miPred's authors compare the performances of miPred with those of triplet-SVM and RNAmicro. They conclude that their system is the best, and claim that the features they use express the right characteristics to discriminate miRNAs from other ncRNAs and coding RNAs.

[17] See Schultes at al., 1999.
[18] See Zhang et al, 2006.
[19] See Freyhult et al., 2005.
[20] See Fera et al., 2004, Gran et al., 2004.
[21] See Pruitt and Maglott, 2001.

4 Future work

We will sketch here several directions on which the research work on identifying microRNAs could be further continued.

• A *direct comparison* of the as many as possible of the SVMs for mirRNA identification would be desirable, using either the most comprehensive training and test data sets (at the time of writing this chapter we suggest those documented for miPred [27]) or up-to-date data sets newly derived from miRbase according to established methodologies.
• We plan to test different strategies for automatic learning of *kernel functions* to be used in connection with SVMs here presented. We will start with testing the InfoBoosted GP algorithm ([Gîrdea and Ciortuz, 2007]) on Triplet-SVM (and its extensions), miR-*abela* and miPred.
• It is worth to explore different *feature selection algorithms* that would eventually work well in connection with SVMs (see [Chen and Lin, 2004]). In particular, one could test the effect of *DFL algorithm* [40] on the feature sets of the SVMs here presented (other than miREncoding). Especially for Microprocessor SVM [18], it would be interesting to know which are the *essential attributes* among the many that it employs in order to identify good candidates for Drosha cutting sites.
• The RF meta-learning algorithm [6] provided better results than MiPred [20]. Therefore it would be interesting to see its effect on the feature sets specific to other SVMs here presented. More generally, it would be good to find out other *(meta-)learning algorithms* that have been reported in the literature as capable of better results than SVMs, and to try them on the miRNA identification data sets. The interesting reader could refer for instance to MDO, the Margin Distribution Optimisation algorithm ([Sebe et al, 2006], chapters 3 and 6) which is known to perform better than both Boosting and SVM on certain UCI data sets.
• The claim of Helvik and his colleagues [18] is that *identifying the Drosha cutting sites* (more precisely the output of Microprocessor SVM) significantly improves the quality of a classification system for miRNA identification. This claim was verified for the miRNA SVM. It would be good to check it (or the effect of using it) in connection with other SVMs here presented.
• It would be very interesting (especially in the case of miPred SVM) to see whether *features on randomised/shuffled sequences* could be replaced by other features without loss of classification performance.
• Finally we would like to make the *connection* with the problem of identifying *miRNA target sites* and/or other interesting classification problems for non-coding RNAs.

Acknowledgements

This work has been started in the summer of 2007, while the author was visiting the SEQUOIA team at INRIA–Futur and Université de Lille 1, France, and then the Bioinformatics team at Biozentrum, University of Basel, Switzerland.

Special thanks go to my hosts, dr. Mihaela Zavolan, dr. Hélène Touzet and dr. Grégory Kucherov.

Appendix 1: Random Forests

Random Forest is a meta-learning algorithm that was introduced by Leo Breiman [6]. This technique was derived from *bagging*, which is (like *boosting*) a well-known strategy for aggregating classification algorithms. It is for this reason that boosting, bagging and random forests are called *ensemble learning* methods. Here we will give first a short presentation of boosting and bagging, assuming for the sake of simplicity that the classifiers they aggregate are *decision trees*.

The main idea behind *boosting* ([Shapire et al., 1998]) is the following: decision trees are constructed successively, and each time a new tree is built, the data points that have been incorrectly predicted by earlier trees are given some extra wight. The learner is thus forced to successively concentrate on more and more difficult cases. In the end, the classification of a given instance is decided by a liniar (weighted) combination of the votes given by all decision trees.

In *bagging* [5],[22] new trees do not depend on earlier trees; each tree is independently constructed using a boostrap sample (i.e. sampling with replacing) of the training data set. Classification of a test instance is done via a simple majority voting among decision trees.

The Random Forest (RF) algorithm extends bagging with and additional layer of randomness, namely the *random feature selection*: while in standard decision trees each node is split using the best split among all variables, in RF each node is split using the best among a subset of features randomly chosen at that node.

Thus, RF uses only *two parameters*:

- the number of variables in the random subset at each node (m_{try})
- the number of trees in the forest (n_{tree}).

