Biological Data Analysis using Incremental Kernel Machines: A Comparative Study

L. Morgado$^{(1)}$, C. Pereira$^{(1,2)}$

CISUC$^{(1)}$ – Center for Informatics and Systems of the University of Coimbra
Polo II Universidade 3030-290 Coimbra, Portugal
lionel@student.dei.uc.pt, cpereira@dei.uc.pt

ISEC$^{(2)}$ - Instituto Superior de Engenharia de Coimbra
Quinta da Nora, 3030 Coimbra, Portugal
cpereira@isec.pt

Abstract

Kernel Machines, and Support Vector Machines in particular, are state-of-the-art performers in the current machine learning paradigm. However, traditional implementations work in a batch fashion, limiting its application. Recent developments conducted to incremental algorithms that further decompose the learning process, which enable the application to large scale datasets, a common feature in biological data analysis. This work presents a set of experiments in order to evaluate the performance of incremental algorithms using five biological benchmark datasets, opening the possibility to further extend these incremental approaches to large scale problems.

1. Introduction

One of the greatest scientific challenges since the last decade has been to understand how biological systems work and how to control them. Problems from biology are typically complex and difficult to solve, because they involve numerous subtle or even unknown conditionings that form a particular synergy when combined. To get answers it has been collected enormous quantities of examples that traditional laboratorial analysis techniques haven't been able to keep up. The use of computational approaches has been given a decisive help to analyze and interpret this data, expanding a new knowledge field referred as computational biology [1, 2, 3]. The Support Vector Machine (SVM) is one of such approaches, with origin from the machine learning techniques [4]. This particular kernel machine formulation has the ability to build discriminative models that combine high accuracy with good generalization, qualities that make the SVM an unavoidable tool with state-of-the-art results when dealing with the biological information, often characterized by high noise rates, abundant outliers and overlapped classes. However, large datasets and algorithms complexity are exceeding the computational capacities available, becoming limiting factors to real-life implementations. To overcome these problems several improvements have been made at the machine architecture level, appearing diverse changed SVM versions and new kernel machines. In practice, this took to further refined and sophisticated decomposition algorithms from which evolved the incremental SVMs.

To clearly present the most important mathematic concepts, the classic SVM formulation is initially introduced in the next section. In Section 3 are presented attempts to increase the performance in kernel machines and introduced the incremental algorithms. Section 4 exposes the exact incremental SVM formulation, and in Section 6 appears LASVM as a different incremental approach connected to SMO, whose fundaments are in Section 5. The comparative study includes experiments in Section 7 and the respective results in section 8. Finally, conclusions and future work are discussed in Section 9.

2. Support Vector Machines

The SVM is a successful class of learning systems strictly connected with statistical learning theory [4]. It tries to find a function *f* that minimizes the expected risk:

$$R[f] = \iint L(y, f(x))dP(y \mid x)dP(x) \quad (1)$$

, where the distribution of the examples $P(x)$ and their classifications $P(y \mid x)$ are unknown, and where *L* is a loss function that measures the error of predicting *y* with $f(x)$.

Based on this principle the SVM builds a classifier, that generally can be described by

$$f(x) = w.\phi(x) + b \ (2)$$

, for a training data set $\{(x_i, y_i) \in \Re^n \times \{-1,1\} \forall i \in \{1,...,N\}\}$.

A transformation is performed in order to change a non-linear input problem representation to a linear one in a given feature space ϕ. Mapping is implicit in a kernel function $K(x_i, x_j)$ given by the dot product

$$K(x_i, x_j) = \phi(x_i).\phi(x_j). \ (3)$$

Considering the distance of a point to the decision hyperplane, it is possible to define a margin width $\frac{2}{\|w\|^2}$, from which the optimisation process is traditionally defined by

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i^p \quad (4)$$

, with typical order values *p* = {1, 2}, and subject to

$$y_i(w\phi(x_i) + b) \geq 1 - \xi_i, \ (5)$$

$$\xi_i \geq 0, \forall i \in \{1,...,N\} \quad (6)$$

, where $C > 0$ is the trade-off parameter and ξ a parameter that minimized the upper bound of the empirical risk. The latter also gives more flexibility to a model by adding tolerance to outliers and noise in the training data set.

The mapping function isn't known in the optimisation process, so it can't be made directly for variables *w* and *b*. In the traditional SVM the Lagrangian plays an important role in this phase. This problem can then be expressed as a quadratic program (QP) in the dual form

$$\min_{0 \leq \alpha_i \leq C} W = \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i Q_{ij} \alpha_j - \sum_{i=1}^{N} \alpha_i + b\sum_{i=1}^{N} y_i \alpha_i \quad (7)$$

, where α are Lagrange coefficients and $Q_{ij} = y_i y_j \phi(x_i)\phi(x_j)$, with $\forall i \in \{1,...,N\}$ and $\forall j \in \{N\}$.

The group of points that achieve a coefficient $\alpha \neq 0$ are called support vectors.

3. Attempts to increase performance in kernel machines

Among the adapted SVMs, we can find the Proximal SVM [5]. This kernel machine methodology can be interpreted as ridge regression applied to classification problems. It works by searching the "proximal" planes, around which the points of each class are clustered and then pushes them as far apart as possible to find a good decision frontier. In the Proximal SVM the inequality constraints used in a standard SVM are replaced by equalities, so the solution can be obtained from a single linear system of linear equations**,** instead from solving the well known QP. Other algorithms such as the DirectSVM, also avoids the QP by searching the solution through an iterative algorithm based on few simple heuristics [6]. Because the QP is demanding on processing, calculations can greatly speed up for both cases. Interior-point methods [7] to solve QP problems with a small number of linear constraints, a reformulation of the standard QP incorporated in the Lagrangian SVM [8], the Cutting-Plane algorithm [9], and a proposal for non-linear training on SVM by directly minimizing the primal problem instead of maximizing the dual problem [10], make all part of the extended list of attempts to manage the complexity of the training algorithm.

