Jurnal Teknologi, 39(D) Dis. 2003: 45–56

A COMPARISON OF QUADRATIC PROGRAMMING SOLVERS IN SUPPORT VECTOR MACHINES TRAINING

N. M. ZAKI[1], S. DERIS[2] & K. K. CHIN[3]

Abstract. Training a Support Vector Machine requires the solution of a very large quadratic programming problem. In order to study the influence of a particular quadratic programming solver on the Support Vector Machine, three different quadratic programming solvers are used to perform the Support Vector Machine training. The performance of these solvers in term of execution time and quality of the solutions are analyzed and compared. A practical method to reduce the training time is investigated.

Keywords: Support vector machines, quadratic programming

Abstrak. Penyelesaian atur cara kuadratik yang sangat besar diperlukan untuk melatih *Support Vector Machine.* Tiga cara penyelesaian atur cara kuadratik yang berbeza telah digunakan untuk melaksanakan latihan *Support Vector Machine* bagi mengkaji keberkesanannya ke atas *Support Vector Machine.* Prestasi bagi kesemua penyelesaian telah dikaji dan dianalisis dari segi masa pelaksanaan dan kualiti penyelesaian. Kaedah praktikal untuk mengurangkan masa latihan tersebut telah dikaji sepenuhnya.

Kata kunci: Support vector machines, atur cara kuadratik

1.0 INTRODUCTION

In the last few years, there has been a surge of interest in Support Vector Machines (SVMs) [1]. SVMs have empirically been shown to give good generalization performance on a wide variety of problems. However, the use of SVMs is still limited to a small group of researchers. One possible reason is that training algorithms for SVMs are slow, especially for large problems. Another explanation is that SVM training algorithms are complex, subtle, and sometimes difficult to implement [2]. Training a Support Vector Machines require the solution of a very large quadratic programming (QP) problem. Solving the QP problem in SVM training appears to be simple and straightforward: any optimization packages that solve linearly constrained convex quadratic problems can be used. The SVM community has used various optimization

1 Department of Software Engineering, Faculty of Computer Science & Information System, University Technology Malaysia. E-mail: nazar@siswa.utm.my
2 Department of Software Engineering, Faculty of Computer Science & Information System, University Technology Malaysia.
3 Engineering Department, Cambridge University, Trumpington Street, Cambridge, UK.

tools to train the SVM. Vapnik [3] used a simple constrained conjugate gradient algorithm and an algorithm for bounded large-scale QP. In [4] the performances (in terms of training time) of four different optimizers are compared, and it is concluded that MINOS 5.4 gives the best performance. Burges from AT&T [1] has even developed a QP solver specifically for training SVMs. It is well known that finite numerical precision can cause QP solvers to give non-optimum solutions. But most of the literature in SVM does not discuss the potential difficulties in solving the QP problem due to limited numerical precision except in [1]. This gives an impression that apart from different length of time required obtaining the solution, it does not matter which optimizer is used. In this paper three different quadratic programming solvers are used to perform the Support Vector Machines training. The performance of these solvers in term of execution time and quality of the solutions are analyzed and compared. A practical method to reduce the training time is investigated.

2.0 OVERVIEW OF SUPPORT VECTOR MACHINES

Support Vector Machines (SVMs) are a class of supervised learning algorithms first introduced by Vapnik [5]. Given a set of labeled training vectors (positive and negative input examples), SVMs learn a linear decision boundary to discriminate between the two classes. The result is a linear classification rule that can be used to classify new test examples. SVMs have exhibited excellent generalization performance (accuracy on test sets) in practice and have strong theoretical motivation in statistical learning theory [5].

