Training and Analysis of Support Vector Machine using Sequential Minimal Optimization

S.Shahbudin[1], A. Hussain[2], S. A. Samad[3]
Electrical, Electronics & System Engineering Department
Faculty of Engineering and Built Environment,
National University of Malaysia
Bangi, Selangor Darul Ehsan, Malaysia
shaqay@yahoo.com[1] , aini@vlsi.eng.ukm.my[2]

N. Md Tahir[4]
Faculty of Electrical Engineering
Technology University of Mara,
Shah Alam, Selangor Darul Ehsan, Malaysia

***Abstract*— Maximizing the classification performance of the training data is a typical procedure in training a classifier. It is well known that training a Support Vector Machine (SVM) requires the solution of an enormous quadratic programming (QP) optimization problem. Serious challenges appeared in the training dilemma due to immense training and this could be solved using Sequential Minimal Optimization (SMO). This paper investigates the performance of SMO solver in term of CPU time, number of support vector and decision boundaries when applied in a 2-dimensional datasets. Next, the chunking algorithm is employed for comparison purpose. Initial results demonstrated that the SMO algorithm could enhance the performance of the training dataset. Both algorithms illustrated similar patterns from the decision boundaries attained. Classification rate achieved by both solvers are superb.**

***Keywords*— Sequential Minimal Optimization, Chunking algorithm, decision boundaries, support vector machine.**

I. INTRODUCTION

Support Vector Machine (SVM) is an eminent technique for solving classification problems. The goal of SVM is to determine a classifier or regression machine that minimizes the empirical risk namely the training set error and the confidence interval which corresponds to the generalization or test set error [1]-[2].

To obtain a SVM classifier with the best generalization performance, appropriate training is required. Training SVM entails the solution of a very large quadratic programming (QP) optimization problem. However, Sequential Minimal Optimization (SMO) with a fixed working set size is amongst the popular decomposition method for training even for very large data sets. Most of the researches [3]-[7], proved that SMO gave superb performances in training infinite N-dimensional data size. For example, in [3], SMO is applied to train SVM for classifying large dataset such as the (UCI) "adult" data set, text categorizations and sparse dataset. Results indicated that SMO algorithm provides a better scaling for both linear and non linear SVM with RBF Gaussian as the kernel function. It is also performed extremely well with sparse data sets even for non-linear SVM. In [7], various sample datasets such as ionosphere, breast cancer and adult dataset have been tested and result proved that SMO-based algorithm is significantly more efficient than other methods available in the optimization toolboxes.

However, in training of 2-dimensional (2D) data, the performance of SMO algorithm is rarely visualized, analyzed and studied. Thus, the purpose of this paper is to explore the SMO capability in training 2D data as compared to chunking algorithm along with visualization of the decision boundary figures of both algorithms. In this study, to analyze the performance of SMO algorithm, the parameters like CPU time, number of support vector and the shape of decision boundaries will be examined and observed.

A description of SVM is detailed in Section 2. In Section 3, several previous training SVM algorithms are explained. Experimental results of both algorithms are presented in Section 4. Finally, in Section 5 we conclude our findings.

II. OVERVIEW OF SVM

In general, a Support Vector Machine (SVM) is a learning machine for two class classification problems, and given a labeled training dataset, $(x_1,y_1),\ldots,(x_l,y_l)$ where $x_i \in R^N$ is a feature vector and $y_i \in \{-1,1\}$ is a class label.

The algorithm seeks to define a decision surface which gives the largest margin or separating between the data classes whilst at the same time minimizing the number of errors. However, this decision surface is not created in the input space, but rather in a very high-dimensional feature space. The resulting model is nonlinear, and is accomplished by the use of kernel functions. The kernel function, *K* indicates a measure of similarity between a pattern x_i, and a pattern x_j from the stored training set. Using the kernel, the dual QP problem in term of Lagrange Multipliers, α_i in the feature space is given in equation (1), that is maximize

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \qquad (1)$$

subject to the constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \quad 0 \le \alpha_i \le C \qquad (2)$$

where i=1,…,l.

After finding the optimal values of α_i, the decision boundary that needs to be constructed is of the form

$$f(x) = \sum_{\alpha_i \neq 0}^{l} y_i \alpha_i K(x_i, x) + b. \quad (3)$$

where the class of x is determined from the sign of $f(x)$. Those x_i corresponding $\alpha_i \neq 0$ is called support vector. The value b is a threshold of the decision boundary from origin. The regularization parameter, C, is the margin parameter that determines the trade-off between maximizing the margin and minimizing the classification error and is chosen by means of a validation set [7].