There are other approaches more radical in the way they seek to decrease the model complexity. For example, in the Relevance Vector Machine (RVM), Bayes principles are used to get models with a decreased number of vectors, by choosing prototypes that use the statistical vectors information to define the decision surface [11]. This approach has also another advantage: it isn't susceptible to kernel restrictions imposed by Mercer's conditions. However its author mentions that it is slower than SVMs when data sets are large.

Despite the advances attained, none of the previously referred approaches shows a general satisfactory performance when dealing with large data sets, especially because loading numerous examples at a time demands a memory usage many times unavailable.

Since less information is associated to simpler models and lower memory requirements, data size and dimension reduction have also been explored. Training over some randomly chosen examples is the easiest way to decrease the set size, but the probability of excluding important information using this method is very high. On the other hand, a larger training set represents an advantage in the sense that the extra information obtained with more examples can contribute to create more accurate models. Therefore, it is important to analyse every individual example. In the Reduced SVM [12], it is tried to respect this criteria by using the entire dataset as a constraint in an optimisation problem, but a small rectangular kernel matrix of user pre-specified size *m*, randomly selected from the entire data set, to replace the fully dense square kernel matrix used in the non-linear support vector machine formulation. Boosting [13], squashing [14], editing [15] and clustering [16], are examples of other used techniques for size reduction.

Another way found to decrease the number of examples while searching for a solution, consists on analysing smaller blocks at each time and only then building a general solution. Chunking [17] and the Sequential Minimal Optimisation (SMO) algorithm [18] can be included in this group. While chunking only solves the QP with all points with non-zero Lagrangian multipliers, SMO avoids handling matrixes by taking the QP to the extreme of solving smaller ones of just two variables at a time. This way SMO doesn't need to use a QP optimiser and storing kernel matrices in memory. Despite needing more iterations to converge, it can speed-up several orders of magnitude due to a radical simplification of the operations executed on each iteration [18].

The need for speed, computational capacity and online algorithms leads us to incremental algorithms. As the word "incremental" says, the main idea is to add/increment new available information to a model, refreshing it. In the genesis of incremental Kernel Machines there is a method that consists in retraining a model consecutively in new blocks of data and the support vectors obtained from previous trains [33]. This approach takes under consideration that the decision surface is described by a small number of support vectors, from the examples, that are preserved along several retrains.

The exact formulation for incremental learning only appeared several years later [19], and made also possible to decrement or "unlearn" a model. The same formulation was used to minimize the computational cost of recalculating a new QP when C and kernel parameters of a SVM are changed [20]. However, its implementation presents a drawback once it uses all the already seen examples to get the final exact solution. An alternative that tries to introduce improvements in this issue is SimpleSVM [21]. This algorithm extends the principles in [19] to the soft-margin case and combines it with block training and active set methods to keep optimality over unconstrained Lagrangian multipliers. According to the authors, despite being reported as a good performer, in cases with many support vectors, SMO is preferred [19]. It is precisely from SMO that LASVM is derived. This algorithm is an online kernel classifier based on the soft-margin SVM able to incrementally build a discriminative model, either adding or removing support vectors iteratively [22].

In fact, real-life problems are dynamic/online rather than static/batch, since information is prone to change. Some work has been developed on incremental/online classification [23, 24, 25] and regression problems [26, 27, 28], but plenty of topics still untouched. To our knowledge no studies have been published about incremental multi-class machines for classification.

On what concerns solving issues connected to a large number of variables, there isn't much work done. For the Proximal SVM [29] the problem was handled without feature selection, manipulating the

Sherman-Morrison-Woodbury formula used for matrix inversion during optimisation. Authors report that this column-incremental linear proximal SVM can deal at least 10^9 attributes. Nevertheless, there isn't any kernel machine capable to apply an incremental algorithm for column and line at the same time.

To solve both the quantity and dimension problems, some authors propose to add a combination of traditional techniques like Principal Component Analysis, and Recursive Feature Elimination, to regular training algorithms [30]. The former is intended to execute space reduction and the latter redundant and non-discriminative feature elimination.

4. The exact incremental SVM

The main idea under the precise incremental algorithm is the construction of a model in an iterative fashion, analysing point by point in such a way that the Karush-Kuhn-Tucker (KKT) conditions for the points already processed still being respected:

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_j Q_{ij}\alpha_j + y_i b - 1 = y_i f(x_i) - 1 \begin{cases} \geq 0 & if \quad \alpha_i = 0 \\ = & if \quad 0 < \alpha_i < C \\ \leq 0 & if \quad \alpha_i = C \end{cases} \quad (8)$$

$$\frac{\partial W}{\partial b} = \sum_j y_j \alpha_j = 0 \quad (9)$$

These conditions divide the training data in three groups: the set S of support vectors ($g_i = 0$), the set E of error support vectors that go beyond the margin ($g_i < 0$), and the remaining set R_e of vectors within the margin ($g_i > 0$). New training data points that don't contribute to the solution are assigned to R_e, while other examples are distributed among *S* or *E*.

