Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Incremental Classification with Generalized Eigenvalues

M.R. Guarracino – C. Cifarelli – O. Seref – P.M. Pardalos - S. Cuciniello

RT-ICAR-NA-06-12 **giugno 2006**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Incremental Classification with Generalized Eigenvalues

M.R. Guarracino[1] S. Cuciniello1 – C. Cifarelli[2] – O. Seref[3] – P.M. Pardalos[3]

Rapporto Tecnico N.:
RT-ICAR-NA-06-12

Data:
giugno 2006

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli
[2] University of Rome La Sapienza
[3] University of Florida

Incremental Classification with Generalized Eigenvalues [*]

C. Cifarelli[1], M. R. Guarracino[2], O. Şeref[3],
S. Cuciniello[2], P. M. Pardalos[3,4,5] [†]

[1] Department of Statistic, Probability and Applied Statistics,
University of Rome “La Sapienza”, Italy

[2] High Performance Computing and Networking Institute,
National Research Council, Italy

[3] Department of Industrial and Systems Engineering
[4] Department of Biomedical Engineering,
[5] McKnight Brain Institute,
University of Florida, Gainesville, FL, 32611 USA

Abstract

Supervised learning techniques are widely accepted methods to analyze data for scientific and real world problems. Most of these problems require fast and continuous acquisition of data, which are to be used in training the learning system. Therefore, maintaining such systems updated may become cumbersome. Various techniques have been devised in the field of machine learning to solve this problem. In this study, we propose an algorithm to reduce the training

[†] Corresponding author: Email: pardalos@cao.ise.ufl.edu Fax: +1 (352) 392-3537

[*] This work is partially supported by National Institute of Health, Italian National Research Council and Centro Ricerche *Enrico Fermi*

data to a substantially small subset of the original training data to train a generalized eigenvalue classifier (GEC). The proposed method provides a constructive way to understand the influence of new training data on an existing classification function. We show through numerical experiments that this technique prevents the overfitting problem of the earlier GEC methods, while promising a comparable performance in classification with respect to the state-of-the-art classification methods.

1 Introduction

Supervised learning refers to the capability of a system to learn from a set of input/output couples, which is called the *training set*. The trained system is able to provide an answer (output) for a new question (input). The term *supervised* originates from the fact that the desired output for the training set of points is given by an external teacher.

Supervised learning systems can find applications in many fields, some of which can be listed as follows. A bank prefers to classify customer loan requests as "good" or"bad" depending on their ability to pay back. The Internal Revenue Service endeavors to identify tax evaders by studying the characteristics of known ones. There are also many applications in biology and medicine. The tissues that are prone to cancer can be detected with high accuracy. New DNA sequences or proteins can be tracked down to their evolutionary origins. Given its amino acid sequence, finding how a protein folds provides important information on its expression level [6]. More examples related to numerical interpolation, handwriting recognition and Montecarlo methods for numerical integration can be found, for example, in [7, 10].

Support Vector Machine (SVM) algorithms [32] are the state-of-the-art for the existing classification methods. SVMs have been one of the most successful methods in supervised learning with applications in a wide spectrum of research areas, ranging from pattern recognition [18] and text categorization [14] to biomedicine [4, 25], brain-computer interface [9, 17], and financial applications [13, 31]. These methods classify the points from two linearly separable sets in two classes, in order to find an optimal separating hyperplane between two classes. This hyperplane maximizes the distance from the convex hulls of each class. SVMs can be extended to the nonlinear cases by embedding the data in a nonlinear space using *kernel functions* [30].

In general, the training part of SVM algorithm relies on the optimization of a quadratic convex cost function. Quadratic Programming (QP) is an extensively studied field of mathematics and there are many general purpose methods to solve QP problems such as quasi-Newton, primal-dual, and interior-point methods. However, the general purpose methods are suitable for small size problems, whereas for large problems, chunking subset selection [26] and decomposition [27] methods use subsets of points. SVM-Light

[15] and LIBSVM [12] are among the most preferred implementations that use chunking subset selection and decomposition methods efficiently.