In SVM classification one of the attracting features is the sparsity representation of its decision boundary. According to [8], the position of the separating hyperplane in the feature space is determined via real-valued weights on the training set examples. Those training examples that are situated far away from the hyper plane do not participate in its specification and thus receive weights of zero. Only the training examples that lie close to the decision boundary between the two classes receive nonzero weights. These training examples are called the support vectors, since removing them would change the location of the separating hyper plane. As an example, Fig. 1 illustrates the support vectors in a two-dimensional feature space. Typically, SVM learning algorithm is defined such that the number of support vectors is less compared to the total number of training examples, thus allowing the SVM to classify new examples efficiently, since the majority of the training examples can be safely ignored.

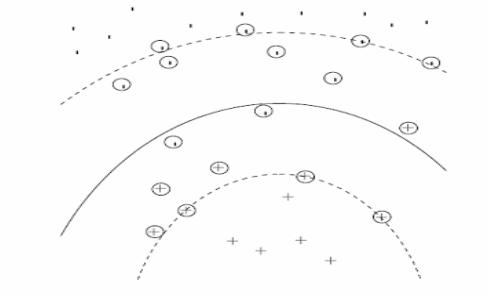


Figure 1. A 2D feature space with a separating hyperplane for non-linear boundary. Both classification boundary and the accompanying soft margins are represented by solid line and dotted lines, respectively where as positive and negative examples fall on opposite sides of the decision boundary. The circled points are the support vectors that lie closest to the decision boundary.

III. PREVIOUS SVM TRAINING METHODS

The first training of a SVM with small data sets was introduced by Vapnik [9], using constrained conjugate gradient algorithm. Conjugate gradient ascent started with an initial estimate for solution, denoted by α_o, and then updates the vector iteratively following the steepest ascent path, that is moving in the direction of the gradient of W(α) evaluated at the position α^t for update t+1.At each iteration, the direction of update is determined by the steepest ascent strategy, but at the same time the step length is kept fixed. In this method, every time α_i reaches zero, the corresponding data point is eliminated and the process will be re-started.

As such, the decomposition or working set method that is the SMO which is known to be an excellent method to train large data set problems will be investigated for its capability in dealing with 2D data. In this study, SMO training capability is explored and compared to another type of training method called the chunking algorithm.

A. Chunking

The Chunking algorithm was proposed by Vapnik [10] in which started with arbitrary subset or 'chunk' the data and train an SVM, Then, the support vectors remain in the chunk while other points are discarded and replaced by a new working set with gross violations of KKT (Karush-Kuhn-Tucker) conditions. In the final iteration the entire set of non-zero Lagrange multipliers is extracted and hence the algorithm solves the QP problem.

B. Sequential Minimal Optimization (SMO)

Recently, SMO has been employed to rapidly train SVM. The idea behind SMO is that the QP problems can be broken up into a series of the smallest possible QP problems and solved analytically by optimizing two α_i at each iteration and keeping the remaining α_i as fixed. These two values can be acquired easily and rapidly and thus helps avoid large matrix computation .Details of SMO can be found in [3]-[4].

The main difference between the SMO method and the chunking algorithm is that SMO solves the QP problem analytically without any extra matrix storage whereas for chunking algorithm, the QP problem needs to be solved iteratively; which involves exhaustive numerical QP steps and thus required exponential memory [3].

IV. EXPERIMENTAL RESULTS

SMO solver is implemented to train the binary SVM classifier with L1-soft margin. Based on [12], for each solver, the convergence of tolerance, ε = 0.001, the value of kernel argument is equal to 1 and the Gaussian radial basis function (RBF) kernel was opted that is $K(x_i,x_j)=\exp(-\|x_i-x_j\|^2/(2\sigma^2))$.

The CPU time of both algorithms were measured on a Pentium R, 3.0 GHz computer with 768MB RAM. Both SMO and Chunking solvers were implemented using the Statistical Pattern Recognition Toolbox [12] and Matlab 7.0 respectively.

Over 190 2-D data acquired from [11] are divided equally for training and testing. These values represent the feature vectors of the second and forth eigenpostures that are generated based on PCA technique. Both the SMO and chunking solvers are used. The system is trained on the training data and its performance measured on the test data. The trained SVM classifier is evaluated on the 2D training data using various values of regularization parameter, C. The example of decision boundary with C=10 attained is as shown in Fig. 2.