The training examples change the coefficients in order to keep the KKT conditions satisfied. This results in a variation of the KKT conditions given by

$$\Delta g_i = Q_{ic}\Delta\alpha_c + \sum_{j \in S} Q_{ij}\Delta\alpha_j + y_i \Delta b, \forall i \in D \in \{c\} \quad (10)$$

$$0 = y_c \Delta\alpha_c + \sum_{j \in S} y_j \Delta\alpha_j \quad (11)$$

, with α_c the coefficient to be incremented. Margin points $S = \{s_1, \dots, s_{lS}\}$ are the only ones that contribute to the new solution. Considering that $g_i = 0$, it is possible to combine equations (10) and (11) in the matrix form

$$\begin{bmatrix} 0 & y_{s1} & \cdots & y_{sl_s} \\ y_{s1} & Q_{s1s1} & \cdots & Q_{s1sl_s} \\ \vdots & \vdots & \ddots & \vdots \\ y_{sl_s} & Q_{sl_s s1} & \cdots & Q_{sl_s sl_s} \end{bmatrix} \cdot \begin{bmatrix} \Delta b \\ \Delta\alpha_{sl} \\ \vdots \\ \Delta\alpha_{sl_s} \end{bmatrix} = - \begin{bmatrix} y_c \\ Q_{slc} \\ \vdots \\ Q_{sl_s c} \end{bmatrix} \Delta\alpha_c \quad (12)$$

We are interested in knowing the variations of α and b, becoming

$$\begin{bmatrix} \Delta b \\ \Delta\alpha_{s1} \\ \vdots \\ \Delta\alpha_{sl_s} \end{bmatrix} = - \begin{bmatrix} 0 & y_{s1} & \cdots & y_{s1_s} \\ y_{s1} & Q_{s1s1} & \cdots & Q_{slsl_s} \\ \vdots & \vdots & \ddots & \vdots \\ y_{sl_s} & Q_{sl_s sl} & \cdots & Q_{sl_s sl_s} \end{bmatrix} \cdot \begin{bmatrix} y_c \\ Q_{slc} \\ \vdots \\ Q_{sl_s c} \end{bmatrix} \Delta\alpha_c \quad (13)$$

, substituting

$$R = \begin{bmatrix} 0 & y_{s1} & \cdots & y_{sl_s} \\ & & \cdots & Q_{slsl} \\ \vdots & \vdots & \ddots & \vdots \\ y_{sl_s} & Q_{sl_s s1} & \cdots & Q_{sl_s sl_s} \end{bmatrix}^{-1} \quad (14)$$

and considering the coefficient sensitivities *β*, comes

$$\begin{bmatrix} \beta \\ \beta_{s1} \\ \vdots \\ \beta_{sl_s} \end{bmatrix} = -R \cdot \begin{bmatrix} y_c \\ Q_{slc} \\ \vdots \\ Q_{sl_s c} \end{bmatrix} \quad (15)$$

The application of this last equation to equation (10), gives the margin variation

$$\Delta g_i = \left(Q_{ic} + \sum_{j \in S} Q_{ij}\beta_j + y_j \beta \right) \Delta\alpha_c = \gamma_i \Delta\alpha_c \quad (16)$$

, where margin sensitivities $\gamma_i = 0$ for all i elements in *S*.

Through *β* is possible to calculate $\Delta\alpha$ and Δb, and *γ* allows to determinate Δg.

4.1. Accounting

Changes in $\Delta\alpha$ and Δb can modify the constitution of *S*, *E* and/or R_e sets, so the previous mathematical relations are valid only in the case where $\Delta\alpha_c$ is small enough not to provoke such phenomenon. It is then necessary to calculate the maximum increment $\Delta\alpha_c$ possible to transfer some points from R_e to *S*.

The maximum increment of $\Delta\alpha_c$ allowed, depends on:

1. $g_c \leq 0$, with equality when *c* joins S;
2. $\alpha_c \leq C$, with equality when *c* joins *E*;
3. $0 \leq \alpha_j \leq C$, with equality 0 when *j* transfers from *S* to R_e, and equality *C* when *j* transfers from *S* to *E*;
4. $g_i \leq 0$, with equality when *i* transfers from *E* to *S*;
5. $g_i \geq 0$, with equality when *i* transfers from R_e to *S*.

If *c* is a new support vector, it can be added expanding the *R* matrix as

$$R \leftarrow \begin{bmatrix} & & & 0 \\ & R & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\gamma_c} \begin{bmatrix} \beta \\ \beta_{s1} \\ \vdots \\ \beta_{sl_s} \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \beta \\ \beta_{sl} \\ \vdots \\ \beta_{sl_s} \\ 1 \end{bmatrix}^T \quad (17)$$

This procedure is reversible leading to a decremental algorithm, to which corresponds the matrix contraction

$$R \leftarrow R_{ij} - \frac{R_{ik} R_{kj}}{R_{kk}} \qquad \forall i, j \in S \bigcup \{0\}; i, j \neq k \quad (18)$$

5. Sequential Minimal Optimization (SMO)

There are two methods typically used to solve the QP: the Conjugate Gradient [17] and Sequential Minimal Optimization [18].

Both algorithms start from a vector α and extends the optimization process through a determined direction *u*, that guides to a vector $\alpha + \lambda * u$, where

$$\lambda^* = \arg\max W(\alpha + \lambda u) \text{ with } 0 \le \lambda \le \phi(\alpha, u). \ (19)$$

The upper bound $\phi(\alpha, u)$ ensures that $\alpha + \lambda u$ is feasible as well:

$$\phi(\alpha,u) = \min \begin{Bmatrix} 0 & if \quad \Sigma_k u_k \neq 0 \\ (B_i - \alpha_i)/u_i & for \quad all \quad i \quad such \quad that \quad u_i > 0 \\ (A_j - \alpha_j)/u_j & for \quad all \quad j \quad such \quad that \quad u_j < 0 \end{Bmatrix} (20)$$

The optimal value comes from

$$\lambda^* = \min \left\{ \phi(\alpha,u), \ \frac{\Sigma_i g_i u_i}{\Sigma_{i,j} u_i u_j K_{ij}} \right\} (21)$$

where $K_{i,j} = K(x_i, x_j)$ and $g = (g_1, ..., g_n)$ is the gradient of $W(\alpha)$, and

$$g_k = \frac{\partial W(\alpha)}{\partial \alpha_k} = y_k - \sum_i \alpha_i K(x_i, x_k) = y_k - \hat{y}(x_k) + b \quad (22)$$

This direction search is dramatically speeded up when most coefficients are 0. SMO takes only one +1 and one –1 example as it tries to find $\Sigma_k u_k = 0$.