There are also efficient algorithms that exploit the special structure of a slightly different optimization problem, such as Generalized Proximal SVMs (GEPSVM) [22], in which the binary classification problem can be formulated as a generalized eigenvalue problem. This formulation differs from SVMs since, instead of finding one hyperplane that separates the two classes, it finds two hyperplanes that approximate the two classes. The prior study requires the solution of two different eigenvalue problems, while a classifier that uses a new regularization technique, known as Regularized General Eigenvalue Classifier (ReGEC) requires the solution of a single eigenvalue problem to find both hyperplanes [11].

Classification problems may involve a large number of training points. One immediate solution is to select a subset of points that would retain the characteristics of the training set. A second problem arises when a new training data point becomes available for training. A desirable method as a solution to the second problem should be based on an efficient evaluation of how the new point may influence the classification function, rather than a complete training of the incrementally augmented training set.

Datasets in almost every application area are ever growing and are continuously updated. Moreover, numerous applications on massive datasets are emerging [1], which require efficient computational procedures to respond to the dynamics of large databases. As machine learning becomes a part of data intensive computation systems, updating the learning system becomes intractable in many cases. Therefore, incremental methods that require some minimal computational burden are strongly preferred. For this purpose several methods, especially in the kernel-based nonlinear classification cases, have been proposed to reduce the size of the training set, and thus, the related kernel [5, 8, 19, 20, 28]. All of these methods show that a sensible data reduction is possible while maintaining a comparable level of classification accuracy.

In this study, a new method that finds a small subset of the training dataset is introduced. The amount of reduction in the training set can be as large as 98% with comparable classification accuracy and improved consistency with respect to the original training set. The proposed subset selection method starts with an initial set of points and incrementally expands this set by adding those points which contribute to improving classification accuracy. The main idea is to use the small subset of points to solve the general eigenvalue problem, and therefore the evaluation of the contributions for new points is performed in conjunction with ReGEC. Thus, we refer to our method as *Incremental ReGEC* (I-ReGEC).

The notation used in the paper is as follows. All vectors are column vectors, unless transposed to row vectors by a prime $'$. Scalar product of two vectors x and y in $\mathbb{R}^n$ will be denoted by $x'y$, 2-norm of x will be denoted by $\|x\|$ and the unit vector will be denoted by e.

The remainder of the the paper is organized as follows. Section 2 describes how the generalized eigenvalue classifier (GEC) methods differ from the generic SVM methods. In Section 3 the subset selection technique is presented. In Section 4, a discussion on how initial points influence the accuracy and stability of resulting classification is given. In section 5 numerical experiments are reported, and finally, in Section 6, conclusions are drawn and future work is proposed.

2 Kernel classification algorithms

The SVM method for classification consists of finding a hyperplane that separates the elements belonging to two different classes. The separating hyperplane is usually chosen to maximize the *margin* between the two classes. The margin can be defined as the minimum distance between the separating hyperplane and the points of either class. The points that are closest to the hyperplane are called *support vectors*, and they are the only points needed to train the classifier. Consider two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{k \times m}$, that represent the two classes, each row being a point in the input space. The quadratic linearly constrained problem to obtain the optimal hyperplane (w, b) is as follows.

$$
\begin{aligned}
& \min f(w) = \frac{w'w}{2} && (1)\\
s.t. \quad & (Aw + b) \geq e \\
& (Bw + b) \leq -e.
\end{aligned}
$$

In case of nonlinearly separable datasets, SMVs can take advantage of kernel techniques to achieve greater separability among classes. In this case, the initial sets of points, which originally reside in the *input space*, are nonlinearly transformed into a space of greater dimension, and the optimal separating hyperplane is found in this transformed space called the *feature space*. This nonlinear mapping can be done implicitly by kernel functions [29], which represent the inner product of the points in the feature space. In this study we use the *Gaussian kernel*,

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma}}. \tag{2}$$

In (2), x_i and x_j denote two points in the original input space. This technique usually obtains better results, as shown in several applications. Results regarding nonlinearly separable problems [2, 3] still hold and a formulation for the eigenvalues problem can easily be derived. The nonlinear implicit mapping is done through the *kernel matrix* $K(A, B)$, whose elements are defined as:

$$K(A, B)_{i,j} = e^{-\frac{\|A_i - B_j\|^2}{\sigma}}, \tag{3}$$

where A_i and B_j are the i^{th} and j^{th} rows of the matrices A and B, respectively.