Suppose our training set S consists of labeled input vectors, (x_i, y_i), i = 1..m where $x_i \in \Re^n$ and $y_i \in \{\pm 1\}$. We can specify a linear classification rule f by a pair $\langle w,b\rangle$, where the normal vector $w \in \Re^n$ and the bias $b \in \Re$ via

$$f(x) = \langle w,b\rangle + b \tag{1}$$

where a point x is classified as positive if $f(x) > 0$. Geometrically, the decision boundary is the hyperplane

$$\{x \in \Re^n : \langle w,b\rangle + b = 0\} \tag{2}$$

In practice, training sets are usually not linearly separable, and we must modify the SVM optimization problem to incorporate a trade-off between maximizing geometric margin and minimizing some measure of classification error on the training set [6]. The quality which upper bounds the generalization error does not depend on the dimensional of the input space and this is the reason why SVMs can use high dimensional spaces without over-fitting. The idea makes it possible to efficiently deal with vary high dimensional futures spaces is the use of kernels:

$$K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle \qquad \text{for all } x, z \in X \tag{3}$$

where ϕ is the mapping from X to an inner product feature space.

Finding the large margin hyperplane, is performed in SVMs by transforming the problem into a QP one, subject to linear constraints. Kuhan-Tucker (KT) theory [3] provides the framework under which the problem can be solved and gives the properties of the solution.

In the data-dependent representation, the Lagrangian

$$L = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{m} \lambda_i \lambda_j y_i y_j K(x_i, x_j) \tag{4}$$

has to be maximized with respect to the λ_i, subject to the constraints

$$\lambda_i \leq 0 \qquad \sum_{i=1}^{m} \lambda_i y_i = 0 \tag{5}$$

There is Lagrangian multiplier λ_i for each training point. Only the points which lie closest to the hyperplane, have $\lambda_i > 0$ and are called support vectors. All the others have $\lambda_i = 0$. The resulting decision function i (1) can be written as:

$$f(x) = sign\left(\sum_{i \in SV} y_i \lambda_i K(x, x_i) - b \right) \tag{6}$$

where is the solution of the constrained maximization problem and SV represents the support vectors.

3.0 NUMERICAL EXPERIMENTS

In order to study the influence of a particular QP solver on the SVM, three different QP solvers are used to perform the SVM training. These optimizers are easily available and have been extensively tested by the optimization community. The performance of these solvers in term of execution time and quality of the solutions are compared. The selected solvers are as follow:

- DONLP2 (Do Non-Linear Programming Ver. 2). This solver implements a new SQP method [7] based on iteratively solving equality constrained sub-problems. The working set is defined as the set of active inequality constraints which are treated as equality constraints in the sub-problems. Most SQP algorithms differ in their strategies in selecting the working set at each iteration. DONLP2 allows any number of changes in the working set in every iteration. It is also supplemented by a regularization technique [8] that deals with the inconsistency of the sub-problems.

- LOQO (Interior Point Methods [9]). This solver starts from a strict interior point (a point inside the feasible region) and moves from this point iteratively towards the solution. At each iteration, the solver estimates a point on the central path (A path in the primal-dual space that simultaneously satisfies the conditions of primal feasibility and dual feasibility) and tries to move towards it while staying in the feasible region.

- QP (MATLAB 5.1). This is an active set method also known as the projection method, described in [10]. A linear programming problem is solved to determine an initial feasible point. An iterative process is then started whereby at each iteration, a step is taken from the current position toward the optimal solution. Each of these steps is calculated to minimize the cost function while remaining within the active constraint boundaries. The active set of constraints is updated at every iteration.

3.1 Data Used

Gordon Peterson and Harold Barney [11], describe a detailed investigation of sustained American English vowels. The experiment presents acoustic measurements of fundamental frequency (F0) and first three formant frequencies (F1-F3). They also conducted experiments where listeners were asked to identify words.

A list of ten words was presented to 76 speakers, each word beginning with [h] and ending with [d], and differing only in the vowel. Each speaker was asked to pronounce two different lists, each list corresponding to a random permutation of the 10 words. Therefore, the total number of recorded words was 1520. The first formant F1 can be related to how far the tongue is raised and F2 to which part of the tongue is raised. Therefore, vowels can be organized according to the tongue's position in plots.