Implementations take into consideration a small positive tolerance τ, that helps to select the direction search *u*. The directions selected have $\phi(\alpha,u) > 0$ and $u' g > \tau$, which allows to increase $W(\alpha)$ before reaching a constraint. These directions are defined by a pair of points called τ-violating pair, for which

$$(i, j) \quad \tau - violating \quad pair \Leftrightarrow \begin{cases} \alpha_i < B_i \\ \alpha_j > A_j \\ g_i - g_j > \tau \end{cases} (23)$$

The τ-violating pair usually chosen is the one that maximizes the directional gradient $u' g$. When no more pairs remain the process finishes and it is obtained an approximated solution. This solution that is associated with a finite number of iterations gets near to the exact solution as τ becomes close to 0.

6. LASVM: an online SMO based SVM

LASVM is an incremental kernel classifier based on the soft-margin SVM [22]. This algorithm builds a discriminative model adding or removing support vectors iteratively through a method strictly connected with SMO. Each turn two candidate points are analysed. New support vectors come from a direction search called PROCESS that involves at least one non support vector from the current kernel expansion, while a method called REPROCESS eliminates meaningless support vectors by changing to 0 the coefficients of one or both the analysed points. Each iteration demands storing a set of all the potential support vectors, coefficients α_i of the kernel expansion and the partial derivatives g_i in order to incrementally build the final discriminative model. Like it is done in SMO, the algorithm stops when a τ-approximated solution is reached. Despite the precise solution being attainable only after running the algorithm for several epochs, LASVM can reach similar accuracy values to exact methods after a single pass over the data. Nevertheless a finishing step similar to a simplified SMO, may be necessary to improve performance on noisy data sets when shortening the number of epochs.

A significant difference that arises when comparing LASVM to SimpleSVM is that the former doesn't seek the precise solution of the QP problem in each step but instead an approximation that improves the dual function $W(\alpha)$. This has an impact in the processing time: while SimpleSVM uses rank 1 matrix updates whose computation cost grows as the square of the number of support vectors, LASVM SMO based updates grow linearly with the number of examples.

LASVM also implements strategies to speed up calculations using a stopping criterion that avoids solving the optimisation problem to a point of higher accuracy than necessary**.**

Searching for informative examples can further increase processing speed when several examples are already available (in offline training or when the algorithm can't follow up the acquisition speed).

LASVM presents active selection rooted in the minimax gradient

$$\arg\min_{k\notin S}\max_{y=\pm 1} y\hat{y}(x_k) = \arg\min_{k\notin S}\left|\hat{y}(x_k)\right| \quad (24)$$

, that chooses the example localized closest to the current decision frontier (higher value), from a small randomly selected group that hasn't yet been PROCESSED. This step enhances tolerance to noisy data since it is executed independently of the point's label.

7. Experiments

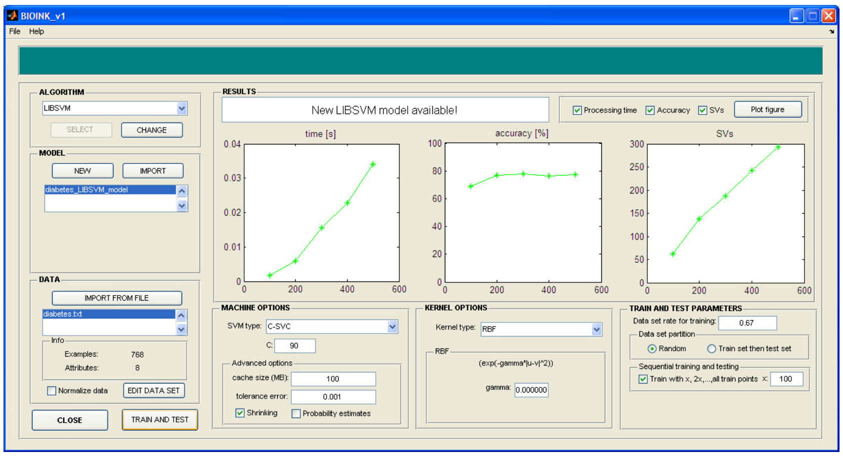
Results reported here were obtained from a performance study of several kernel machines on biological benchmark data sets. They include parameters such as accuracy, processing time and the complexity of the trained models, for binary and multi-class classification tasks.

To accomplish the study a Matlab application with a user interface has been created, which integrates all algorithms used here in a single tool. This first version incorporates functionalities that facilitate data manipulation and the analysis of results.

The algorithms used here were: LIBSVM (version 2.85), the exact incremental algorithm from Cauwenberghs and Poggio (*C&P*), SimpleSVM (version 2.31) and the Relevance Vector Machine (version 1.00). Discriminative models were trained under a 1.6 GHz processor PC and Microsoft Windows XP running MATLAB 7.2 (R2006a). The incremental algorithm LASVM was also tested, but not included in this tool since the original code was not developed for Windows. It was then executed under gOS, an operating system that belongs to the Linux family.

All algorithms were tested on the binary problems, but only LIBSVM and SimpleSVM worked on multi-class, since the other machines weren't prepared for it. The architecture of LIBSVM and SimpleSVM also don't cope with multi-class classification, nevertheless a one-versus-one methodology is available in both applications.

It is important to keep in mind that this is a comparative study where we emphasize the relation between the algorithms. Despite indicating processing times, they shell not be seen as inflexible variables, since there were several aspects of the computational environment that were not totally controlled.



