**Politecnico di Milano**

**Dipartimento di Elettronica e Informazione**


# Support Vector Machines in Bioinformatics: a Survey

**TECHNICAL REPORT [TP-2012/01]**

**Davide Chicco**
**davide.chicco@elet.polimi.it**

Published online:
12th October, 2012

**Abstract**

Many scientists and researchers have been considering Support Vector Machines (SVMs) as one of the most powerful and robust algorithm in machine learning. For this reason, they have been used in many fields, such as pattern recognition, image processing, robotics, and many others. Since their appearance in 1995, from an idea of Vladimir Vapnik, bioinformatics community started to use this new technique to solve the most common classification and clustering problems in the biomolecular domain.
In this document, we first give a general description of Support Vector Machine technique, a technique based on the statistical learning theory (Section 1). Then we provide a survey of the many applications of the algorithm in the bioinformatics domain (Section 2). Finally, we report a short list of SVM implementation codes available on the internet (Section 3).

**About this survey** This document is freely available and can be download from `http://www.DavideChicco.it` author's website.
Alessandro Lazaric (INRIA, Lille, France, EU) kindly supervised and corrected this document before publication.

# Contents

**1 Support Vector Machines: elements of statistical learning theory** **4**
1.1 Binary classification problem . . . . . 4
1.2 Optimal separation hyperplane . . . . . 7
1.3 Non-linear separable problems . . . . . 8
1.4 Non-linear problems . . . . . 11
1.5 The Kernel functions . . . . . 13
1.6 Algorithm steps resume . . . . . 15

**2 SVMs in Bioinformatics** **16**
2.1 Gene and gene function classification . . . . . 17
2.2 Cancer tissue classification . . . . . 20
2.3 Protein sequences remote homologies recognition . . . . . 24
2.4 Translation initiation site recognition in DNA . . . . . 27
2.5 Other applications . . . . . 30

**3 SVM code** **30**

# 1 Support Vector Machines: elements of statistical learning theory

*"There is nothing so practical as good theory" (Kurt Lewin)*

Support Vector Machines (SVM) are a set of supervised learning algorithms, used in many fields for pattern recognition, classification, and clustering of data. Invented by Vladimir Vapnik [1] in 1995, at Bell AT&T laboratories, they can be considered as a special case of the Tichonov Regularization method [2].
Given $N \in \mathbb{N}$ elements in input, and two disjointed classes in output, the basic SVM takes the input elements, elaborates them (*learns by them*, actually), and, for each of them, predicts if it belongs to the first class or the second one. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other.

## 1.1 Binary classification problem

We consider a problem of binary classification. We have an input space $X \subseteq \mathbb{R}^d$, where $d \in \mathbb{N}$, an output space $Y = \{-1, +1\}$, and a training set, $S$, where: $S = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\} \in (X \times Y)$.
Our problem is to find a function (relation) that map each element of $X$ into $Y$. In the classification context, this means to find a decision rule that, given an input element $x \in X$, tell if it belongs to $-1$ class or to $+1$ class.

We can build a linear binary classificator from a function $f : X \subseteq \mathbb{R}^N \to \mathbb{R}$ in this way: if $f(x) \geq 0$, $x$ belongs to $-1$ class; else, $x$ belongs to $+1$ class.
So, in the linear case:

$$f(x) = (w \cdot x) + b \tag{1}$$

where $w \in \mathbb{R}^d$. Since this function *discriminates* if input $x$ elements belongs to -1 or +1 class, it is called *discriminant function.*
In the geometric representation of this problem, $f$ represents the hyperplane

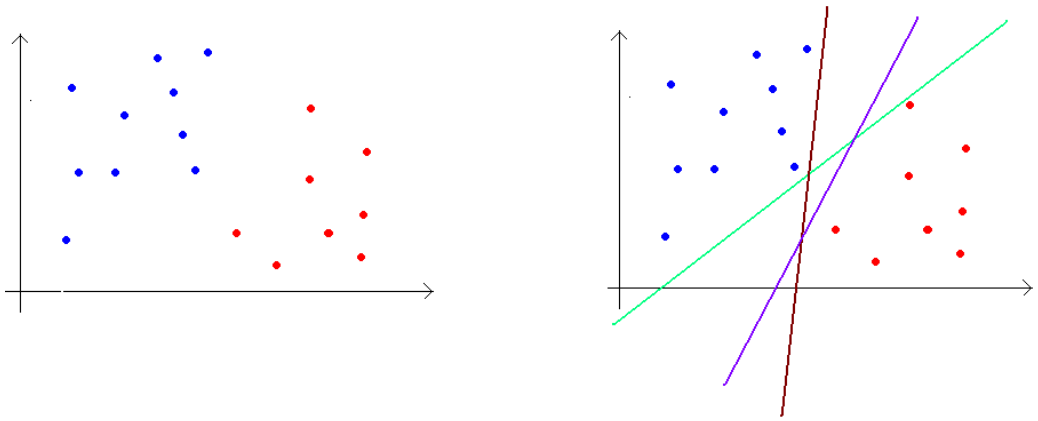

Figure 1: In the left image, the input points to classify. On the right image, point classes are divided by many possible hyperplane. Which is the best separator?

of all the possible planes that are able to correctly classify the input data. The modeling of this hyperplane is managed by $w$ and $b$ parameters. These important $w$ and $b$ parameters are *learned* by the algorithm from the traning set data.

This problem could be represented as the choice of the best separator hyperplane, in a Cartesian system, of two classes of points $(-1, +1)$, as shown in Fig. 1.

SVM algorithm defines the optimal separation hyperplane as the one that maximize the area between the nearest points of the two classes. These nearest points are so called Support Vectors and are very important because they are the only ones that influence the solution of the problem, as shown in Fig. 2. In addition, if the binary classification problem training set can be replaced with only support vector points, the problem would be mathematically equivalent, and computationally less expensive.

In constrained optimization, it is often possible to convert the primal problem (i.e. the original form of the optimization problem) to a *Lagrangian dual problem*. The Lagrangian dual problem is obtained by forming the Lagrangian, using nonnegative Lagrange multipliers to add the constraints to

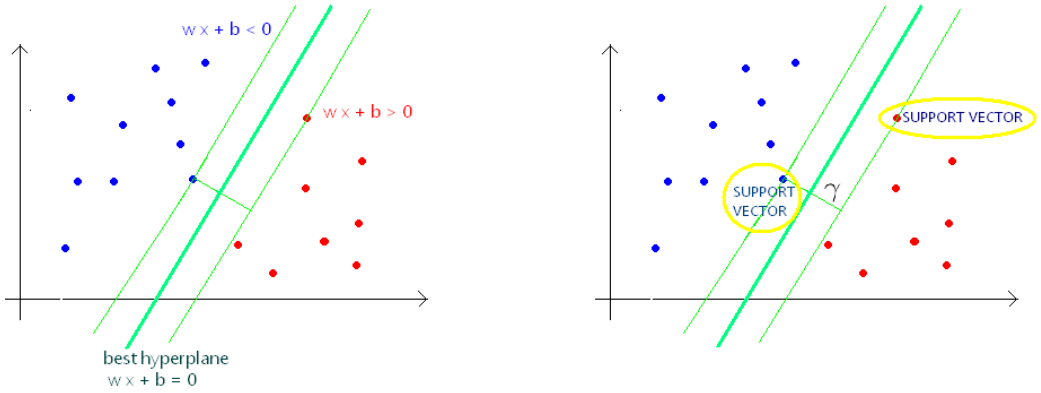


Figure 2: The optimal separation hyperplane maximizes the distance between the support vector points

the objective function, and then solving for some primal variable values that minimize the Lagrangian [3].
We could build the *dual form* of $f(x)$ linear equation 1, as follows:

$$L(w, b, \alpha) = f(x) = \sum_{i=1}^{N} \alpha_i \; y_i \;\; (w \cdot x) + b \tag{2}$$

where $\alpha_i$ is the parameter we will study to choose the best Lagrangian function, between the possible ones.