Mangasarian et al. [22] proposes to classify these two sets of points A and B using two hyperplanes in the input space, each closest to one set of points, and furthest from the other. In order to satisfy the previous condition for the points in A, the two hyperplanes

$$K(x, C)u_1 - \gamma_1 = 0, \quad K(x, C)u_2 - \gamma_2 = 0, \tag{4}$$

need to be the closer to one set of points and the farther from the other. This condition can be written for the first hyperplane as:

$$\min_{\omega,\gamma\neq 0} \frac{\|K(A, C)u - e\gamma\|^2}{\|K(B, C)u - e\gamma\|^2}. \tag{5}$$

Substituting the following,

$$G = [K(A, C) \ \ - e\gamma]'[K(A, C) \ \ - e\gamma]$$

$$H = [K(B, C) \ \ - e\gamma]'[K(B, C) \ \ - e\gamma],$$

equation (5) can be rewritten as

$$\min_{z\neq 0} \frac{z'Gz}{z'Hz}, \tag{6}$$

with $z' = [u' \ \gamma]$. This is the Rayleigh quotient of the generalized eigenvalue problem $Gz = H\lambda z$. Since H is positive definite, the stationary points of

(6) are achieved at the eigenvectors with the objective function equal to the corresponding eigenvalue. This means that the solution to (6) is obtained at the eigenvector with the the minimum eigenvalue.

To obtain the second hyperplane, we need to solve a problem in which the objective function is the reciprocal of (6). It is well known that this problem has the same eigenvectors of the original problem with eigenvalues which are reciprocals of the corresponding eigenvalues from the first problem. Therefore, the eigenvector $z' = [u' \ \gamma]$ related to the maximum eigenvalue of (6) provides the solution to the latter problem, and thus gives the coefficients of the second hyperplane.

Since the matrices G and H can be deeply rank deficient, there is the possibility that the null spaces of the two matrices have a non trivial intersection. This leads to a problem that can be ill-conditioned and therefore a regularization technique needs to be applied in order to numerically solve the problem.

Mangasarian proposes to solve the following two regularized optimization problems, where $C^T = \begin{bmatrix} A^T & B^T \end{bmatrix}$ and δ is the regularization parameter:

$$\min_{w,\gamma \neq 0} \frac{\|K(A,C)u - e\gamma\|^2 + \delta \| \left[\begin{smallmatrix} u \\ \gamma \end{smallmatrix} \right] \|^2}{\|K(B,C)u - e\gamma\|^2} \tag{7}$$

and

$$\min_{w,\gamma\neq 0} \frac{\|K(B,C)u - e\gamma\|^2 + \delta \| \left[\begin{smallmatrix} u \\ \gamma \end{smallmatrix} \right] \|^2}{\|K(A,C)u - e\gamma\|^2}. \tag{8}$$

The number of eigenvalue problems can be reduced from two to one, using the new regularization method *ReGEC*, proposed by Guarracino et al. [11], by solving the following generalized eigenvalue problem:

$$\min_{w,\gamma\neq 0} \frac{\|K(A,C)u - e\gamma\|^2 + \delta\|\tilde{K}_B u - e\gamma\|^2}{\|K(B,C)u - e\gamma\|^2 + \delta\|\tilde{K}_A u - e\gamma\|^2}. \tag{9}$$

Here $\tilde{K_A}$ and $\tilde{K_B}$ are diagonal matrices with the diagonal entries from the kernel matrices $K(A,C)$ and $K(B,C)$. The new regularization provides classification accuracy results comparable to the ones obtained by solving equations (7) and (8).