Although RF is a somehow counter-intuitive strategy, it has been proved *robust against overfitting*, and it gave good results when compared to other machine learning techniques including SVMs, neural networks, discriminat analysis, etc.

Appendix 2: Discrete Function Learning

The Descrete Function Learning (DFL) algorithm introduced by Zheng and Kwoh [40] is a feature selection algorithm based on information theory. We will first recall the *basic notions* of entropy and mutual information.

Let X and Y be two discrete random variables. The *entropy* of Y is defined as

$$H(Y) = -\sum_{y} P(Y=y) \log P(Y=y)$$

[22] The name *bagging* comes from *b*ootstrap *agg*regat*ing*.

and it will be rewritten for convenience as

$$-\sum_y p(y)\log p(y) = E(\log \frac{1}{p(y)})$$

$H(Y)$ describes the *diversity* of (the values taken by) Y:[23] the grater the diversity of Y, the larger the value of $H(Y)$.[24] $H(Y)$ is always non-negative. $H(Y) = 0$ if $P(Y) = 1$, i.e. we are certain about the value of Y.

The *joint entropy* of X given Y is

$$H(X,Y) = -\sum_x \sum_y p(x,y)\log\ p(x,y)$$

$H(X,Y)$ is the amount of information needed on average to specify the values of the two variables. Extending the above definition to the case when X is a vector of two of more variables is straightforward.

The *conditional entropy* of Y given X is

$$H(Y \mid X) = -\sum_x \sum_y p(x)\, H(Y \mid X{=}x) = -\sum_x \sum_y p(x,y)\log p(y \mid x)$$

while the *mutual information* between X and Y is

$$I(X;Y) = H(Y) - H(Y \mid X) = H(X) - H(X \mid Y)$$

$I(X;Y)$ characterises the *relation* between X and Y: the stronger the relation, the larger the value of $I(X;Y)$. $I(X;Y) = 0$ only when X and Y are independent, i.e. $P(X,Y) = P(X)P(Y)$.

It can be shown that the following *chain rule* is true:

$$H(X_1, X_2, \ldots, X_n) = H(X_1) + H(X_2 \mid X_1) + \ldots + H(X_n \mid X_1, X_2, \ldots, X_{n-1})$$

and therefore

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = \sum_x \sum_y p(x,y)\log \frac{p(x,y)}{p(x)p(y)}.$$

The *conditional mutual information* $I(X;Y \mid Z)$, i.e. the mutual information between X and Y given Z, is defined by

$$I(X;Y \mid Z) = H(X \mid Z) - H(X \mid Y,Z) = \sum_{x,y,z} p(x,y,z)\log \frac{p(x,y \mid z)}{p(x \mid z)\ p(y \mid z)}.$$

Another *chain rule*, this time for the mutual information, can be proven:

[23] Manning and Schütze in *Foundations of Statistical Natural Language Processing* (MIT Press, 1999): $H(Y)$ "measures the amount of information in the variable" Y, "the average length of the message needed to transmit the outcome of that variable".

[24] In the definition of $H(Y)$ and also in the sequel it is assumed that the base of the logarithm is 2, and $0\, log\, 0 = 0$.

$$I(X_1, X_2, \ldots, X_n; Y) = I(X_1; Y) + I(X_2; Y \,|\, X_1) + \ldots + I(X_n; Y \,|\, X_1, \ldots, X_{n-1}).$$

The following *theorem* holds, stating that in a certain case there is a "functional" linking between the variables X and Y involved in the expression mutual information:

$$\text{If } I(X;Y) = H(Y) \text{ then } Y \text{ is a function of } X.$$

The interested reader could see [10] for the proof of this theorem. It is immediate that $I(X;Y) = H(Y)$ if and only if $H(Y \mid X) = 0$. The meaning of $H(Y \mid X) = 0$ is that there is *no more diversity* in the values of Y when X is already known.

Zheng and Kwoh proved in [39] that the above theorem entails the following *generalisation*:

> If $X_1, X_2, \ldots, X_n$ and Y are random variables, and
> $I(X_1, X_2, \ldots, X_n; Y) = H(Y)$,
> then Y is a function of $X_1, X_2, \ldots, X_n$.

This result constitutes the basis of the following DFL algorithm.

Let us consider a set of training instances characterised by $X_1, X_2, \ldots, X_n$ as input (categorical) attributes, and Y the class (i.e. output) attribute. We *aim* to find the input attributes that contribute most to the class distinction.