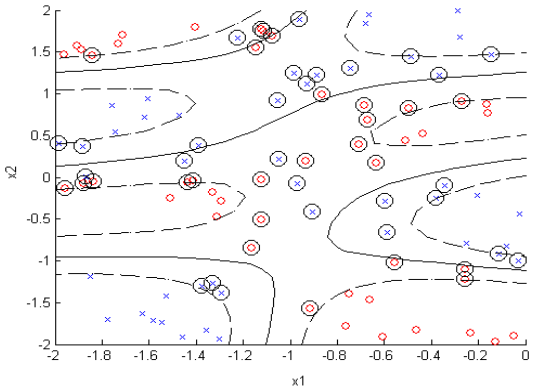


Figure 2. Sample of 2D eigenpostures dataset using SMO solver (C=10)

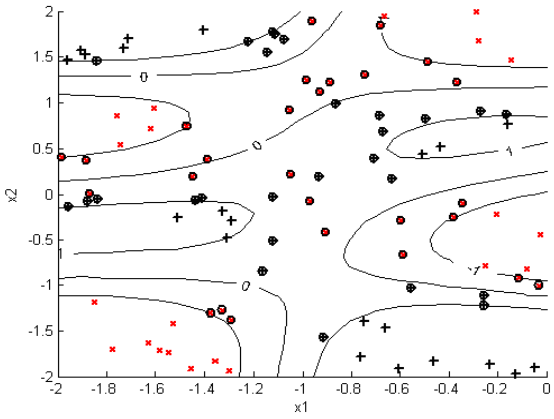


Figure 3. Sample of 2-D eigenpostures dataset using Chunking solver (C=10)

From both figures, it is observed that the patterns of decision boundary are alike. However, there exist differences in the CPU time as tabulated in Table I. The CPU time of the Chunking algorithms is 1.6853 seconds whilst the SMO algorithm recorded the CPU time of only 0.1288 seconds. The number of support vector for both solvers are almost equal that is 54 for chunking solver and 52 for SMO solver. The same goes for the classification rates with both solvers gained an accuracy of 90%.

Further, results for decision boundaries of the SMO and chunking solvers with regularization parameter C of value 100 are depicted in Fig. 4 and Fig. 5 respectively. It is again observed that both decision boundaries are almost similar with equal number of support vectors of 30. Again, the classification rates of both solvers are similar specifically 93.75% for SMO and 93.47% for the chunking solver. As before, the CPU time for SMO is faster than the chunking solver. More results with various value of C are tabulated in Table I.

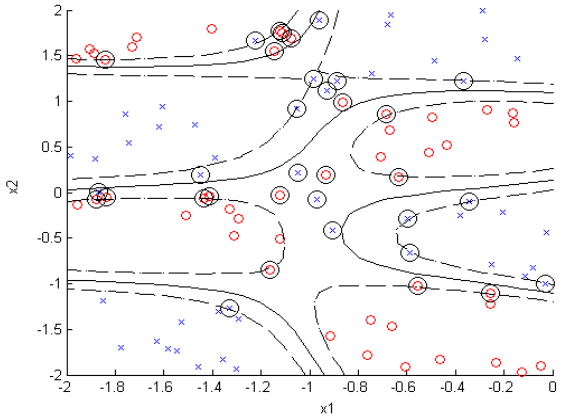


Figure 4. Sample of 2D eigenpostures dataset using SMO solver (C=100)

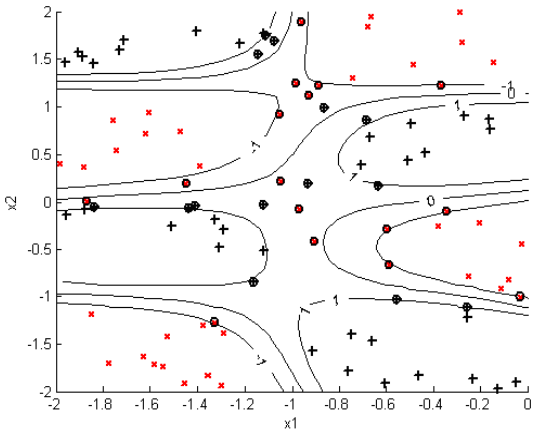


Figure 5. Sample of 2-D eigenpostures dataset using Chunking solver (C=100)

TABLE I. PERFORMANCE COMPARISON OF SMO AND CHUNKING SOLVER WITH FROM C=10 TO C=500 (EIGENPOSTURES DATASET)