3.2 Comparisons of Training Results from the Three Different Solvers

The three solvers mentioned above are used to train the SVM that classifies the Peterson & Barney speech database, a 10-class problem with feature vectors of dimensions. The training durations for three different sets of data are shown in Table 1. The total Elapsed time is used here because the code that calls the Matlab 5.1 engine does not register

Table 1 Training duration for different solver on 3 different data sets

	LOQO	DONLP2	MATLAB QP
Data Set 1 (80 data points)	1.5 sec	17 sec	3.3 sec
Data Set 2 (160 data points)	8 sec	240 sec	251 sec
Data Set 3 (320 data points)	49 sec	4200 sec	3412 sec

the proper user time usage account. Care has been taken to ensure that the system loads are similar for these experiments. Each duration reported in Table 1 is approximated from the average of the training time of 10 classifiers.

The number of data points in the training data will determine the size of the QP problems i.e. for data set 1, the QP in the SVM training will have 80 free variables. From the results in Table 1, we can conclude that LOQO is very fast compared to the other two solvers, DONLP2 out-performed the MATLAB QP by a factor of two when the number of variables is small (data set) but as the number of variables increases, it is slower compared to MATLAB QP. Evaluating the performance of these solvers in terms of training results reveals some very surprising facts. Some of the training fails to converge to an optimal solution. The classifiers trained by different solvers give different solutions even though each of these training converged to the optimal solution. This will be investigated in more detail below. In all three sets of experiments, LOQO never arrived at the optimum solution (30 classifiers in total) both MATLAB QP and DONLP2 give optimum solutions in almost all of the classifiers, except for data set 3. The number of failures for each solver in all three sets of experiments is shown in Table 2. MATLAB QP fails slightly more often than DONLP2 and for both solvers, the number of failures increases with the amount of training data. This is consistent with the general expectations of QP problems. As the number of variables increases, the dimensionality of the search space increases. This results in the possibility of non-optimum solution as a consequent of finite numerical precision [12] and [13].

Table 2 Number of training failure out of 10 1-to-rest classifiers in each set of experiments

	LOQO	DONLP2	MATLAB QP
Data Set 1 (80 data points)	10/10	0/10	1/10
Data Set 2 (160 data points)	10/10	0/10	1/10
Data Set 3 (320 data points)	10/10	7/10	7/10

The Peterson & Barney data has feature vectors of dimensions. It is possible to plot the decision boundaries of the SVM classifiers for these data using 2D plots. These plots will be useful in comparing the results of converging and non-converging SVM training. Figures 1 to 6 show the decision boundaries of a few sample classifiers in these experiments. From these boundaries, the following observations are made:

- Although the LOQO training for class 1-to-rest of data set 1 fails to converge the decision boundary (Figure 2) of the resulting classifier is very similar to the optimum classifier (Figure 1).

- In data set 1, MATLAB QP fails to train the classifier for class 10-to-rest. The decision boundary (4) of this classifier is significantly different from the optimum classifier (Figure 3).

- Another non-converging LOQO training (class 2-to-rest), also has different boundaries from the optimum classifier (Figure 5 and 6).