DFL Algorithm:

$V = \{X_1, X_2, \ldots, X_n\},\ U_0 = \emptyset,\ s = 1$
do
$\quad A_s = argmax_{X_i \in V \setminus U_{s-1}} I(U_{s-1}, X_i; Y)$
$\quad U_s = U_{s-1} \cup \{A_s\}$
until $I(U_s; Y) = H(Y)$

The DFL algorithm selects at each execution of the do ... until loop an attribute A_s which, among those that have not been previously selected, brings in the most mutual information w.r.t. the already selected input attributes U_{s-1} and the output Y.

It can be shown that $I(X_{i_1}, \ldots, X_{i_{k-1}}; Y) \geq I(X_{i_1}, \ldots, X_{i_{k-1}}, X_{i_k}; Y)$, therefore $I(U_s; Y)$ increases step by step during the execution of the DFL algorithm.

If Y is a function of $X_1, \ldots\ X_n$, then the algorithm is guaranteed to find $U_s \subseteq \{X_1, \ldots\ X_n\}$ such that $I(U_s; Y) = H(Y)$. Otherwise — or in the very probable event that the training data is noisy —, the following *improvements* can be applied to the DFL algorithm:

The 'until' condition can be replaced with either

$$H(Y) - I(U_s; Y) < \epsilon$$

or

$$s > K$$

where $\epsilon > 0$ and K, a positive integer, can be used as parameters.

The DFL algorithm can be further extended so to that besides identifying U_s as the *essential attributes* that discriminate between the values of Y, it can also learn Y as a (classification) function depending on those selected attributes. Towards this aim, the 1-NN (one nearest neighbour) algorithm [24] is employed, using for instance the Hamming distance. Given a test vector $X_{i_1} = v_1, \ldots, X_{i_k} = v_k$, the (extended) DFL algorithm first finds its nearest neighbour among the training data, and then outputs its label (Y's value) as the class assigned to the given test vector.

The DFL web site [2] mentions a couple of interesting implementation details that make the (incremental) computation of $I(U_s; Y)$ and the learned function very efficient.

References

1. ftp://ftp.sanger.ac.uk/pub/mirbase and http://microrna.sanger.ac.uk/sequences/.
2. http://www.ntu.edu.sg/home5/pg04325488/help/IntroDFL.htm.
3. Peter Bengert and Thomas Dandekar. Current efforts in the analysis of RNAi and RNAi target genes. *Briefings in Bioinformatics*, 6(1):72–85, 2005.
4. Eric Bonnet, Jan Wuyts, Pierre Rouzé, and Yves Van De Peer. Evidence that microRNA precursors, unlike other non-coding RNAs, have lower folding free energies than random sequences. *Bioinformatics*, 20(17):2911–2917, 2004.
5. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
6. Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.
7. J. Brown and Sanseau P. A computational view of microRNAs and their targets. *Drug Discovery Today*, 10:595–601, 2005.
8. Keya Chaudhuri and Raghunath Chatterjee. MicroRNA detection and target prediction: Integration of computational and experimental approaches. *DNA Cell Biology*, 26(5):321–37, 2007.
9. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
10. T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.
11. Nello Cristianini and John Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.
12. Andrew Fire, Siqun Xu, Mary Montgomery, Steven Kostas, Samuel Driver, and Craig Mello. Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans. *Nature*, 391(6669):806–811, February 1998.
13. R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, Inc., second edition, 1987.
14. Hin Hark Gan, Samuela Pasquali, and Tamar Schlick. Exploring the repertoire of RNA secondary structure motifs using grapgh theory; implications for RNA design. *Nucleic Acids Research*, 31(11):2926–2943, 2003.
15. Sam Griffiths-Jones. The microRNA registry. *Nucleic Acids Research*, 32(Database-Issue):109–111, 2004.
16. Sam Griffiths-Jones, Simon Moxon, Mhairi Marshall, Ajay Khanna, Sean Eddy, and Alex Bateman. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Research*, 33(Database issue):D121–D124, January 2005.