We define the *functional margin* of a point $(x_i, y_i)$ with respect with a hyperplane $(w, b)$, as:

$$\gamma_i = y_i \; (w \cdot x_i + b) \tag{3}$$

If we consider the separating hyperplane having form $(w \cdot x) + b = 0$, the signed distance of a point $(x_i, y_i)$ from this hyperplane is:

$$d_i = \frac{(w \cdot x_i) + b}{||w||} \tag{4}$$

Since all the couples $(\lambda w, \lambda b)$, where $\lambda \in \mathbb{R}^+$, generate the same $f$ function, we use the canonical form of the hyperplane. This is obtained by scaling

$(w, b)$ in order to make the distance between the hyperplane and its nearest points equal to $\frac{1}{||w||}$. Because of this transformation, the nearest points are those that solve the equation:

$$y_i(w \cdot x_i + b) = 1 \tag{5}$$

Coming back to our training set $S$, we can say that $S$ is *linearly separable* if a parameter couple $(w, b)$ exists, where $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$, and where:

$$\forall_i \; y_i(w \cdot x_i + b) \geq 1 \tag{6}$$

## 1.2 Optimal separation hyperplane

The optimal separation hyperplane represents the best choice to divide the input element of $X$ into the output classes $Y$.
Since the the optimal separation hyperplane maximize the $\frac{2}{||w||}$ margin, its parameters $(w, b)$ are the minimum possible ones that minimizes $\frac{1}{2}\ (w \cdot w)$, between all the possible $w$ and $b$ that respect the condition in (6):

$$\min_{w,b} : \frac{1}{2}(w \cdot w) \tag{7}$$

$$subject\ to\ \ \forall_i \; y_i(w \cdot x_i + b) \geq 1 \tag{8}$$

To do this, we consider again the dual form of $f$, as described in (2). We can rewrite it as follows:

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{N} \alpha_i \; [y_i \;\; (w \cdot x + b) - 1] \tag{9}$$

If we compute the derivatives respect with $b$ and $w$ variables, they are:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} y_i \; \alpha_i = 0 \tag{10}$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} y_i \, \alpha_i \, x_i = 0 \tag{11}$$

If we replace formulas in (10) and in (11) into Lagrangian in (9), we get a dual form equation dependent only on moltiplicators $\alpha_i$.
From formula in (10) and (9), we get

$$w = \sum_{i=1}^{N} y_i \, \alpha_i \, x_i \tag{12}$$

And this is very important, because if we find a proper $w$ such that

$$y_i(w \cdot x_i + b) = 1 \quad \& \quad \alpha_i \neq 0 \tag{13}$$

then $x_i$ **a support vector**.

The value of $\alpha_i$ can be found by solving the following (dual) quadratic optimization problem: find an $\alpha$ such that

$$max(w_\alpha) : \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \, y_i y_j \, x_i x_i \tag{14}$$

where $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$

And, finally, by using Wolfe Dual Form and Kuhn Tucker conditions, the hyperplane (1) can be written as:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \, (x \cdot x_i) + b \tag{15}$$

## 1.3 Non-linear separable problems

In the previous section, we described the problem of binary classification in the linear separable case, and we assumed that there were no error in the training set. Now we will analyze the case of presence of error points inside

the two classes to be classified, as shown in Fig. 3.

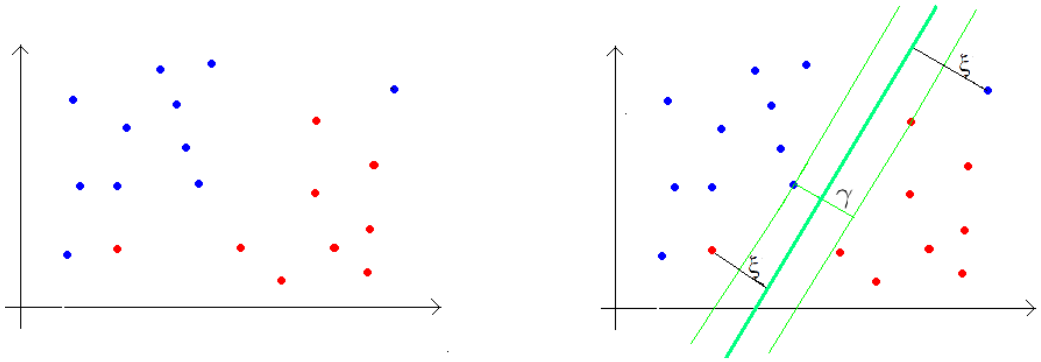


Figure 3: These two classes contain some error points, that differ $\xi$ from the optimal hyperplane.

In the previous cases our aim was to find a separator surface that divided **all** the points in the correct way; on the contrary, now we are looking for a mathematical model that divides the **most part** of the points in the correct way.

To do this, we introduce the $\xi$ (pron.: *xi*) variables, called the *slacks*, that represent the error distance of the *wrong* points from the optimal separating hyperplane: $\xi = (\xi_1, \xi_2, ..., \xi_N)$, where $\forall_i \; \xi_o \geq 0$
New constraints are so introduced in the formula in (6):

$$\forall_i \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i \tag{16}$$

In section 1.2 we defined the best hyperplane as the one that minimize $\frac{1}{2}\,(w \cdot w)$, between all the possible $(w, b)$ couples that respect the condition in (6). Now, with error $\xi$, we have to rewrite the optimal hyperplane equation in (7). We want to find the couple $(w, b)$ that minimize the equation:

$$\begin{aligned} \min_{w,b} : \frac{1}{2}(w \cdot w) + C \sum_{i=2}^{N} \xi_i \\ subject\ to : \quad \forall_i \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i \\ \forall \xi_i \geq 0 \\ C \in \mathbb{R} \end{aligned} \tag{17}$$

Parameter $C$ allow us to control the number of badly classified points.
We also redefine the Lagrangian in (9) this way:

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) + C \sum_{i=2}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \, [y_i \ \ (w \cdot x + b) - 1 + \xi_i] - \sum_{i=1}^{N} \beta_i \xi_i \tag{18}$$

where $\beta_i$, similarly to $\alpha_i$, are real values introduced by Lagrangian transformation.
Again, like we did for Lagrangian in linearly separable case, we compute the derivatives respect with $b$, $w$ and $\xi_i$ variables. The derivatives of $b$ and $w$ are same of the previous case (10)(11), while for $\xi_i$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i + \beta i = 0 \tag{19}$$

If we replaced derivatives in (10), (11) and (19) into Lagrangian in (18), we would get a dual form equation dependent only on $\alpha_i$ and $\beta_i$ moltiplicators. Then, if we consider that all the $\beta_i$ can be written in terms of $\alpha_i$ terms.
By using again Wolf Dual form, we can reach the same formula in (12) and (14), with the difference that now the condition on $\alpha_i$ changes to $0 \leq \alpha_i \leq C$.
While, for Kuhn Tucker conditions, $b$ and $\xi_i$ can be found as:

$$\alpha_i \, [y_i(w \cdot x_i) + b - 1] = 0 \tag{20}$$

$$(C - \alpha_i) \, \xi_i = 0 \tag{21}$$

All the cases in which $\alpha_i = C$ are considered errors.
Despite these little variations respect with the linear case, also in this case $f$ could be written as the formula in (15):

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \, (x \cdot x_i) + b \tag{22}$$

## 1.4 Non-linear problems

The linear model introduced in the previous section has some limits: linear classifiers often are not suitable for more complex systems and problems. In fact, the points represented in figures (1) and (2) can be completely separated by a single line. But, unfortunately, not often the problem is so easy: the points (training set data) sometimes could also not be separate by a line (Fig. 4 ).

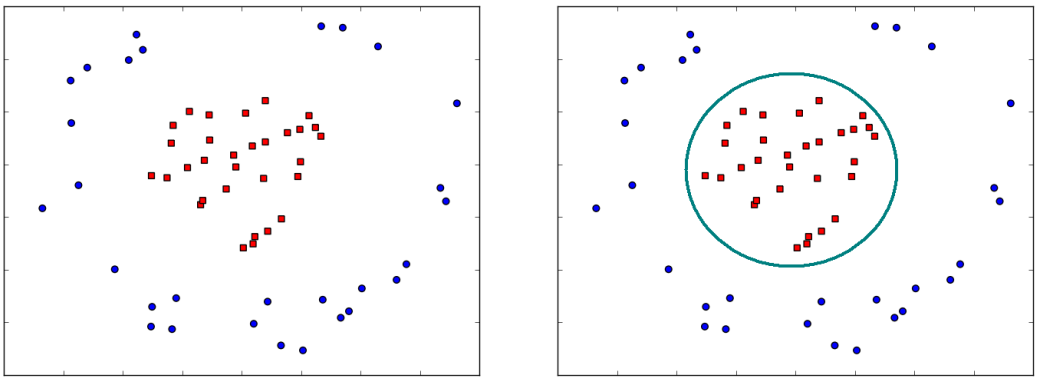

Figure 4: Example of two classes of points not linearly separable.

In this case, basic idea of Support Vector Machines is **to map** input data into a **convenient space** where they can be separated with an opportune hyperplane. For this transformation, they use a special mapping function $\Phi : x \rightarrow \phi(x)$, that takes data from input space and project them to a feature space with higher dimensionality, as shown in Fig. (5).

The discriminant function (15) becomes:

$$f(x) = \sum_{i=1}^{N} w_i \phi_i(x) + b \tag{23}$$

where $\phi : X \rightarrow F$ is a non-linear mapping, from input space to features space.

We can now redefine the classification problem in two steps: first, we have to choose an appropriate non-linear mapping function, and to map data into the new feature space; second, we have to use this new linear system to train our data.

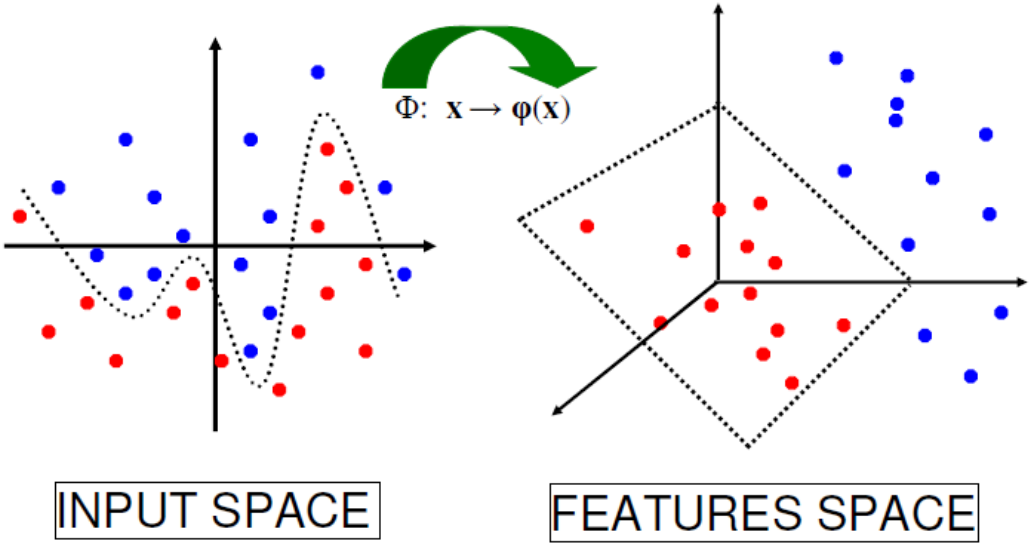


Figure 5: Transformation function $\Phi$ (image on courtesy of Daniele Loiacono).

The mapping function $\phi$ associates each input element (point) to an alternative representation of it, from $X$ input space having dimensionality $d$, to a feature space $F$ having dimensionality $D$:

$$x = (x_1, x_2, ..., x_d) \ \rightarrow \ \phi = (\phi_1(x), \phi_2(x), ..., \phi_D(x)) \tag{24}$$

Projecting all the input points with (24) equation is equivalent to map the $X$ input space to $F$ feature space: $F = \{\phi(x) : x \in X\}$. We recall that $X$ input space and $F$ feature space have different dimensions ($d$ and $D$): usually $D >> d$, so the mapping causes a dimensionality increase.

Coming back to the optimal hyperplane problem, considering the mapping, we can rewrite the hyperplane function in (15) as follows:

$$f(x) = \sum_{i=1}^{N} \alpha_i \ y_i \ (\phi(x) \cdot \phi(x_i)) + b \tag{25}$$

At this point, it would be great to have an instrument that, given the original input space points, calculate the $(\phi(x) \cdot \phi(x_i))$ product directly in the feature space, instead of computing it in the input space and then mapping the point in the feature space.
To do this, we introduce a keypoint of Support Vector Machines: the **kernel** function.