The eigenvectors related to minimum and maximum eigenvalues obtained from the solution of (9) provide the proximal planes P_i, $i = 1,2$ to classify the new points. The distance of a point x from hyperplane P_i is:

$$dist(x, P_i) = \frac{\|K(x,C)u - \gamma\|^2}{\|u\|^2}, \tag{10}$$

and the class of a point x is determined as

$$class(x) = argmin_{i=1,2}\{dist(x, P_i)\}. \tag{11}$$

3 Incremental subset selection algorithm

The dimension of generalized eigenvalue problem (9) is equal to $n + k$, the number of points in the training set, plus 1. Since the computational complexity of the operation is in the order of $O((n + k)^3)$, it is important to develop methods that are capable of finding a small and robust set of points that retains the characteristics of the entire training set and provides comparable accuracy results. A kernel built from a smaller subset is computationally more efficient in predicting new points compared to kernels that use the entire training set. Furthermore, a smaller set of points reduces the probability of over-fitting the problem. Finally, as new points become available, the cost of retraining the algorithm decreases if the influence of the new points on the classification function is only evaluated by the small subset, rather than the whole training set. The main idea is to exploit the efficiency of solving a small eigenvalue problem. Therefore, we use ReGEC as the internal method to evaluate the classification accuracy on the entire training set.

The algorithm takes an initial set of points C^0 and the entire training set C as input, such that $C \supset C_0 = A_0 \cup B_0$, and A_0 and B_0 are sets of points in C_0 that belong to the two classes A and B. We refer to C_0 as the *incremental subset*. Let $\Gamma_0 = C \setminus C_0$ be the initial set of points that can be included in the incremental subset. ReGEC classifies all of the points in the training set C using the kernel from C_0. Let P_{A_0} and P_{B_0} be the hyperplanes found by ReGEC, R_0 be the classification accuracy and M_0 be the points that are misclassified. Then, among the points in $\Gamma_0 \cap M_0$ the point that is farthest from its respective hyperplane is selected, i.e.

$$x_1 = x_i : \max_{x \in \{\Gamma_0 \cap M_0\}} \left\{ dist(x, P_{class(x)}) \right\}, \tag{12}$$

where $class(x)$ returns A or B depending on the class of x. This point is the candidate point to be included in the incremental subset. This choice is based on the idea that a point very far from its plane may be needed in the classification subset in order to improve accuracy. We update the incremental set as $C_1 = C_0 \cup \{x_1\}$. Then, we classify the entire training set C using the points in C_1 to build the kernel. Let the classification accuracy be R_1. If $R_1 > R_0$ then we keep the new subset; otherwise we reject the new point, that is $C_1 = C_0$. In both cases $\Gamma_1 = \Gamma_0 \setminus \{x_1\}$. The algorithm repeats until the condition $|\Gamma_k| = 0$ is reached. The algorithm can be summarized as follows:

Algorithm 1 I-ReGEC(C_0, C)

```
$\Gamma_0 = C \setminus C_0$
$\{R_0, M_0\} = Classify(C, C_0)$
$k = 1$
while $|\Gamma_k| > 0$ do
    $x_k = x : \min_{x \in \{M_k \cap \Gamma_{k-1}\}} \left\{ dist(x, P_{class(x)}) \right\}$
    $\{R_k, M_k\} = Classify(C, \{C_{k-1} \cup \{x_k\}\})$
    if $R_k > R_{k-1}$ then
        $C_k = C_{k-1} \cup \{x_k\}$
        $\Gamma_k = \Gamma_{k-1} \setminus \{x_k\}$
        $k = k + 1$
    end if
end while
```

In Figure 1 a graphical example of this approach is shown. The classification surfaces of the two classes (dark and white), generated using 400 training points of the Banana dataset [23], clearly define the aim of our strategy. Indeed, when the ReGEC algorithm is trained on all of the training points the classification boundaries are significantly affected by noisy points (left). On the other hand, I-ReGEC method achieves clearly defined boundaries (right). Furthermore, the number of points needed in the example to generate the classification hyperplane are only 23 in I-ReGEC compared to 400 points in ReGEC.