17. Gregory Hannon. RNA interference. *Nature*, 418(6894):244–251, 2002.
18. Snorre Helvik, Ola Jr. Snøve, and Pål Sætrom. Reliable prediction of Drosha processing sites improves microrna gene prediction. *Bioinformatics*, 23(2):142–149, 2007.
19. J. Hertel and P. F. Stadler. Hairpins in a haystack: recognizing microRNA precursors in comparative genomics data. *Bioinformatics*, 22(14):e197–e202, July 2006.
20. Peng Jiang, Haonan Wu, Wenkai Wang, Wei Ma, Xiao Sun, and Zuhong Lu. MiPred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features. *Nucleic Acids Research*, 2007.
21. Thorsten Joachims. http://www-ai.informatik.uni-dortmund.de/FORSCHUNG/VERHAREN/SVM_svm_light.eng.html.
22. R. Lee, R. Feinbaum, and Ambros V. The *c. elengans* heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell*, 75:843–854, 1993.
23. Chih-Jen Lin. www.csie.ntu.edu.tw/∼cjlin/libsvm.
24. Tom Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, March 1997.
25. Bojan Mohar. The laplacian spectrum of graphs. In Y. Alavi, G. Chartrand, O. Ollermann, and A. Schwenk, editors, *Graph Theory, Combinatorics and Applications*, pages 871–898. John Willey and Sons, New York, 1991.
26. J.-W. Nam, K.-R. Shin, J. Han, Y. Lee, V. N. Kim, and B.-T. Zhang. Human microRNA prediction through a probabilistic co-learning model of sequence and structure. *Nucleic Acids Research*, 33(11):3570–3581, 2005.
27. Kwang Loong Stanley Ng and Santosh Mishra. De novo SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures. *Bioinformatics*, 23(11):1321–1330, 2007.
28. Yasubumi Sakakibara, Kiyoshi Asai, and Kengo Sato. Stem kernels for RNA sequence analyses. In Sepp Hochreiter and Roland Wagner, editors, *Proceedings of BIRD 2007, The First International Conference on Bioinformatics Research and Development*, volume 4414 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 2007.
29. P. Sethupathy, M. Megraw, and A. Hatzigeorgiou. A guide through present computational approaches for the identification of mammalian microrna targets. *Nature Methods*, 3(11):881–886, 2006.
30. Alain Sewer, Nicodème Paul, Pablo Landgraf, Alexei Aravin, Sébastien Pfeffer, Michael J. Brownstein, Thomas Tuschl, Erik van Nimwegen, and Mihaela Zavolan. Identification of clustered microRNAs using an *ab initio* prediction method. *BMC Bioinformatics*, 6:267, 2005.
31. John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.
32. Karol Szafranski, Molly Megraw, Martin Reczko, and Artemis Hatzigeorgiou. Support vector machines for predicting microRNA hairpins. In Hamid Arabnia and Homayoun Valafar, editors, *Proceedings of the 2006 International Conference on Bioinformatics & Computational Biology, BIOCOMP'06*, pages 270–276. CSREA Press, 2006.
33. L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
34. Stefan Washietl, Ivo Hofacker, and Peter Stadler. Fast and reliable prediction of noncoding RNAs. *Proceedings of the Natural Academy of Science USA*, 102(7):2454–2459, 2005.

35. Christopher Workman and Anders Krogh. No evidence that mRNAs have lower folding free energies than random sequences with the same dinucleotide distribution. *Nucleic Acids Research*, 27(24):4816–4822, 1999.
36. Chenghai Xue, Fei Li, Tao He, Guo-Ping Liu, Yanda Li, and Zhang Xuegong. Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. *BMC Bioinformatics*, 6:310, 2005.
37. Sungroh Yoon and Giovanni De Micheli. Computational identification of microRNAs and their targets. *Birth Defects Research (Part C)*, 78(2):118–128, 2006.
38. Yun Zheng, Wynne Hsu, Mong-Li Lee, and Limsoon Wong. Exploring essential attributes for detecting microRNA precursors from background sequences. In Mehmet M. Dalkilic, Sun Kim, and Jiong Yang, editors, *2006 VDMB Workshop on Data Mining in Bioinformatics*, volume 4316 of *Lecture Notes in Computer Science*, pages 131–145. Springer, 2006.
39. Yun Zheng and Chee Keong Kwoh. Dynamic algorithm for inferring qualitative models of gene regulatory networks. In *Proceedings of The 3rd IEEE Computational Systems Bioinformatics Conference (CSB'04)*, pages 353–362, Washington, DC, USA, 2004. IEEE Computer Society.
40. Yun Zheng and Chee Keong Kwoh. Identifying simple discriminatory gene vectors with an information theory approach. In *Proceedings of The 2005 IEEE Computational Systems Bioinformatics Conference (CSB'05)*, pages 13–24, Washington, DC, USA, 2005. IEEE Computer Society.