LIBSVM

diabetes_LIBSVM_model

diabetes.txt

Info
Examples: 768
Attributes: 8

☐ Normalize data

New LIBSVM model available!

☑ Processing time ☑ Accuracy ☑ SVs

SVM type: C-SVC
C: 90
Advanced options
cache size (MB): 100
tolerance error: 0.001
☑ Shrinking ☐ Probability estimates

Kernel type: RBF
RBF
(exp(-gamma*|u-v|^2))
gamma: 0.000000

Data set rate for training: 0.67
Data set partition
[✓ Random] Train set then test set
Sequential training and testing
☑ Train with x, 2x,...,all train points x: 100

Figure 1 – Screenshot of the Matlab tool developed.

7.1. Data sets description

All data sets are under public domain, and can be found at UCI repository homepage [31]. Experiments included 3 binary and 2 multiclass classification tasks. Pima Indians Diabetes, Wisconsin Breast Cancer and Promoters were used in the first kind of task, and Yeast and Splice in the latter.

Table 1 – Brief data sets description

Data set	# Attributes	# Instances	Type	Description
WBC	10	699	Binary classification	Attributes are cell properties used to identify between a benign and a malign class.
PID	8	768	Binary Classification	Information from several female patients with at least 21 years old of Pima Indian heritage, used to determine diabetes.
Promoters	58	106	Binary classification	A promoter is a region of DNA that facilitates the transcription. To check if a nucleotide sequence is a promoter, this data set was transformed from the original string format to numbers using the available functions from the Matlab bioinformatics toolbox.
Yeast	8	1484	Multi-class classification	The problem posed in this dataset is to recognize, the localization site of a given yeast known some of its properties. Every string attribute was transformed to a numeric format.
Splice	61	3190	Multi-class classification	Splice junctions are points on a DNA sequence at which 'superfluous' nucleotides are removed during the process of protein creation. The objective is to detect the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out), and vice-versa.

7.2. Training Parameters

Parameters used for training were obtained empirically. They are the ones by which was obtained the higher accuracy values for each data set.

Training used the first 2/3 of the data, and the remaining 1/3 was used for testing. Exceptionally, data set Promoters used points with an index multiple of 2 and 3 for training and the remaining for testing, in order to place examples from all classes among train and test collections.

To get more details about the learning process, the number of training points was gradually increased.

Table 2 – Classification parameters

Binary classification

	Training size	*Test size*	*Γ*	*C*	*Kernel type*
PID	512	256	0.0000007	90	RBF
WBC	466	233	0.07	10	RBF
Promoters	71	35	0.04	5	RBF

Multi-class classification

	Training size	*Test size*	*Γ*	*C*	*Kernel type*
Yeast	989	495	0.3	110	RBF
Splice	2127	1063	0.0009	70	RBF

8. Results

Results concerning the accuracy, processing time and model complexity are graphically presented for each data set (annex A).

A consensual result concerns processing time for binary classification: LIBSVM and LASVM are far the fastest applications, followed by C&P algorithm, SimpleSVM, and finally the RVM.

The relation between accuracy values and the number of vectors kept in the discriminative RVM model for Diabetes and Promoters contrast to the ones obtained for WBC. In the first two cases considerably simpler models than the ones constructed by the other approaches, could obtain similar or even higher accuracy results, while for WBC set the RVM got worst accuracy despite using even more examples to define the decision hyperplane.

LASVM and LIBSVM have a similar behaviour for the smaller data sets. However the former is demarked since it keeps less SVs as data sets grow, building discriminative models with lower complexity.

The *C&P*, LASVM and RVM algorithms weren't used with Yeast and Splice because the software packages weren't prepared for these multi-class data sets. In this case LIBSVM performed better than SimpleSVM, achieving the best results for both data sets: The accuracy rates are similar for a shorter training time.

Table 3a – General results

	LASVM			**C&P**			**SimpleSVM**		
DATA SET	**Accuracy (%)**	**Time (s)**	**SVs**	**Accuracy (%)**	**Time (s)**	**SVs**	**Accuracy (%)**	**Time (s)**	**SVs**
PID	79,7	immediate	256	79,7	1,8	307	80,5	3,6	303
WBC	99	immediate	47	98,7	0.5	56	98,7	0.4	55
Promoters	83.3	immediate	70	72,2	0,2	32	83,3	0,3	68
Yeast	-	-	-	-	-	-	54,6	16,9	1349
Splice	-	-	-	-	-	-	87.3	327	2107

Table 3b – General results (cont.)

	LIBSVM			**RVM**		
DATA SET	**Accuracy (%)**	**Time (s)**	**SVs**	**Accuracy (%)**	**Time (s)**	**RVs**
PID	79,3	immediate	311	76,2	24	14
WBC	98,7	immediate	55	82	450	78
Promoters	91,4	immediate	70	97,2	0,8	16
Yeast	58.5	0,25	672	-	-	-
Splice	86	1.8	924	-	-	-

9. Conclusion and Future Work

In this work, several incremental approaches for the SVM were presented and evaluated on five biological datasets. Among the used kernel machines, LIBSVM and LASVM have given, in general, the best results, confirming the potential of the SMO based techniques. Also, the complexity of a model tends to increase as the number of examples provided becomes significantly larger.

LASVM combines active learning with a decremental technique in order to optimise the complexity *vs.* accuracy trade-off. The positive effect of such approach is expressed in the achieved results, where LASVM generates less support vectors than LIBSVM, for the same accuracy.

In this preliminary analysis, it was also verified that RVMs are slower than the other options available. However, considering the fact that there were data sets like promoters and diabetes where it was able to create considerably simpler models without significant loss of accuracy or even achieving higher values.