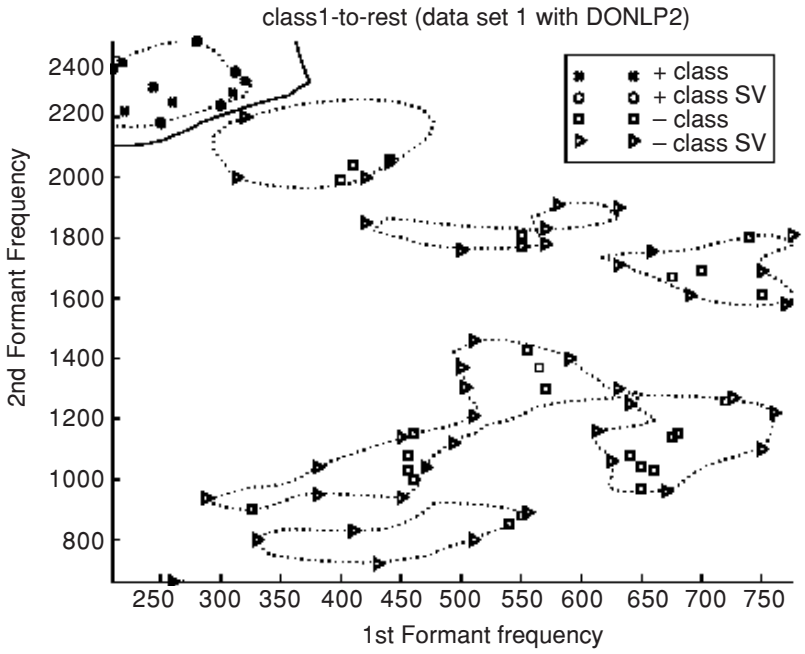

Figure 1 Decision boundary of data set 1 class 1(DONLP2 training converges)

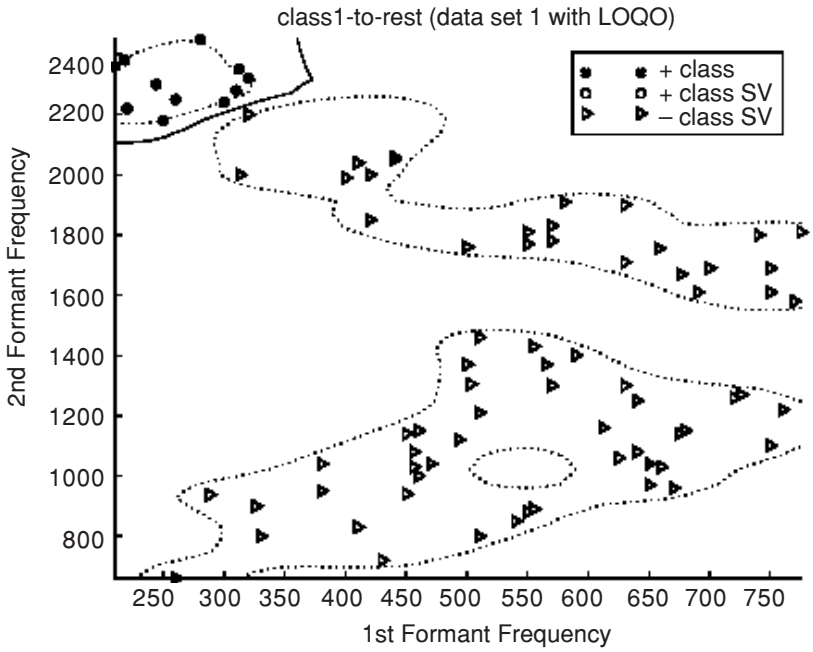

Figure 2 Decision boundary of data set 1 class 1(LOQO training fail to converge)

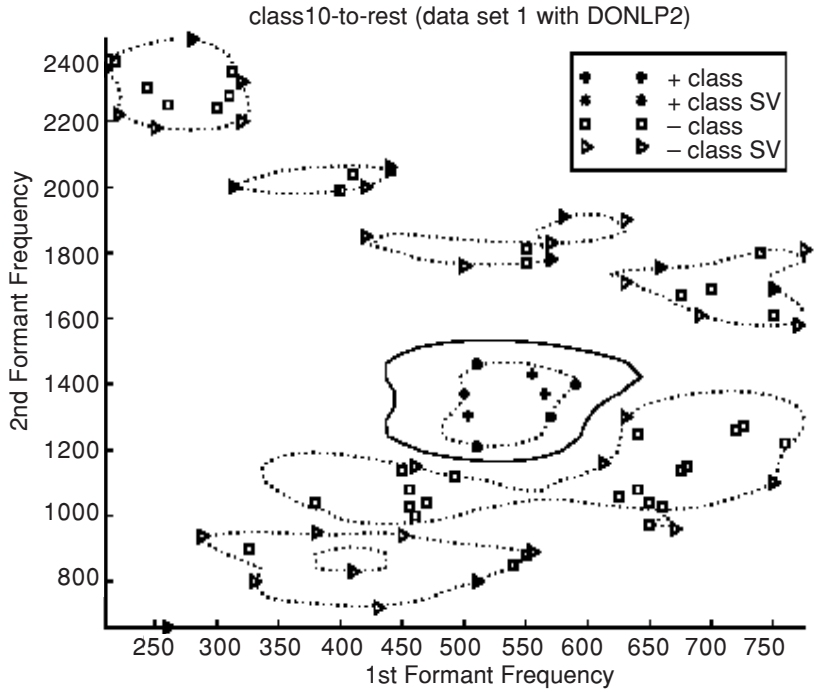

Figure 3 Decision boundary of data set 1class 10 (DONLP2 training converges)