## 1.5 The Kernel functions

A Kernel function K is defined this way: $\forall x, z \in X \ : \ K(x, z) = \phi(x) \cdot \phi(z)$, where $\phi$ is any mapping function from $X$ to $F$.
We can, once again, rewrite the optimal hyperplane solution equation from (25) in the following way:

$$f(x) = \sum_{i=1}^{N} \alpha_i \ y_i \ \ K(x, x_i) + b \tag{26}$$

With kernel, the equation to minimize (17) can be written again this way:

$$\begin{aligned} \min_{w,b,\xi} : \frac{1}{2}(w \cdot w) + C\sum_{i=2}^{N}\xi_i \\ subject\ to: \quad \forall_i \quad y_i(w \cdot \phi(x_i) + b) \geq 1 - \xi_i \\ \forall \xi_i \geq 0 \\ C \in \mathbb{R} \end{aligned} \tag{27}$$

As you can understand, kernel functions are a very powerful tool to solve classification problem, because they can be used to project (move) many problems from a space where they *cannot* be solved to a proper space where they *can* be solved. For this reason, the choice of the correct kernel function to use is very important and ticklish.
Since the kernel function has somehow to *stand in for* the scalar product, it must respect some properties.

For any $x, y, z \in X$ and $\alpha \in \mathbb{R}$:
1) $x \cdot x = 0$ only if $x = 0$
2) $x \cdot x > 0$ otherwise
3) $x \cdot y = y \cdot x$
4) $(\alpha x \cdot y) = \alpha(x \cdot y)$
5) $(z + x) \cdot y = (z \cdot y) + (x \cdot y)$

In addition, the kernel function has also to respect the condition given by Mercer theorem.[4] Since kernels have to respect these conditions, there are few functions that can be considered *kernel*.
The most common and famous are:
**Linear** $K(x_i, x_j) = (x_i \cdot x_j)$
**Polynomial** $K(x_i, x_j) = [(x_i \cdot x_j) + 1]^d$ where $d \in \mathbb{N}, d \neq 0$
**Gaussian** $K(x_i, x_j) = e^{\frac{-||x_i - x_j||^2}{2\sigma^2}}$ where $\sigma^2$ is the standard deviation

In particular, separation surface generated by Gaussian kernel can be interpreted as a weighted sum of Gaussian distributions, of size $\sigma$, with center on the Support Vector points. The choice of $\sigma$ parameter vastly influences the extent of the obtained surface.

Kernel functions can be also composed. If $K_1$ and $K_2$ are kernels, K is kernel, too, if:
1) $K(x_i, x_j) = K_1(x_i, x_j) + K_2(x_i, x_j)$
2) $K(x_i, x_j) = \alpha\, K_1(x_i, x_j) \quad \text{with } \alpha \in \mathbb{R}$
3) $K(x_i, x_j) = K_1(x_i, x_j) \cdot K_2(x_i, x_j)$

## 1.6 Algorithm steps resume

In this section we will sum up all the algorithm phases. A workflow of SVM algorithm is shown in Fig. 6.

### Training phase

- Input: training set; kernel function $\phi$
- Main procedure: minimize the optimal hyperplane equation in (27) subject to its constraints
- Output: prediction model ($w, b, \xi$ parameters of the optimal hyperplane equation)

### Prediction phase

- Input: validation set, prediction model
- Main procedure: apply the prediction model to the validation set
- Output: accuracy of prediction in percentage

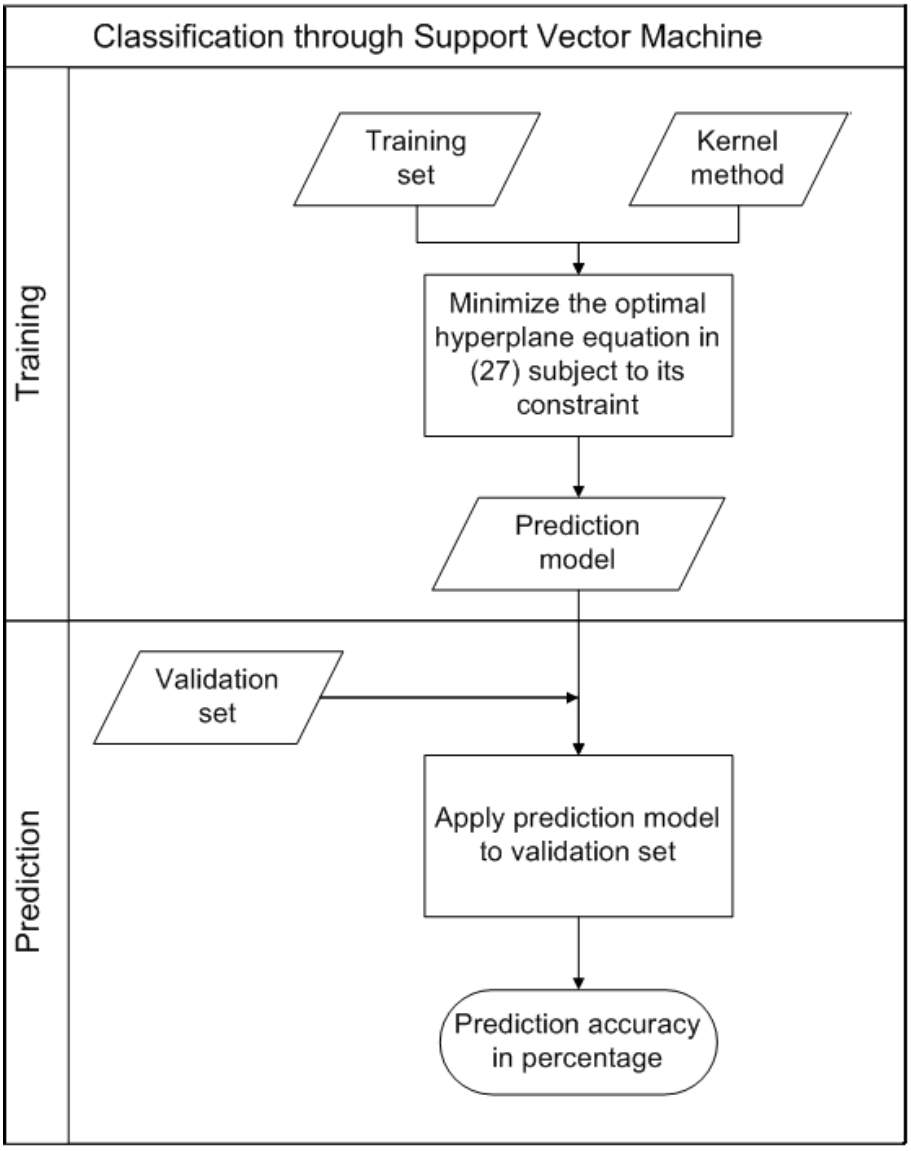


Figure 6: Workflow of the classification algorithm

# 2 SVMs in Bioinformatics

Since SVMs demonstrated to be very powerful and strong algorithms, they have been applied to many domains, including bioinformatics. Bioinformatics classification problems are manifold, and so are the applications of SVMs in bioinformatics literature.