The *C&P* incremental algorithm and its counterpart SimpleSVM, have shown a similar performance. Their processing time is larger than SMO based techniques, but the results concerning to the accuracy and number of support vectors were similar.

As an overall conclusion, there is still plenty of work to do on incremental algorithms in order to exploit all their potential. Future work will be directed toward algorithms similar to LASVM, exploiting the combination of feature selection methods and space reduction with incremental learning. Principal Component Analysis based techniques, recursive feature selection and active learning should be considered.

10. References

[1] Jaakkola, T.; Diekhans, M.; Haussler, D.; A discriminative framework for detecting remote protein homologies. *J. Comput. Biol.*, **7**, 95–114, 2000;

[2] Demuth, J. P.; Bie, D. T.; Stajich, J., E.; Cristianini, N.; Hahn., M. W.; *The Evolution of Mammalian Gene Families*; PLoS ONE vol. 1, No e85, December 2006;

[3] Unger, R.; Moult, J.; *Genetic algorithms for protein folding simulations*; Journal of Molecular Biology 231: 75-81, 1993;

[4] Vapnik, V.; *Statistical Learning Theory*; Wiley, New York, 1998;

[5] Fung, G.; Mangasarian, O.; Proximal Support Vector Machine Classifier; Proceedings KDD-2001, Knowledge discovery and Data Mining, San Francisco, 2001;

[6] Roobaert, D.; *DirectSVM: A Fast and Simple Support Vector Machine Perceptron*; Neural Networks for Signal Processing X—Proceedings of the 2000 IEEE Workshop (pp. 356–365). New York: IEEE;

[7] Ferris, M.; Munson, T.; *Interior-point methods for massive for support vector machines*; SIAM J. OPTIM. Vol. 13, No. 3, pp. 783–804, 2003;

[8] Mangasarian, O. L.; Musicant, D. R.; *Lagrangian Support Vector Machines*; Journal of Machine Learning Research 1, 2001;

[9] Joachims, T.; *Training Linear SVMs in Linear Time*; KDD'06, 2006;

[10] Chapelle, O.; *Training a Support Vector Machine in the Primal*; 2006;

[11] Tipping, M. E.; *The Relevance Vector Machine*, Advances in Neural Information Processing Systems, San Mateo, CA, 2000;

[12] Lee, Y.-J.; Mangasarian, O. L.; *RSVM: Reduced support vector machines*. Technical Report 00-07, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, July 2000. Proceedings of the First SIAM International Conference on Data Mining, Chicago, April 5-7, 2001;

[13] Pavlov, D.; Mao, J.; Dom, B. ; *Scaling-up Support Vector Machines Using Boosting Algorithm*; 15th International Conference on Pattern Recognition, Volume 2 p. 2219, 2000;

[14] Pavlov, D.; Chudova, D.; Smyth, P.; *Towards Scalable Support Vector Machines using Squashing*; 2000;

[15] Bakir, G. H.; Bottou, L.; Weston, J.; *Breaking SVM Complexity with Cross-Training*; Advances in Neuronal Information Processing Systems 17, Cambridge, 2005;

[16] Boley, D.; Cao, D.; *Training Support Vector Machine using Adaptive Clustering*; University of Minnesota, 2003;

[17] Vapnik, V. N.; *Estimation of Dependences Based on Empirical Data*; New York, NY: Springer-Verlag, 1982.

[18] Platt, J. C.; *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*; Advance sin Kernel Methods – Support Vector Learning, MIT Press, 1999;

[19] Cauwenberghs, G.; Poggio, T.; *Incremental and Decremental Support Vector Machine Learning*; Advances in Neural Information Processing Systems 13, MIT Press, 2001;

[20] Diehl, C. P.; Cawenberghs, G.; *SVM Incremental Learning, Adaptation and Optimization*; Proceedings of the International Joint Conference on Neural Networks, 2003;

[21] Vishwanathan, S. V. N. ; Smola, A. J. ; Murty, M. N. ; *SimpleSVM*; Proceedings of the Twentieth International Conference on Machine Learning, Washington DC, 2003;

[22] Bordes, A. ; Ertekin, S. ; Weston, J. ; Bottou, L. ; *Fast Kernel Classifiers with Online and Active Learning*; Journal of Machine Learning Research, 2005;

[23] Tax, D. M. J.; Laskov, P.; *Online SVM Learning: From Classification to Data Description and Back*; Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop, pp. 499-508m, 2003;

[24] Laskov, P.; Gehl, C.; Krüger, S.; Müller, K. - R.; *Incremental Support Vector Learning: Analysis, Implementaton and Applications*; The Journal of Machine Learning Research 7**,** 1909 - 1936, 2006**;**

[25] Fine, S.; Scheinberg, K.; *Incremental Learning and Selective Sample via Parametric Optimization Framework for SVM*; Advances in Neural Information Processing Systems, 2001;

[26] Martin, M.; *On-line Support Vector Machine Regression*; Proceedings of the 13th European Conference on Machine Learning, 2002;

[27] Parrella, F. ; Online Support Vector Regression – A thesis presented for the degree of Information Science; Department of Information Science, University of Genoa, Italy, 2007;

[28] Ma, J. ; Theiler, J. ; Perkins, S. ; *Accurate On-line Support Vector Regression*; Neural Computation, 15, 2683-2703, Massachussets Institute of Technology, 2003;

[29] Proximal SVM home page: http://www.cs.wisc.edu/dmi/svm/psvm/ (last visit: 19/4/2008);

[30] Lei, H.; Govindaraju, V. ; *Speed Up Multi-class SVM Evaluation by PCA and Feature Selection*;

[31] UCI Page: http://archive.ics.uci.edu/ml/ (last visit: 4 May 2008);

[32] Syed, N. A. ; Liu, H. ; Sung, K. K. ; *Incremental Learning with Support Vector Machines*; 1999;