Training Algorithms	**Parameter Measured**			
	Regularization Parameter, C	***Number of support vectors***	***CPU Times (s)***	***Classification Accuracy (%)***
SMO solver	10	52	0.1288	91.67
	50	38	0.2591	92.78
	100	30	0.4097	93.75
	500	24	1.6363	93.75
Chunking solver	10	54	1.6853	91.51
	50	37	1.3853	92.65
	100	30	1.7758	93.47
	500	25	2.4234	96.76

From the results, the decision boundaries for both solvers depict similar patterns but recorded different CPU time. The CPU time of SMO solver is shorter than the chunking solver irrespective of the C values. It is also observed that the regularization parameter, C is inversely related to the number of support vectors, that is, as the value increases, the number of support vectors will decrease. Additionally, even though the CPU time of both solvers is different the solvers performances in terms of the classification error rates are unaffected.

Next, the Iris and Ripley datasets acquired from the software [12] and UCI Machine Learning Repository Databases respectively are utilized to verify these findings. Both datasets are in the 2D form. The Ripley dataset [12]-[13] comprised of 250 training data and 1000 patterns as the testing data. The Iris dataset consists of 120 patterns with 60 patterns are used as the training data and the remainder as testing data. The trained SVM classifier is evaluated on both datasets using various values of regularization parameter, from C =10 to C=500. Results are as summarized in Table II and Table III respectively.

TABLE II. PERFORMANCE COMPARISON OF SMO AND CHUNKING SOLVER FROM C=10 TO C=500 (IRIS DATASET)

Training Algorithms	**Parameter Measured**			
	Regularization Parameter C	***Number of support vectors***	***CPU Times (s)***	***Classification Accuracy (%)***
SMO solver	10	12	0.0556	95.00
	50	10	0.0563	95.00
	100	9	0.0420	98.33
	500	8	0.0161	98.33
Chunking solver	10	12	0.3042	95.00
	50	9	0.7916	96.67
	100	9	0.4533	98.33
	500	8	0.2266	98.33

TABLE III. PERFORMANCE COMPARISON OF SMO AND CHUNKING SOLVER WITH FROM C=10 TO C=500 (RIPLEY DATASET)

Training Algorithms	**Parameter Measured**			
	Regularization Parameter C	***Number of support vectors***	***CPU Times (s)***	***Classification Accuracy (%)***
SMO solver	10	94	0.3469	85.60
	50	85	0.4931	87.60
	100	83	0.5743	87.60
	500	77	2.9142	89.20
Chunking solver	10	94	0.7706	82.60
	50	85	0.9766	84.35
	100	83	1.4559	84.40
	500	76	3.0644	89.45

From Table II and III, it is observed for both datasets that the SMO performed faster than the chunking solver from the CPU time attained. In addition, the number of support vectors obtained is equivalent for both solvers towards Ripley and Iris datasets. Also, the number of support vectors will reduce when C increased. For both solvers, the Iris and the Ripley dataset obtained excellent classification rate for various value of C. It is also observed that both solvers generated similar decision boundaries as depicted in Fig. 6(a)-(d) and Fig. 7(a)-(d) based on the different values of C applied. Fig. 6 depicts the results using Iris dataset whereas Fig. 7 depicts the results using Ripley dataset.

V. CONCLUSIONS

As a conclusion, the SMO algorithm realized better training method for SVM 2D data size based on the CPU time attained. From the visualized decision boundaries, both solvers illustrated similar pattern for both value of C applied along with similar and excellent classification rate. Furthermore, the number of support vectors is equal for both solvers with different values of C. Initial results demonstrated that the SMO is an efficient approach for training the SVM even for 2D data samples.

REFERENCES

[1] V.N. Vapnik, "The nature of statistical learning theory," Springer, New York, 1995.

[2] N.Cristianini, J.Shawe-Taylor,"An Introduction to Support Vector Machines:and other kernel-based learning methods,"New York: Cambridge University Press, 2000.

[3] J. Platt, "Fast training of support vector machines using sequential minimal optimization:' in Advances in Kemrl Marhods -Support Vector Laming, B. Schdlkopi, C.1.C Burges. and A.J. Smala. editors, pages 185-208. MIT Press, Cambridge. MA, 1999.

[4] Ginny Mak , "The Implementation Of Support vector machines using the sequential minimal optimization algorithm," Master thesis, 2000.