Between all the several papers and research available, we decide to analyze more in details four main application of SVM in bioinformatics (Section (2.1), Section (2.2), Section (2.3), Section (2.4)). we report other application papers in Section 2.5.

## 2.1 Gene and gene function classification

The application of SVMs in gene and gene function classification were described in many papers, such as [6][7][8]. As example, here we describe in details the problem and the application depicted in [6]

**Domain, format and technologies** A *DNA microarray* is a collection of microscopic DNA spots attached to a solid surface. Scientists use DNA microarrays to measure the expression levels of large numbers of genes simultaneously.
Each DNA spot contains picomoles ($10^{-12}$ moles) of a specific DNA sequence, known as *probes*. These can be a short section of a gene or other DNA element that are used to hybridize a Complementary DNA (cDNA) or RNA (cRNA) sample (called *target*) under high-stringency conditions [5].

**The problem** DNA microarray technology provides biologists with the ability to measure the expression levels of thousands of genes in a single experiment. Initial experiments suggest that genes of similar function yield similar expression patterns in microarray hybridization experiments.
As data from such experiments accumulates, it will be essential to have accurate means for extracting biological significance and using the data to assign functions to genes.
Each data point produced by a DNA microarray hybridization experiment represents the ratio of expression levels of a particular gene under two different experimental conditions. The result, from an experiment with $n$ genes on a single chip, is a series of $n$ expression-level ratios. Typically, the numerator of each ratio is the expression level of the gene in the varying condition of interest, whereas the denominator is the expression level of the gene in some

reference condition. The data from a series of $m$ such experiments may be represented as a gene expression matrix, in which each of the $n$ rows consists of an $m$-element expression vector for a single gene.

**Method** Authors defined the expression measurement $X_i$ as the logarithm of the ratio of expression level $E_i$ for gene $X$ in experiment $i$ to the expression level $R_i$ of gene $X$ in the reference state, normalized so that the expression vector $\vec{X} = (X_1, ..., X_n)$ has Euclidean length 1.
The resulting equation is:

$$X_i = \frac{log(E_i/R_i)}{\sqrt{\sum_{j=1}^{n}(E_j/Rj)}} \tag{28}$$

The expression measurement $X_i$ is positive if the gene is induced (turned up) respect with the reference state, and negative if it is repressed (turned down). The inducted state and the repressed state are effects of certain treatments.

Each vector $\vec{X}$ in the gene expression matrix may be thought of as a point in an $m$-dimensional expression space. A simple way to build a binary classifier is to construct a hyperplane separating class members (positive examples) from non-members (negative examples) in this space: that is why Michael P. S. Brown and other authors chose to use SVM classifier.

Support vector machines (SVMs) use a training set to specify in advance which data should cluster together. As applied to gene expression data, an SVM would begin with a set of genes that have a common function: for example, genes coding for ribosomal proteins or genes coding for components of the proteasome. In addition, a separate set of genes that are known not to be members of the functional class is specified. These two sets of genes are combined to form a set of training examples in which the genes are labeled positively if they are in the functional class and are labeled negatively if they are known not to be in the functional class.
A set of training examples can easily be assembled from literature and

database sources. Using this training set, an SVM would learn to discriminate between the members and non-members of a given functional class based on expression data. Having learned the expression features of the class, the SVM could recognize new genes as members or as nonmembers of the class based on their expression data.

**Experiments** Authors analyzed expression data from 2,467 genes from the budding yeast *Saccharomyces cerevisiae* organism measured in 79 different DNA microarray hybridization experiments. From these data, they were able to recognize five functional classes from the Munich Information Center for Protein Sequences Yeast Genome Database (MYGD).
Using the class definitions made by the MYGD, they trained SVMs to recognize six functional classes: tricarboxylic acid (TCA) cycle, respiration, cytoplasmic ribosomes, proteasome, histones, and helix-turn-helix proteins. The MYGD class definitions come from biochemical and genetic studies of gene function whereas the microarray expression data measures mRNA levels of genes.
Performance was tested by using a three-way cross-validated experiment. The gene expression vectors were randomly divided into three groups. Classifiers were trained by using two-thirds of the data and were tested on the remaining third. This procedure was then repeated two more times, each time using a different third of the genes as test genes.

**Results and discussion** The performance of the SVM classifiers was compared with that of four standard machine learning algorithms, as shown in Table 7. M. Brown and W. Grundy considered the following methods: SVMs using the scaled dot product kernel raised to the first, second, and third power (D-p 1 SVM, D-p 2 SVM, D-p 3 SVM), the radial basis function SVM (Radial SVM), Parzen windows, Fisher's Linear Discriminant (FLD), and the two decision tree learners, C4.5 and MOC1.

To judge overall performance, [6] paper defines the cost of using the method M as $C(M) = fp(M) + 2 \cdot fn(M)$, where $fp(M)$ is the number of false posi-

tives for method M, and $fn(M)$ is the number of false negatives for method M. The false negatives are weighted more heavily than the false positives because, for these data, the number of positive examples is small compared with the number of negatives.
The cost for each method is compared with the cost $C(N)$ for using the null learning procedure, which classifies all test examples as negative. Authors define the cost savings of using the learning procedure M as $S(M) = C(N) - C(M)$.
Each method clustered genes in the desidered classes, and, afterwards, every classification was labeled in false positives, false negatives, true positives, and true negatives.
As you can see in Table 7 for tricarboxylic acid (TCA) data, SVMs outperformed other methods, showing higher $S(M)$ than other values.

**Table 1. Comparison of error rates for various classification methods**

| Class | Method | FP | FN | TP | TN | S(M) |
|---|---|---|---|---|---|---|
| TCA | D-p 1 SVM | 18 | 5 | 12 | 2,432 | 6 |
| | D-p 2 SVM | 7 | 9 | 8 | 2,443 | 9 |
| | D-p 3 SVM | 4 | 9 | 8 | 2,446 | 12 |
| | Radial SVM | 5 | 9 | 8 | 2,445 | 11 |
| | Parzen | 4 | 12 | 5 | 2,446 | 6 |
| | FLD | 9 | 10 | 7 | 2,441 | 5 |
| | C4.5 | 7 | 17 | 0 | 2,443 | −7 |
| | MOC1 | 3 | 16 | 1 | 2,446 | −1 |

Figure 7: Results shown in the [6] article

## 2.2 Cancer tissue classification

The application of SVMs in cancer tissue classification were described in many papers, such as [9] [10] [11] [12]. As example, here we describe in details the problem and the application depicted in [10].

**Domain, format and technologies** Also in this case, the technology used is microarray (described in Section (2.1)).

**The problem** Microarray expression experiments allow the recording of expression levels of thousands of genes simultaneously. These experiments primarily consist of either monitoring each gene multiple times under many conditions, or alternately evaluating each gene in a single environment but in different types of tissues, especially cancerous tissues. Those of the first type have allowed for the identification of functionally related genes due to common expression patterns (Section (2.1)), while the latter experiments have shown promise in classifying tissue types (diagnosis) and in the identification of genes whose expressions are good diagnostic indicators.
In order to extract information from gene expression measurements, different methods have been employed to analyze this data, including SVMs.