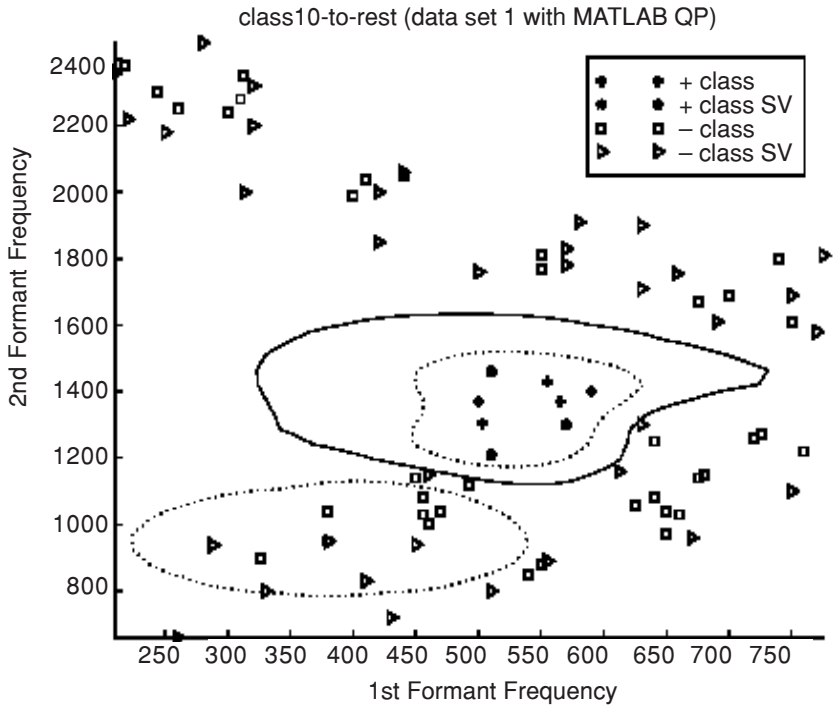

Figure 4 Decision boundary of data set 1class 10 (MATLAB QP training fails to converge)

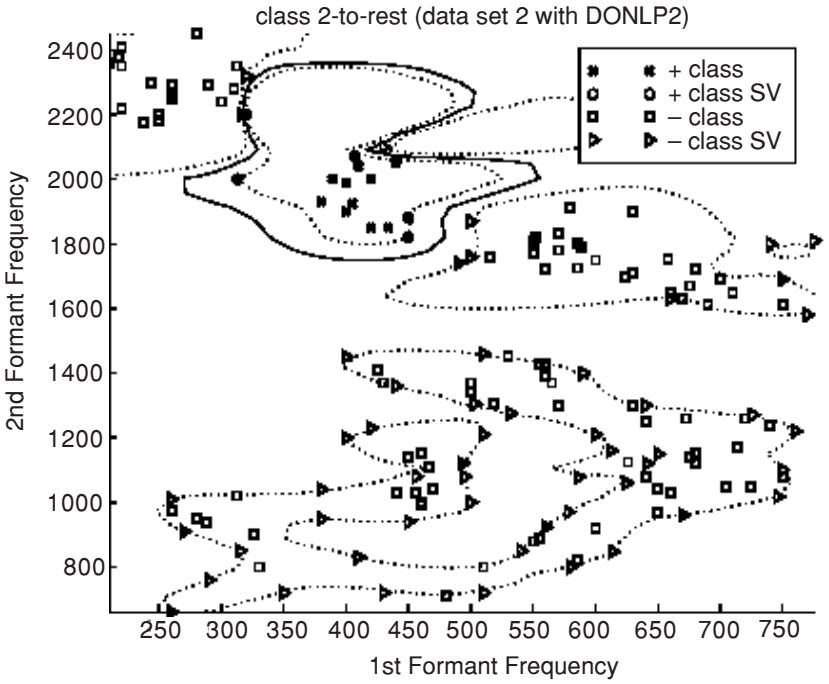

Figure 5 Decision boundary of data set 2 class 2 (DONLP2 training converges)