[5] E. Osuna, R. Freund and F. Girosi, "Support vector machines: Training and Applications," A.I. Memo AIM-1602, MIT A.I. Lab, 1996.

[6] Francis R. Bach & Gert R. G. Lanckriet, Michael I. Jordan , "Fast kernel learning using sequential minimal optimization," Report No. UCB/CSD-04-1307 February, 2004.

[7] Shiego Abe, "Support vector machines for pattern classification" Advances in Pattern Recognition , Springer 2005.

[8] Michael P. S. Brown,William Noble Grundy,David Lin,Nello Cristianini,Charles Sugnet,Manuel Ares,Jr. David Haussler , "Support Vector Machine Classification of Microarray Gene Expression Data," Technical Report No: UCSC-CRL-99-09, June 12, 1999.

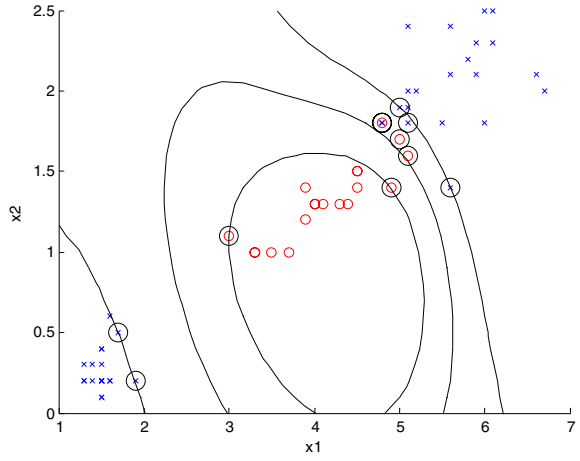
[9] C.Burges and V.Vapnik, "A new method for constructing artifial neural networks,"Technical report, AT&T Bell Laboratories,May 1995

[10] V.Vapnik, "Estimation of Dependences Based on Empirical Data," Springer-Verlag, 1982.

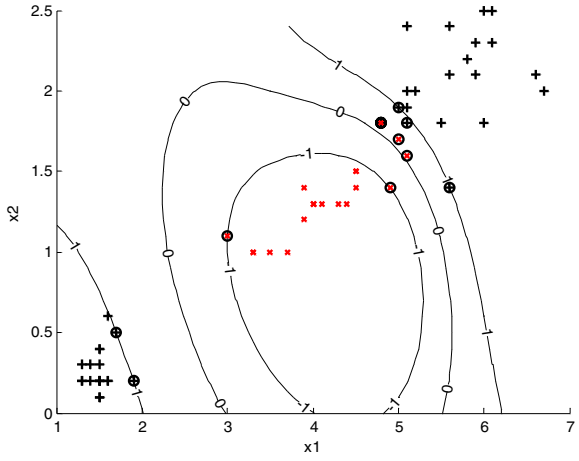
[11] Nooritawati Md Tahir, Aini Hussain, Salina Abdul Samad, Hafizah Husain & Mohd Marzuki Mustafa, "Eigenposture For Classification" Journal of Applied Sciences, Asian Network for Scientific Information, ANSINET, 6(2), 2006.

[12] Statistical Pattern Recognition Toolbox http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html

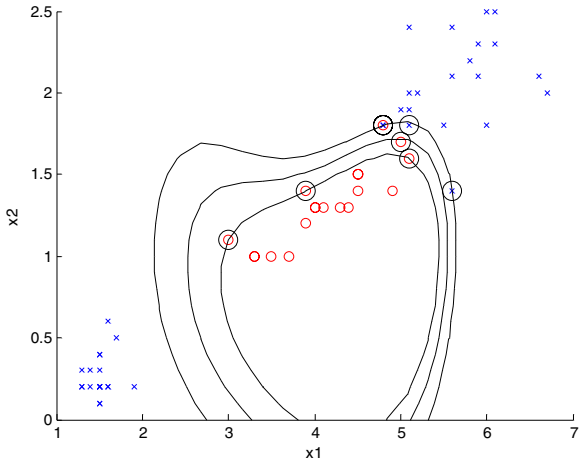
[13] B.D. Ripley, "Neural networks and related methods for classification," J. Royal Statistical Soc. Series B, 56:pp.409–456, 1994.