In recent years, several methods have been developed for performing gene expression experiments. Measurements from these experiments can give expression levels for genes (or Expressed Sequence Tags, ESTs) in tissue or cell samples.
Datasets used by authors for experiments in [10] consist of a relatively small number of tissue samples (less than 100) each with expression measurements for thousands of genes.

**Method** In these papers, a systematic and principled method is introduced, that analyses microarray expression data from thousands of genes, tested in multiple tissue or cell samples. The primary goal is the proper classification of new samples. Terrence Furey and Nigel Duffy do this by training the SVM on samples classified by experts, then testing the SVM on samples it has not seen before. Authors demonstrate how SVMs can not only classify new samples, but can also help in the identification of those which have been wrongly classified by experts. SVMs are not unique among classification methods in this regard, but authors show they are effective. Their method is demonstrated in detail on data from experiments involving 31 ovarian cancer, normal ovarian and other normal tissues. This way, it was possible to identify one tissue sample as mislabeled, and another as an outlier.

**Experiments** Microarray expression experiments are performed using 97,802 DNA clones, each of which may or may not correspond to human genes, for 31 tissue samples. These samples are either cancerous ovarian tissue, normal ovarian tissue, or normal non-ovarian tissue.
For the purpose of these experiments, the two types of normal tissue are considered together as a single class. The expression values for each of the genes are normalized such that the distribution over the samples had a zero mean and unit variance.
Hold-one-out cross-validation experiments are performed. The SVM is trained using data from all but one of the tissue samples. The sample not used in training is then assigned a class by the SVM.
A single SVM experiment consists of a series of hold-one-out experiments, each sample being held out and tested exactly once. Initially, experiments are carried out using all expression scores with diagonal factor settings of 0, 2, 5 and 10.
The genes are then ranked in the manner described previously, and datasets consisting of the top 25, 50, 100, 500 and 1000 features are created. Experiments using similar diagonal factors to those above are performed using these smaller feature sets. The best classification is done using the top 50 features with a diagonal factor of 2 or 5.
Though the smaller datasets achieve slightly better scores compared to using all features, T. Furey and N. Duffy did not believe this improvement to be significant.

**Results and discussion** After ranking the features using all 31 samples, authors attempt to sequence the ten top-ranked genes to determine if they are biologically significant. This analysis seems to suggest that the feature selection method is able to identify clones that are cancer-related, and rank them highly.
Despite additional effort is needed to develop ways of identifying meaningful features in these types of datasets, from a tumor biologists point of view however, the accumulation of tumor-related genes at the top is a very useful

feature.

The [10] paper results indicate that SVMs are able to classify tissue and cell types based on this data. The datasets currently available contain relatively few examples and thus do not allow one method to demonstrate superiority. Microarray expression experiments have great potential for use as part of standard diagnosis tests performed in the medical community. Authors shown along with others that expression data can be used in the identification of the presence of a disease and the determination of its cell lineage.

In Figure (8) you can see a graph representing the SVM classification margins for ovarian tissues. When classifying, the SVM calculates a margin which is the distance of an example from the decision boundary it has learned. In this graph, the margin for each tissue sample calculated is shown.
A positive value indicates a correct classification, and a negative value indicates an incorrect classification. The most negative point corresponds to tissue N039. The second most negative point corresponds to tissue HWBC3.

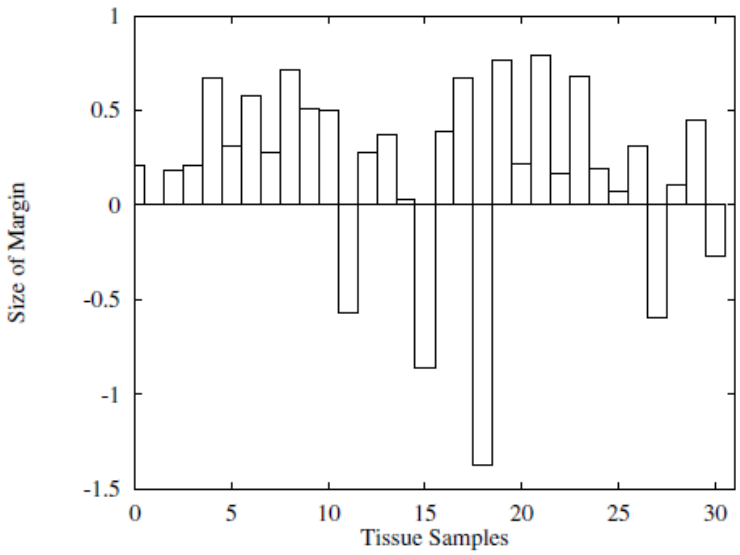


Figure 8: Results shown in the [10] article.

## 2.3 Protein sequences remote homologies recognition

The application of SVMs in recognition of remote homologies between protein sequences were described in many articles like [14] [15]. In this section, we will describe paper in [15].

**Domain, format and technologies** The Structural Classification of Proteins (SCOP) domain is currently recorded into a database, called PDB90, now having version 1.37. This database was used as the source of both training and test sequences of experiments in [15] paper. PDB90 eliminates a large number of essentially redundant sequences from the SCOP database. The use of the domain database allows for accurate determination of a sequences class, eliminating the ambiguity associated with searching whole-chain protein databases.

**The problem** A core problem in statistical biosequence analysis is the annotation of new protein sequences with structural and functional features.
To a degree, this can be achieved by relating the new sequences to proteins for which such structural properties are already known, i.e., by detection of protein homologies.
Many statistical, sequence-based tools have been developed for detecting protein homologies (BLAST, Fasta, templates, profiles, position-specific weight matrices, and Hidden Markov Models (HMMs)). Recent experiments have used the Structural Classification of Proteins classification of protein structures to test many of these methods to see how well they detect remote protein homologies that exist between protein domains that are in the same structural superfamily, but not necessarily in the same family.

**Method** In this approach T. Jaakola and M. Diekhans use generative statistical models built from multiple sequences, in this case HMMs, as a way of extracting features from protein sequences. This maps all protein sequences to points in a Euclidean feature space of fixed dimension. Authors then use

a general discriminative statistical method to classify the points representing protein sequences by domain superfamily.
This is quite distinct from methods that train the parameters of the HMM itself to give a more discriminative model The statistical modeling approach to protein sequence analysis involves constructing a generative probability model, such as an HMM, for a protein family or superfamily.

Sequences known to be members of the protein family are used as (positive) training examples. The parameters of a statistical model representing the family are estimated using these training examples, in conjunction with general a priori information about properties of proteins. The model assigns a probability to any given protein sequence. If it is a good model for the family it is trained on, then sequences from that family, including sequences that were not used as training examples, yield a higher probability score than those outside the family. The probability score can thus be interpreted as a measure of the extent to which a new protein sequence is homologous to the protein family of interest.