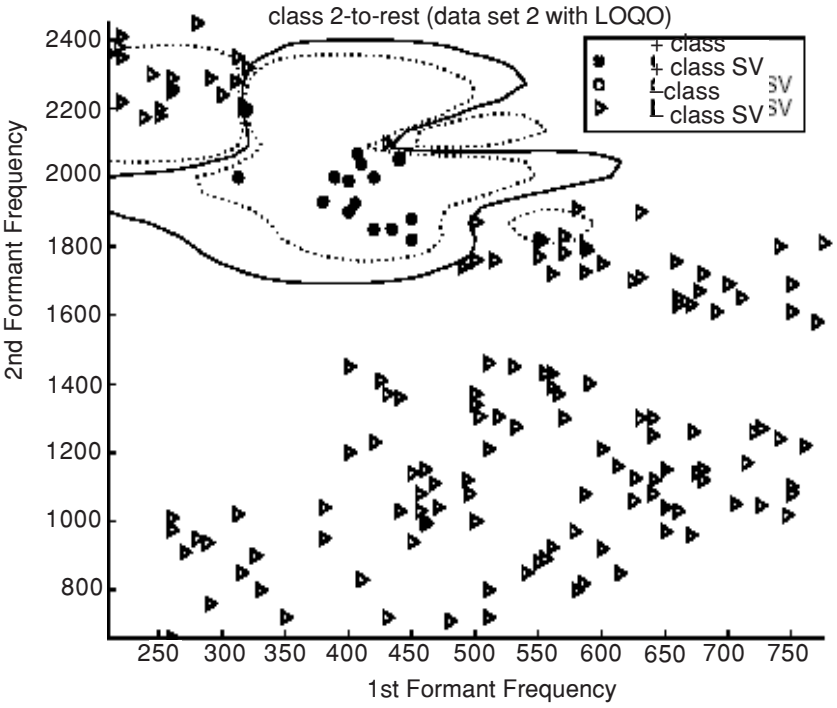

Figure 6 Decision boundary of data set 2 class 2 (LOQO training fail to converge)

From these observations, it is clear that the results of a non-converging training cannot be trusted fully. The decision boundary of the classifier from non-converging training can potentially give very different classification results. A set of test data is used to evaluate the classification accuracy of all these classifier. The accuracy for each set of classifier is shown in Table 3.

Table 3 The accuracy of different sets of classifiers

	LOQO	DONLP2	MATLAB QP
Data Set 1 (80 data points)	77.50%	77.50%	77.50%
Data Set 2 (160 data points)	72.50%	70.50%	72.50%
Data Set 3 (320 data points)	56.88%	60.94%	60.94%

In data set 2 the non-converging LOQO training gives better accuracy than the converging DONLP2 training. From Figures 5 and 6 we conclude that DONLP training over fits the classifier while LOQO training appears to provide better generalization. This observation cannot be taken as conclusive evidence that LOQO always give better generalization. Figures 7 and 8 show that the LOQO training over-generalized and caused high classification error (the accuracy of LOQO training is 4% lower than that of the DONLP training in data set 3).