[33] Chang, C. C.; Lin, C. J.; LIBSVM: a Library for Support Vector Machines, 2004. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm;

[34] Liu, Q.; He, Q.; Shi, Z.; *Incremental Nonlinear Proximal Support Vector Machines*; Springer-Verlag Berlin Heidelberg, 2007;

[35] Do, T. ; Poulet, F. ; *Incremental SVM and Visualization Tools for Bio-medical Data Mining*; Proceedings of the European Workshop on Data Mining and Text Mining for Bioinformatics, 2003;

Annex A – Graphical Results

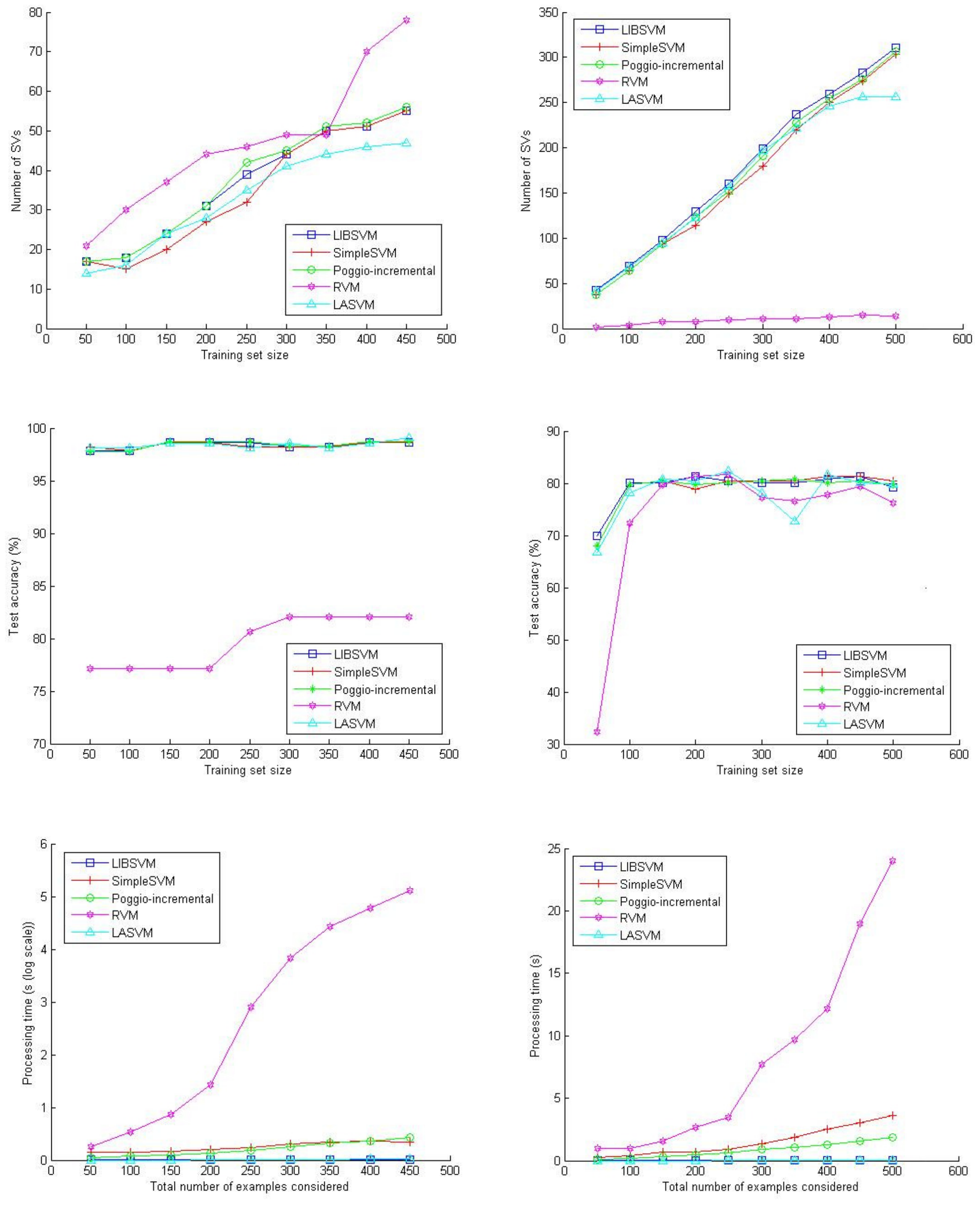


Figure 1 – Results for WBC (left column) and for Diabetes data set (right column). From top to bottom: model complexity, accuracy and processing time.