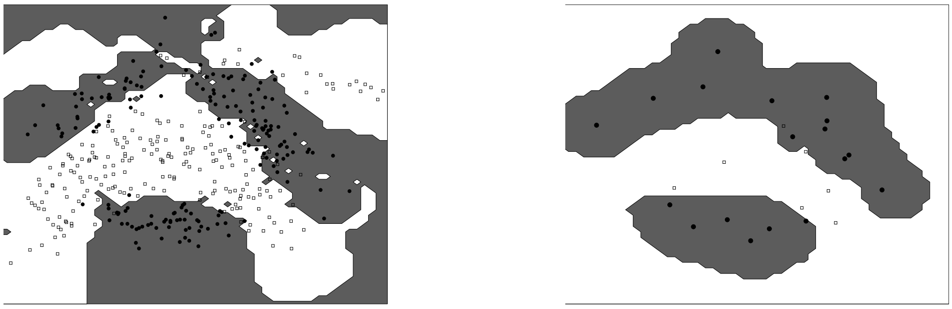

Figure 1: Classification surfaces produced by ReGEC and I-ReGEC on the two dimensional dataset Banana.

4 Initial points selection

In the previous section, we assumed that we have a starting set of points for I-ReGEC. However, we have not mentioned the bias this initial set introduces. Since the initial points permanently become a part of the incremental subset, it is intuitive that such points should be chosen carefully. In this section we show how the initial set of points influence the performance of the incremental selection algorithm. Clustering techniques can be adapted to obtain better data representations [16]. For this purpose, we compare k randomly selected starting points for each class, and a set of points determined by a simple *k-means* method [21], also for each class. We show that it is possible to reach higher classification accuracy and a more consistent representation of the training set using k-means method.

The two datasets used for the comparison have 2 dimensions, in order to show the consistency of the k-means method over random selection, graphically. From each class, k points are chosen for both random and k-means methods. The first dataset is the *Banana* dataset with 400 training points and 4900 test points. The second set of points is the *Chessboard* dataset. It contains 16 squares, with a total of 1000 training and 5400 test points.

First, classification parameters are determined using a ten fold cross validation technique using the training and test points. An initial set of starting points is chosen *a)*randomly, and *b)*using the barycenters of the clusters produced by the k-means method. Each set is used as input to I-ReGEC algorithm, which returns a final incremental subset of points C^*, and the final classification accuracy. Using the same parameters we repeat the procedure of choosing initial points and running I-ReGEC 100 times for both the random and the k-means methods as the generator of the initial sets.

Let C_i^* be the final subset of points produced in the t^{th} repetition. Then, for each kernel produced by C_i, we classify a dense set of evenly distributed points in the rectangle that encloses the entire dataset. Let x be one of such points in the rectangle and $y_i \in \{-1, 1\}$ be the classification result using the kernel based on C_i. Then the value $\hat{y} = |\sum_{i=1}^{100} y_i|/100$ is an estimator of the probability that x is always classified in the same class. We can say that the closer $\hat{y}$ is to 1, the more consistently it is classified. In Figure 2, white color is associated to the points for which $\hat{y} = 1$ and black for $\hat{y} = 0.5$. The lighter regions in Figure 2 are more consistent compared to dark regions, where the points have the same probability to be classified in one of the two classes.

In Figure 2, the influence of the starting points on the resulting classification can be seen clearly. The Banana dataset has few clusters of data and consequently, for a choice of $k = 5$, the average classification accuracy slightly changes between random initial points, which produce a classification accuracy of 84.5%, and k-means initial points, with accuracy of 85.5%.

In order to compare the consistency of the two initial points selection strategies, we measure the standard deviation of the $\hat{y}$ values for the points in the rectangle. The k-means method acieves a standard deviation of 0.01 compared to the standard deviation of 0.05 from the random method, which means that k-means method has a higher classification consistency than random selection.

For the Chessboard dataset, the clusters are clearly separated for each class when $k = 8$. The difference is more pronounced both in terms of classification accuracy and consistency. Random selection of initial points could only reach a classification accuracy of 72.1 %, whereas k-means reaches 97.6 % accuracy. The difference in classification consistency is far more evident compared to the Banana dataset, with a standard deviation of 1.45 for random selection and 0.04 for k-means. We can empirically infer from the results that a knowledge regarding the dataset and the choice of initial points influences both classification accuracy and classification consistency. This influence may be greater as the number of clusters increases.