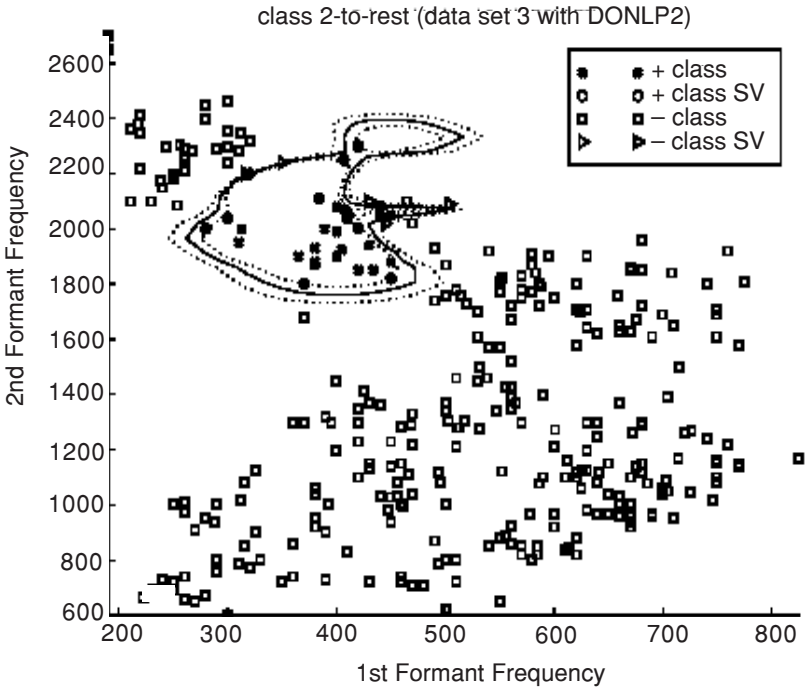

Figure 7 Decision boundary of data set 3 class 2 (DONLP2 training converges)

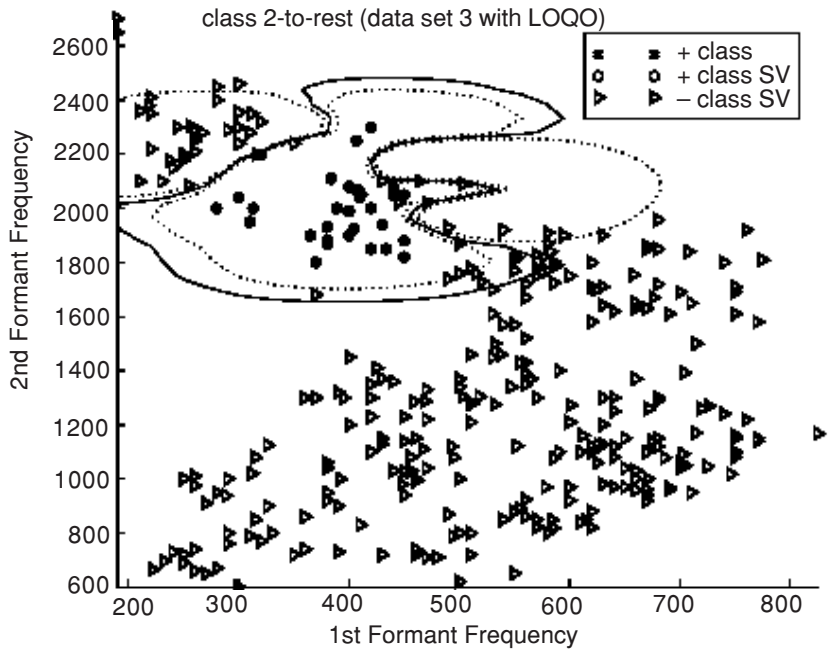

Figure 8 Decision boundary of data set 3 class 2 (LOQO training fail to converge)

The MATLAB QP training gives exactly (the difference in Lagrange Multiplier values is less than 10^{-5}) the same classifier as in the DONLP2 training for data sets 1 and 3, except for class 10-to-rest in data set 1. Table 3 shows that the accuracy for these two sets of classifier is the same. For data set 2, MATLAB QP training gives better accuracy than DONLP2 training.

4.0 REDUCTION OF EXECUTION TIME OF QP SOLVERS

The results in Tables 1 and 2 show that LOQO is very fast but does not give optimum solutions. Since both DONLP2 and MATLAB QP solve the QP problem iteratively, it is possible to initialize these two solvers with the results from LOQO in order to reduce the time required to solve the QP problem. Using the same framework as in Section 4, the results in Table 4 are obtained.