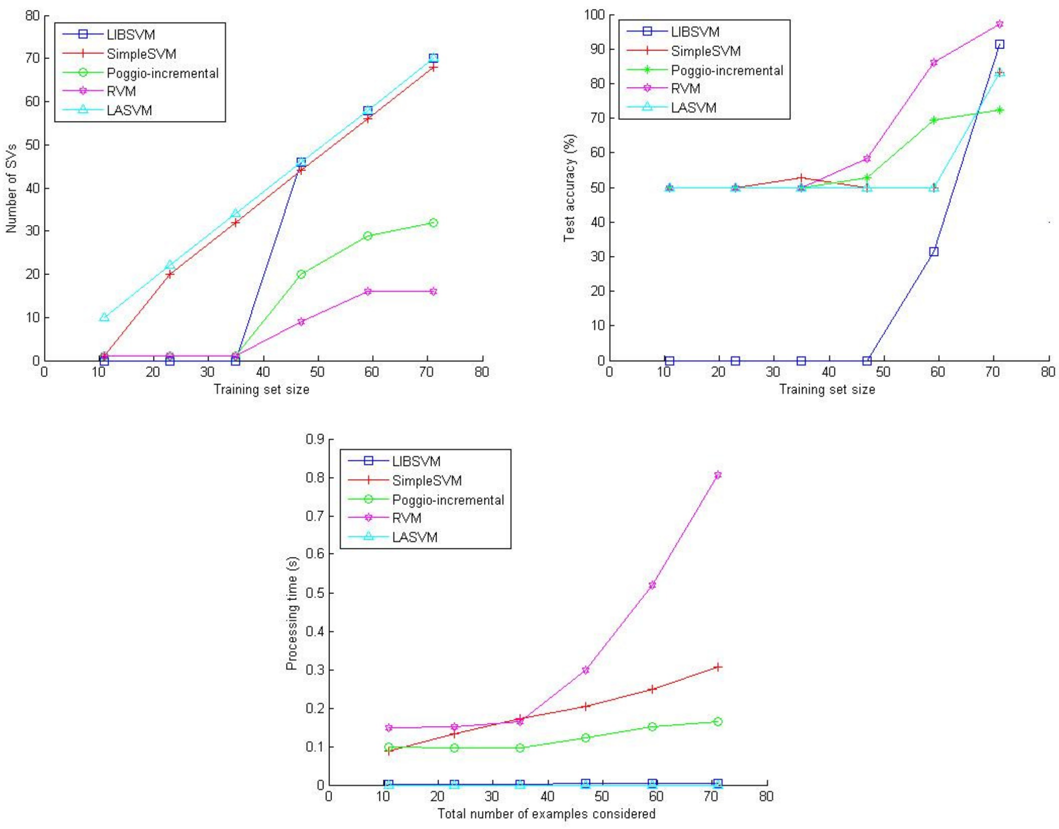


Figure 2 – Results for promoters data set: number of SVs by set size (top left), test accuracy (top right) and processing time (bottom).

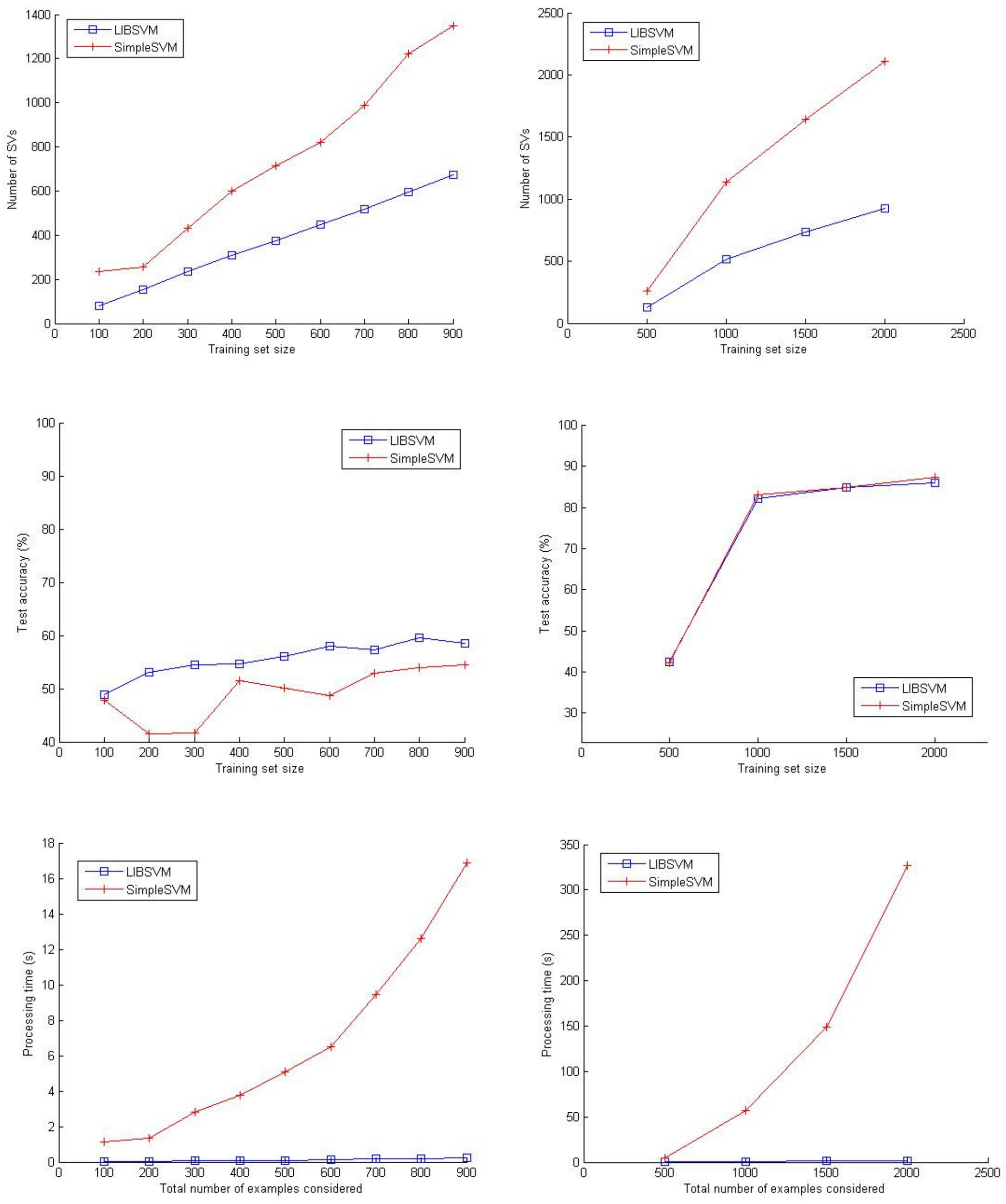


Figure 3 – Results for Yeast (left column) and for Splice data set (right column). From top to bottom: model complexity, accuracy and processing time.