We also investigated the effect of the number of initial points k for each class using the k-means method on the Chessboard dataset. In Figure 3, the graph on top is the classification accuracy versus the total number of initial points $2k$ from both classes. It reaches its peak at 16 (for $k = 8$), after which it slightly decreases and continues at a steady state of accuracy for higher values of k. This result empirically shows that there is a minimum k, with which we reach high accuracy results. Although the decrease in the accuracy is not significant for larger values of k, the kernel to be used in I-ReGEC unnecessarily increases. This is shown by the bottom graph in Figure 3 which shows the number of points selelcted by I-ReGEC versus the nuber of initial points. Again, no additional points are added to the initial 16 (for $k = 8$), and the number of points added are almost the same beyond. This

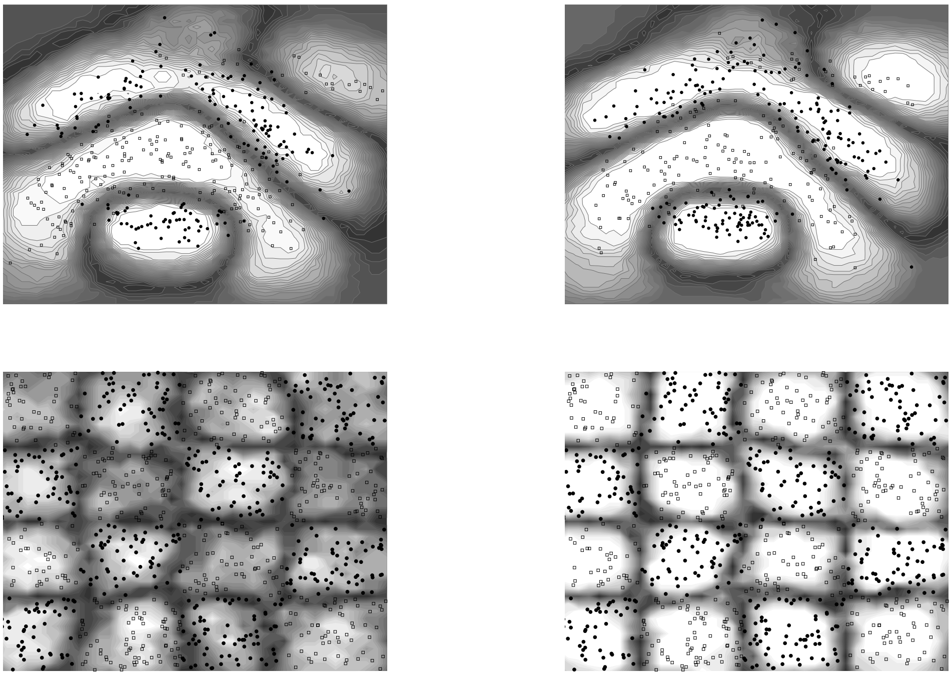

Figure 2: Classification consistency of I-ReGEC: Light regions show higher consistency than darker regions. Top row shows the results from *Banana* dataset (k = 5), and bottom row from *Chessboard* dataset (k = 8). Figures on the left are produced using a random selection of initial points, and figures on the right using k-means method.

means that the initial set of points reaches a minimum at an ideal number of k and it grows linearly with k. One simple and practical way of finding a good k is to increase k incrementally and detecting the lowest value of k with higher classification accuracy.

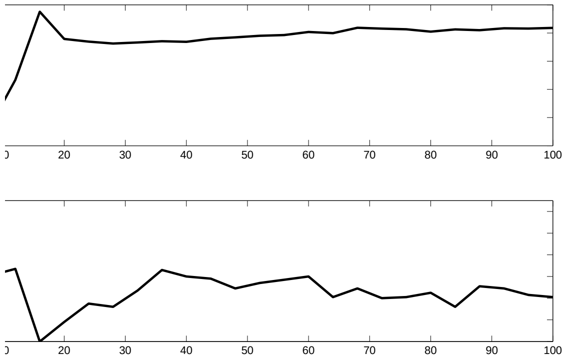

Figure 3: Performance of I-ReGEC for with respect to k: Top figure shows the k vs. classification accuracy; bottom figure shows k vs. the number of additional points included on top of the initial points.