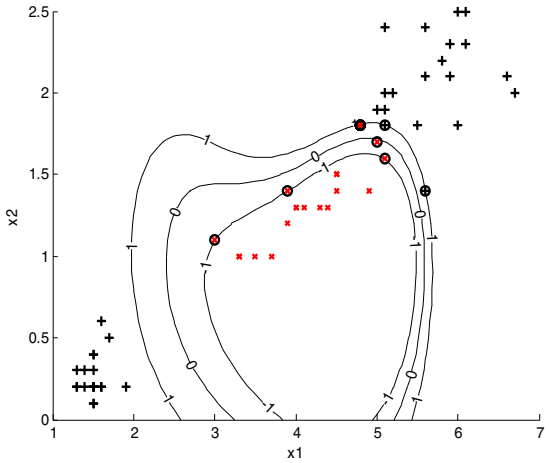
(a) When C=10 for SMO solver



(b) When C=10 for chunking solver

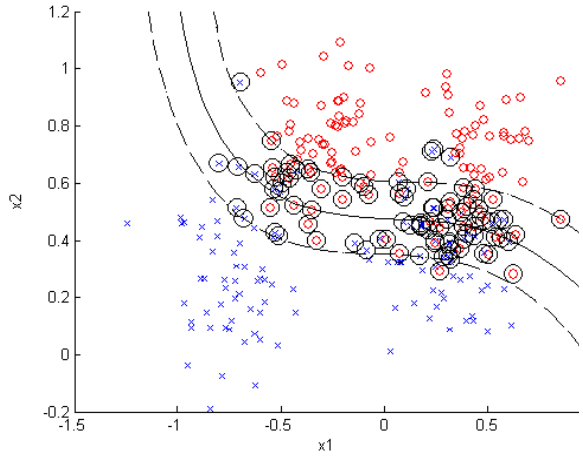


(c) When C=100 for SMO solver

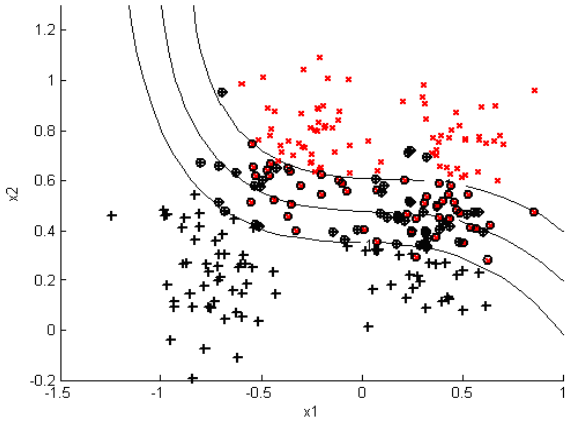


(d) When C=100 for chunking solver

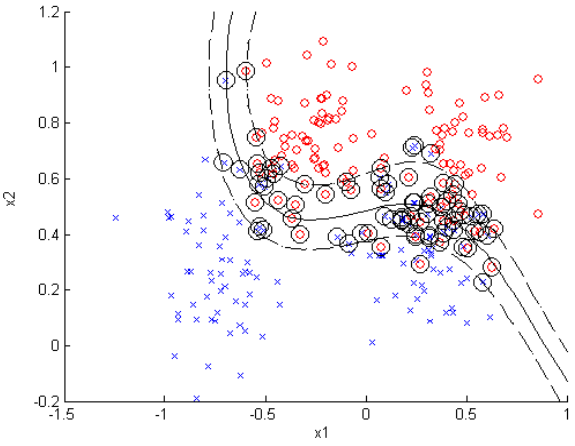
Figure 6. Results obtained using iris dataset



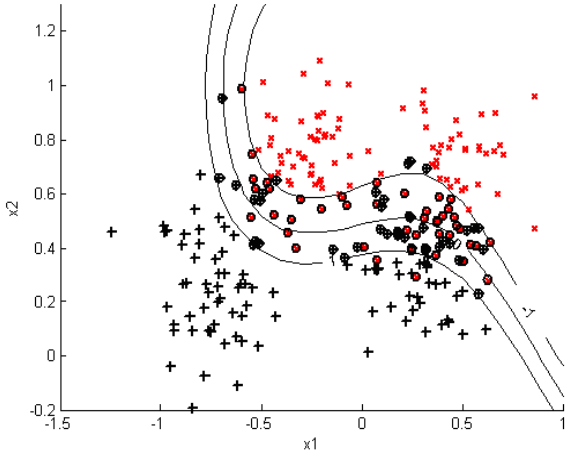
(a) When C=10 for SMO solver



(b) When C=10 for chunking solver



(c) When C=100 for SMO solver



(d) When C=100 for chunking solver

Figure 7. Results obtained using Ripley dataset