**Experiments** T. Jaakola and M. Diekhans designed a set of experiments to determine the ability of SVM-Fisher kernel discriminative models to recognize remote protein homologs. The SVM-Fisher kernel methods were compared to BLAST and the generative HMMs built using the SAM-T98 methodology. The experiments measured the recognition rate for members of superfamilies of the SCOP protein structure classification scheme.
Authors simulate the remote homology detection problem by with holding all members of a SCOP family from the training set and training with the remaining members of the SCOP superfamily. Sequences are then tested from the withheld family to see if they are recognized by the model built from the training sequences.
Since the withheld sequences are known remote homologs, T. Jaakola and M. Diekhans are able to demonstrate the relative effectiveness of the techniques in classifying new sequences as remote members of a superfamily.

In a sense, T. Jaakola and M. Diekhans are asking, Could the method discover a new family of a known superfamily?

**Results and discussion** The [15] paper provide a comparison of the results of the best performing approaches for each of the methods.Since the numeric scores produced by each method are not directly comparable, authors use the *rate of false positives (RFP)* as performance measure.
The RFP for a positive test sequence is defined as the fraction of negative test sequences that score as high or better than the positive sequence. Considering the performances of all four methods on each of the 33 protein families T. Jaakkola and M. Diekhans tested, as measured by the maximum and median RFP, SVMs almost always outperformed Blast and Samt methods, as shown in Figure (9).
In Figure (9) T. Jaakkola and M. Diekhans compare the overall performance for the four methods on the 33 test families. For each family they computed the maximum RFP for that family. Possible values for this RFP are shown on the X-axis. On the Y-axis authors plot the number of SCOP families, out of the 33 families that we tested, for which the given method achieves that RFP performance or better.

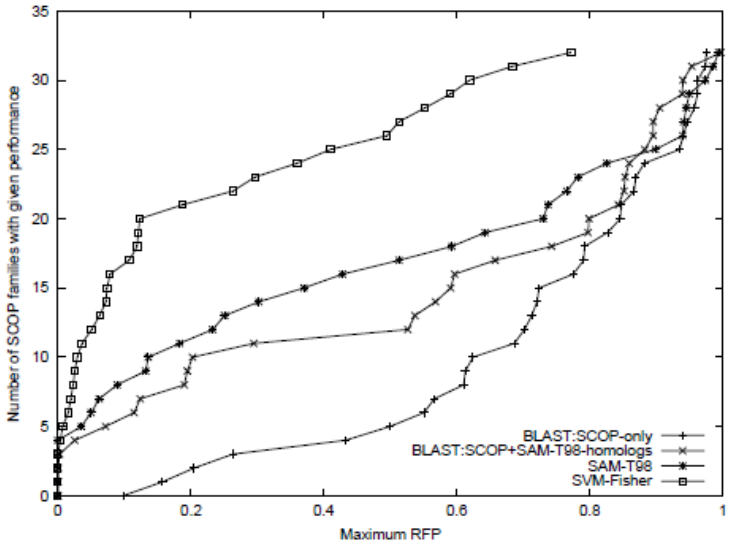


Figure 9: Results shown in the [15] article.

## 2.4 Translation initiation site recognition in DNA

The application of SVMs in translation initiation site recognition in DNA were by Zien et al. in this paper ([18]).

**Domain, format and technologies** The original sequence set of Pedersen and Nielsen has been assembled from high quality nuclear genomic sequences of a selected set of vertebrates taken from GenBank. All introns were removed, in analogy to the splicing of mRNA sequences. Only high quality entries with at least 10 nucleotides upstream and 150 downstream of the start codon were selected. In order to avoid over-optimistic performance estimates resulting from biased data samples, the set was thoroughly reduced for redundancy.
As a consequence, the results below represent lower limits to the performance to be expected on real world data, which is heavily redundant. The data selection protocol left 3,312 sequences). From the work of other investigators, authors expect typical features of TIS to differ for different branches in the evolutionary tree. This implies that the trained classificator will only be valid for mammals, and that retraining on other sequence sets will be necessary for different groups of species.

**The problem** Living systems are determined by the proteins that they produce based on their genomes. But only parts of the genomic text in fact code for proteins, and that parts are called Translation Initiation Sites (TIS). These parts are called coding sequence (CDS).
Therefore, given a piece of DNA or mRNA sequence, it is a central problem in computational biology to determine whether it contains CDS, and, if so, for which protein it codes.

Since living cells are able to distinguish between CDS and other nucleotide

sequence parts without utilizing any homology information, this should also be possible for computer programs, in principle.
In fact, there are algorithms that identify CDS merely relying on properties intrinsic to nucleotide sequences, like GENSCAN and ESTScan. All models discussed so far can be called generative, as they can be used to generate potential TIS sequences with approximately the true probability distribution.

Applying such models, a sequence is considered a TIS if the probability with which the sequence is generated by the model exceeds some threshold. The more closely the true distribution is approximated, the better this approach works. By using so-called discriminative methods, often a superior distinction can be achieved between true TIS and similarly looking pieces of sequence (called pseudo sites).
By using so-called discriminative methods, often a superior distinction can be achieved between true TIS and similarly looking pieces of sequence (called pseudo sites). This paper shows that applying support vector machines (SVMs) for TIS recognition can outperform other established computer machine learning methods.

**Method** Experiments were realized using literature-described Neural Network methods, and SVM methods with some differences. Although the SVM with the polynomial kernel already performs better than both established methods, the results can still be improved by modifying the kernel function. Paper authors design an improved kernel function by incorporating the basic biological hypothesis that, while certain local correlations are typical for TIS, dependencies between distant positions are of minor importance or do not even exist. A. Zien and G. Ratsch want the feature space to reflect this: at each sequence position, paper authors compare the two sequences locally, within a small window of length 2l+1 around that position. They call this kernel locality-improved.

**Experiments** For each potential start codon (the nucleotide sequence ATG) on the forward strand, one data point is generated. This leads to 13,503 data

points, of which 3,312 (24.5%) represent true TIS and the rest (10,191 points, 75.5%) represent pseudo sites. Authors prefer this skewed distribution to a balanced data set, as it is a (crude) approximation to the situation that is expected for real ESTs. Each data point is represented by a sequence window of 200 nucleotides centered around the respective ATG triplet. For triplets near the borders of the available sequence, the positions missing from the 200 nucleotide window are filled with N, the symbol for unknown. Pedersen and Nielsen, who used these datasets to implement a Neural Network prediction method, divide the data into six parts of nearly equal size (circa 2200 points each) and equal fraction of true TIS. Each part is in turn reserved for testing the classification learned from the other five parts.