5 Numerical results

I-ReGEC is tested on publicly available benchmark data sets. Results regarding its performance in terms of classification accuracy are presented. We used data from IDA [23] and from UCI [24] repositories, both of which are widely used to compare the performance of new algorithms to existing methods. The accuracy results for the nonlinear kernel are taken from [23]. Accuracy results are calculated using an Intel Xeon CPU 3.20GHz, 6GB RAM running Red Hat Enterprise Linux WS release 3 with Matlab 6.5. Matlab function *eig* for the solution of the generalized eigenvalue problem is used for ReGEC.

In Table 1, for each data set, name, dimension of the training and test sets, and the number of features are reported. In Table 2, classification accuracy is evaluated using Gaussian kernel for ReGEC, I-ReGEC, and SVM,

Dataset	**train**	**test**	**m**
Banana	400	4900	2
German	700	300	20
Diabetis	468	300	8
Haberman	275	31	4
Bupa	310	35	6
Votes	391	44	16
WPBC	99	11	32
Thyroid	140	75	5
Flare-solar	666	400	9

Table 1: Datasets characteristics

using ten-fold cross-validation to determine parameters. A Gaussian kernel is used for each classifier and the value of the best kernel parameter σ together with the k value for the k-means method for I-ReGEC are also included in the table. The k value for each dataset is empirically determined as follows: first, the best σ value is determined for $k = 2$ using ten-fold cross-validation; then, the best k value is determined by gradually increasing its value.

I-ReGEC is nearly always more accurate than ReGEC. The slight difference in accuracy for the two datasets where ReGEC gives better results could be due to the cross validation procedure. We have also compared the accuracy results of I-ReGEC with SVM. Results are always slightly lower than SVM, except for one data set. The relative difference of accuracy, i.e., the absolute difference of the accuracies of I-ReGEC and SVM, divided by the maximum value, is less then 8.2%, except the case of Flare-solar (11.50%) and Bupa dataset (15.55%).

Dataset		**ReGEC**		**I-ReGEC**				**SVMs**
	train	σ	acc	chunk	k	σ	acc	acc
Banana	400	0.2	84.44	15.7	5	0.2	85.49	89.15
German	700	500	70.26	29.09	8	10	73.5	75.66
Diabetis	468	500	74.56	16.63	5	400	74.13	76.21
Haberman	275	1200	73.26	7.59	2	20000	73.45	71.7
Bupa	310	200	59.03	15.28	4	800	63.94	69.9
Votes	391	50	95.09	25.9	10	100	93.41	95.6
WPBC	99	1000	58.36	4.2	2	50	60.27	63.6
Thyroid	140	0.8	92.76	12.40	5	1.5	94.01	95.2
Flare-solar	666	3	58.23	9.67	3	3	65.11	65.8

Table 2: Classification accuracy for ReGEC, I-ReGEC and SVM algorithms using gaussian kernel

In Table 3 the dimension of incremental datasets and the percentage with respect to the dimension of the training set is given. In all cases, I-ReGEC produced a subset composed of less then 8.85% of the training set with a comparable classification accuracy on the test sets with respect to the original ReGEC method.

Dataset	**I-ReGEC**	
	chunk	% of train
Banana	15.7	3.93
German	29.09	4.16
Diabetis	16.63	3.55
Haberman	7.59	2.76
Bupa	15.28	4.93
Votes	25.9	6.62
WPBC	4.2	4.3
Thyroid	12.40	8.86
Flare-solar	9.67	1.45

Table 3: Incremental dataset using I-ReGEC and percentage of training set

6 Conclusions and future work

In this study, we describe I-ReGEC, a novel incremental classification technique, with dramatic results in reducing the cardinality of training sets, when applied to general eigenvalue classifiers. The proposed method achieves a high classification consistency and classification accuracy comparable with other SVM methods. Furthermore, it allows efficient online updating of the classification function when new training points become available. I-ReGEC method can be improved by adaptive techniques for the selection of the initial points, in order to find better strategies to build the incremental subset. Furthermore, new criteria for including new points to the incremental subset or removing less promising points from the incremental subset may be considered.

References

[1] J. Abello, P.M. Pardalos, and M.G.C. Resende, editors. *Handbook of massive data sets*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[2] K. Bennet and C. Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2(2):1–13, 2000.