Table 4 Reduction in training duration for solver initialized with results from LOQO

	DONLP2	MATLAB QP
Data Set 1 (80 data points)	17'!6.95 sec	33 '! 30.68 sec
Data Set 2 (160 data points)	240 '! 68.98 sec	251 '! 173.3 sec
Data Set 3 (320 data points)	4200 '! 897.12 sec	3412 '! 2564.3 sec

The classifier trained with the LOQO initialization has the same boundary compared to the classifier trained with zero initialization; this includes the classifier trained by MATLAB QP in data set. The LOQO initialization has set the search off in the same direction as the other two solvers, so it arrives at the same solution. This method of initialization can cause the training to fail and result in zero being assigned to the ë value for each training data.

5.0 CONCLUSION

Support vector machines is found to be a capable learning machine. It has the ability to handle difficult pattern recognition tasks. The formulation of the SVMs is elegant in that it is simplified to a QP problem [14]. In order to study the influence of this quadratic programming problem in the SVMs, three different quadratic programming solvers are used to perform the SVMs training. Based on the analysis and the hypothesis, these observations have some very serious implications; First: the classification boundary in SVM is very sensitive to solution of the QP problem. The solution given by DONLP2 and MATLAB QP training must be very "close" to each other (and also to the actual optimum solution). Second: the goal of SVM training is to obtain the optimum classification boundary. There is no guarantee that any of the training in these experiments does actually arrive at the optimum boundary. The convergence test i.e. the KT condition, is not sufficient to identify these non-optimum results. This problem is fundamentally related to the formulation of the SVM and there is no obvious solution to this problem. The results and analysis in this paper show that the training duration increases very rapidly with the number of training data points (which are equal to the number of variables for the QP problem). Solving a QP problem with a size of more than a few thousand is very challenging in terms of memory requirement and computation cost. New decomposition algorithms have to be used, improved, tested and analyzed in terms of execution time, quality of the solutions, and performance.

REFERENCES

[1] Burges, C.,1998. "A tutorial on support vector machines for pattern recognition". *Data Mining and Knowledge Discovery*. 2(2).

[2] Platt, J. C., 1999. *Fast Training of Support Vector Machines using Sequential Minimal Optimization by Advances in Kernel Methods*. MIT Press.

[3] Vapnik, V., 1995. *The Nature of Statistical Learning Theory*. Springer Verlag, New York.

[4] Osuna, E., R. Freund., and F. Girosi. 1997. "*Support vector machines, Training and applications*". Technical report, MIT AI Lab. CBCL.

[5] Vapnik, V., 1998. *Statistical Learning Theory*. Springer, 1998.

[6] Cristianini, N., and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press.

[7] Spellucci, P., 1996. "*A SQP method for general nonlinear programs using only equality constrained sub-problems*". Technical report, Dept. of Mathematics, Technical University at Darmstadt.

[8] Spellucci, P., 1996. "*A new technique for inconsistent quadratic programming problems in the SQP methods*". Technical report, Dept. of Mathematics, Technical University at Darmstadt.

[9] Vanderbei, R. J., 1994. "*Logo: An interior point code for quadratic programming*". Technical report, Program in Statistics & Operations Research. Princeton University.

[10] Gill, P. E., W. Murray., and M. H. Wright., 1991. *Numerical Linear Algebra and Optimization.* Vol. 1. Addison Wesley.

[11] Peterson, G., H. Barney., 1952. "Control methods used in a study of vowels". *Journal of the Acoustical Society of America.* vol. 24, pp. 175-184.

[12] Fletcher, R., 1990. *"Practical Methods of Optimization"*. Wiley & Sons, Chichester, UK.

[13] More, J. J., G. Toraldo., 1991. "On the solution of large quadratic programming problems with bound constraints". *SIAM J Optimization.*

[14] Zaki, N. M., S. Deris., K. K. Chin, 2002. "Extending the decomposition algorithm for support vector machines training". *Journal of ICT.* 1(1), pp: 17-29.