**Results and discussion** A. Zien and G. Ratsch have compared the performance of important methods for sequence classification on a bio-molecular problem of practical relevance, and show that SVMs are competitive to other, more frequently used machine learning methods and show a simple way to include prior knowledge to improve performance. The paper also provides evidence that their advanced TIS recognition can be of use for other existing programs, as shown in Figure 10, that lists the methods comparison results ($d$ is the degree of the polynomial).

| Algorithm | Parameter setting | Overall error |
|---|---|---|
| Neural network | | 15.4% |
| Salzberg method | | 13.8% |
| SVM, simple polynomial | $d = 1$ | 13.2% |
| SVM, locality-improved kernel | $d_1 = 4, l = 4$ | 11.9% |
| SVM, codon-improved kernel | $d_1 = 2, l = 3$ | 12.2% |
| SVM, Salzberg kernel | $d_1 = 3, l = 1$ | 11.4% |

Figure 10: Results shown in the [18] article.

## 2.5 Other applications

For sake of brevity, we renounced to explain other bioinformatics applications of SVMs. Here we report a short list of them: - Classification of genes based on their promoter regions [13]
- Recognition of true splice sites in DNA [14] [15]
- Recognition of transmembrane receptor proteins [16]
- Prediction of the cleavage site of signal peptides from the amino-acid sequence of a protein [17]
- Protein fold recognition [19][20][21]
- Protein-protein interactions prediction [22]
- Protein secondary structure prediction [23]
- Protein localization [24]
- DNA hairpin molecule analysis [25]
- Signal peptide detection [26]
- Virtual screening of small molecules [27]

# 3 SVM code

Here's a short list of the best SVM implementation code available online:
- `svm 1.0` by William Noble Grundy, University of California, Santa Cruz, CA, Usa. Developed in C. Used in [10]. `http://compbio.soe.ucsc.edu/svm.1.0/doc/svm.html`
- `SvmLight` by Thorsten Joachims, technische Universitat von Dortmund, Germany, EU. Developed in C. `http://svmlight.joachims.org/`
- `LibSvm` by Chih-Chung Chang and Chih-Jen Lin, National Taiwan University, Taiwan, Taipei. Developed in C++, or Java. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. [28]

Other are listed here: `http://www.support-vector-machines.org/SVM_`

soft.html

# References


[1] *Support-vector networks*. Corinna Cortes and Vladimir Vapnik, Machine Learning, 1995, Volume 20, Number 3, Pages 273-297.

[2] *Tikhonov regularization*, Wikipedia in English. `http://en.wikipedia.org/wiki/Tikhonov_regularization`

[3] *Duality (optimization)*, Wikipedia in English. `http://en.wikipedia.org/wiki/Dual_problem`

[4] *Mercer theorem*, Wikipedia in English. `http://en.wikipedia.org/wiki/Mercer_theorem`

[5] *DNA microarray*, Wikipedia in English. `http://en.wikipedia.org/wiki/DNA_microarray`

[6] *Knowledge-based analysis of microarray gene expression data by using support vector machines*. Michael P. S. Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Walsh Sugnet, Terence S. Furey, Manuel Ares, Jr., David Haussler. Proc. Natl. Acad. Sci. USA, vol. 97, pages 262-267

[7] *Support Vector Machine Classification of Microarray Gene Expression Data*. Michael P. S. Brown William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., David Haussler

[8] *Gene functional classification from heterogeneous data*. Paul Pavlidis, Jason Weston, Jinsong Cai and William Noble Grundy, Proceedings of RECOMB 2001

[9] *Support vector machine classification of microarray data*. S. Mukherjee, P. Tamayo, J.P. Mesirov, D. Slonim, A. Verri, and T. Poggio, Technical Report 182, AI Memo 1676, CBCL, 1999.

[10] *Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data*. Terrence S. Furey, Nigel

Duffy, Nello Cristianini, David Bednarski, Michel Schummer, and David Haussler, Bioinformatics. 2000, 16(10):906-914.

[11] *Gene Selection for Cancer Classification using Support Vector Machines.* I. Guyon, J. Weston, S. Barnhill and V. Vapnik, Machine Learning 46(1/3): 389-422, January 2002

[12] *Molecular classification of multiple tumor types.* C. Yeang, S. Ramaswamy, P. Tamayo, Sayan Mukerjee, R. Rifkin, M Angelo, M. Reich, E. Lander, J. Mesirov, and T. Golub, Intelligent Systems in Molecular Biology

[13] *Promoter region-based classification of genes.* Paul Pavlidis, Terrence S. Furey, Muriel Liberto, David Haussler and William Noble Grundy, Proceedings of the Pacific Symposium on Biocomputing, January 3-7, 2001. pp. 151-163.

[14] *Exploiting generative models in discriminative classifiers* T. Jaakkola and D. Haussler, Preprint, Dept. of Computer Science, Univ. of California, 1998.

[15] *A discrimitive framework for detecting remote protein homologies.* T. Jaakkola, M. Diekhans, and D. Haussler, Journal of Computational Biology, Vol. 7 No. 1,2 pp. 95-114, (2000)

[16] *Classifying G-Protein Coupled Receptors with Support Vector Machines.* Rachel Karchin, Master's Thesis, June 2000.

[17] *Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings.* J.-P. Vert, Pacific Symposium in Biocomputing.

[18] *Engineering support vector machine kernels that recognize translation initiation sites.* A. Zien, G. Ratsch, S. Mika, B. Scholkopf, T. Lengauer, and K.-R. Muller, BioInformatics, 16(9):799-807, 2000.

[19] *Multi-class protein fold recognition using support vector machines and neural networks*. Chris Ding and Inna Dubchak, Bioinformatics, 17:349-358, 2001

[20] *Support Vector Machines for predicting protein structural class*. Yu-Dong Cai, Xiao-Jun Liu , Xue-biao Xu, and Guo-Ping Zhou, BMC Bioinformatics (2001).

[21] *The spectrum kernel: A string kernel for SVM protein classification*. Christina Leslie, Eleazar Eskin and William Stafford Noble Proceedings of the Pacific Symposium on Biocomputing, 2002.

[22] *Predicting protein-protein interactions from primary structure*. Joel R. Bock and David A. Gough, Bioinformatics 2001 17: 455-460

[23] *A Novel Method of Protein Secondary Structure Prediction with High Segment Overlap Measure: Support Vector Machine Approac*. Sujun Hua and Zhirong Sun, Journal of Molecular Biology, vol. 308 n.2, pages 397-407, April 2001.

[24] *Support vector machine approach for protein subcellular localization prediction*. Sujun Hua and Zhirong Sun, Bioinformatics 2001 17: 721-728

[25] *Rapid discrimination among individual DNA hairpin molecules at single-nucleotide resolution using an ion channel*. Wenonah Vercoutere, Stephen Winters-Hilt, Hugh Olsen, David Deamer, David Haussler, Mark Akeson. Nature Biotechnology 19, 248 - 252 (01 Mar 2001).

[26] *Predicting Signal Peptides with Support Vector Machines*. Neelanjan Mukherjee, Sayan Mukherjee. SVM '02 Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines. Springer-Verlag London, UK 2002.

[27] P. Mah. Kernel design for virtual screening of small molecules with support vector machines. PhD thesis, Ecole des Mines de Paris, 2006.

[28] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`