[3] K. Bennett and O. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[4] M. Brown, W. Grundy, W. Lin, D. Cristianini, N. Sugne, C. Furey, T. Ares, and D. Haussler. Knowledge-base analysis of microarray gene expressiondata by using support vector machines. *PNAS*, 97(1):262–267, 2000.

[5] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415, 2000.

[6] C. Cifarelli and G. Patrizi. Solving large protein folding problem by a linear complementarity algorithmwith 0-1 variables. *Optimization Methods and Softwares*, 2006. Accepted.

[7] F. Cucker and S. Smale. On the mathematical foundation of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2001.

[8] C. Domeniconi and D. Gunopulos. Incremental support vector machine construction. In *First IEEE International Conference on Data Mining (ICDM'01)*, pages 589–593, 2001.

[9] G.N. Garcia, T. Ebrahimi, and J.M. Vesin. Joint time-frequency-space classification of eeg in a brain-computer interface application. *Journal on Applied Signal Processing*, pages 713–729, 2003.

[10] F. Giannessi. Complementarity problems and their applications to structural engineering. In Pitagora, editor, *Methods and algorithms for optimization*, pages 507–514, Bologna, 1982.

[11] M. R. Guarracino, C. Cifarelli, O. Seref, and P.M. Pardalos. A classification algorithm based on generalized eigenvalue problems. *Optimization Methods and Software*, 2006. Accepted.

[12] C.W. Hsu, C.C. Chang, and C.J. Lin. A practical guide to support vector classification. http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf, 2004.

[13] Z. Huang, H. Chen, C. J. Hsu, W.H. Chenb, and S. Wuc. Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decision Support Systems*, 37:543–558, 2004.

[14] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nellec and Cine Rouveirol, editors, *Proceedings of the European Conference on Machine Learning*, pages 137–142, Berlin, 1998. Springer.

[15] T. Joachims. *Making large-Scale SVM Learning Practical.* Advances in Kernel Methods - Support Vector Learning. MIT-Press, 1999.

[16] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, New York, 1990.

[17] T. N. Lal, M. Schroeder, T. Hinterberger, J. Weston, M. Bogdan, N. Birbaumer, and B. Scholkopf. Support vector channel selection in bci. *IEEE Transactions on Biomedical Engineering*, 51(6):1003–1010, 2004.

[18] S. Lee and A. Verri. Pattern recognition with support vector machines. In *SVM 2002*, Niagara Falls, Canada, 2002. Springer.

[19] Y.J. Lee and O.L. Mangasarian. Rsvm: Reduced support vector machines. In *First SIAM International Conference on Data Mining*, 2001.

[20] K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 6(14):1449 – 1459, 2003.

[21] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of Berkeley Symposium on Math Stat Probability*, 1965.

[22] O. L. Mangasarian and E. W. Wild. Multisurface proximal support vector classification via generalized eigenvalues. Technical Report 04-03, Data Mining Institute, September 2004.

[23] S. Mika, G. Ratsch, J.Weston, B. Scholkopf, and K.R. Muller. Fisher discriminant analysis with kernels. *IEEE Neural Networks for Signal Processing*, IX:41–48, 1999.

[24] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[25] W.S. Noble. *Kernel Methods in Computational Biology*, chapter Support vector machine applications in computational biology, pages 71–92. MIT Press, 2004.

[26] R.F.E. Osuna and F. Girosi. An improved training algorithm for support vector machines. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, 1997.

[27] J. Platt. *Advances in Kernel Methods: Support Vector Learning*, chapter Fast training of SVMs using sequential minimal optimization, pages 185–208. MIT press, Cambridge, MA, 1999.

[28] L. Ralaivola. Incremental support vector machine learning: A local approach. *Lecture Notes in Computer Science*, 2130:322–330, 2001.

[29] B. Scholkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, 2001.

[30] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis.* Cambridge University Press, Cambridge, UK, 2004.

[31] T.B. Trafalis and H. Ince. Support vector machine for regression and applications to financial forecasting. In *International Joint Conference on Neural Networks (IJCNN'02)*, Como, Italy, 2002. IEEE-INNS-ENNS.

[32] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.