Efficient Sequential Minimal Optimisation of Support Vector Classifiers

Gavin C. Cawley*[1]

*School of Information Systems, University of East Anglia, Norwich, Norfolk, U.K. NR4 7TJ. E-mail: gcc@sys.uea.ac.uk.

Abstract

This paper describes a simple modification to the sequential minimal optimisation (SMO) training algorithm for support vector machine (SVM) classifiers, reducing training time at the expense of a small increase in memory used proportional to the number of training patterns. Results obtained on real-world pattern recognition tasks indicate that the proposed modification can more than halve the average training time.

1 Introduction

The Support Vector Machine (SVM) classification method has demonstrated impressive results in several difficult, real-world pattern recognition tasks. The Sequential Minimal Optimisation (SMO) algorithm has proved one of the most popular training algorithms for support vector machines, due to its simplicity and competitive training times possible. Platt reports that the run-time of the sequential minimal optimisation algorithm is dominated by evaluation of the kernel function. This paper describes a simple modification that reduces the number of kernel evaluations involving patterns associated with Lagrange multipliers at the upper bound C. This modification can be implemented without a significant increase in the complexity of the code, and is demonstrated to more than halve training time at the expense of a modest increase in storage requirements.

The remainder of this paper is structured as follows: Section 2 provides an overview of the support vector pattern recognition method, introducing the notation adopted in this paper, and section 3 an overview of the sequential minimal optimisation algorithm. Section 4 describes a simple modification improving the efficiency of the sequential minimal optimisation algorithm. Results obtained on real-world benchmark tasks are presented in section 5. The key results are summarised in section 6.

[1]This work was supported by the European Commission, grant number IST-99-11764, as part of its Framework V IST programme.

2 Support Vector Classification

The support vector machine [1, 2], given labelled training data

$$\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{\ell}, \qquad \boldsymbol{x}_i \in \boldsymbol{X} \subset \mathbb{R}^d, \quad y_i \in \{-1, +1\},$$

constructs a maximal margin linear classifier in a high dimensional feature space, $\Phi(\boldsymbol{x})$, defined by a positive definite kernel function, $k(\boldsymbol{x}, \boldsymbol{x}')$, specifying an inner product in the feature space,

$$\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}').$$

A common kernel is the Gaussian radial basis function (RBF),

$$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-||\boldsymbol{x}-\boldsymbol{x}'||^2/2\sigma^2}.$$

The function implemented by a support vector machine is given by

$$f(\boldsymbol{x}) = \left\{ \sum_{i=1}^{\ell} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) \right\} - b. \tag{1}$$

To find the optimal coefficients, $\boldsymbol{\alpha}$, of this expansion it is sufficient to maximise the functional,

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{2}$$

in the non-negative quadrant,

$$0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, \ell, \tag{3}$$

subject to the constraint,

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0. \tag{4}$$

C is a regularisation parameter, controlling a compromise between maximising the margin and minimising the number of training set errors. The

Karush-Kuhn-Tucker (KKT) conditions can be stated as follows:

$$\alpha_i = 0 \quad \Rightarrow \quad y_i f(\boldsymbol{x}_i) \geq 1, \tag{5}$$

$$0 < \alpha_i < C \quad \Rightarrow \quad y_i f(\boldsymbol{x}_i) = 1, \tag{6}$$

$$\alpha_i = C \quad \Rightarrow \quad y_i f(\boldsymbol{x}_i) \leq 1. \tag{7}$$

These conditions are satisfied for the set of feasible Lagrange multipliers, $\boldsymbol{\alpha}^0 = \{\alpha_1^0, \alpha_2^0, \ldots, \alpha_\ell^0\}$, maximising the objective function given by equation 2. The bias parameter, b, is selected to ensure that the second KKT condition is satisfied for all input patterns corresponding to non-bound Lagrange multipliers. Note that in general only a limited number of Lagrange multipliers, $\boldsymbol{\alpha}$, will have non-zero values; the corresponding input patterns are known as support vectors. Let $\mathcal{I}$ be the set of indices of patterns corresponding to non-bound Lagrange multipliers,

$$\mathcal{I} = \{i \quad : \quad 0 < \alpha_i^0 < C\},$$

and similarly $\mathcal{J}$ be the set of indices of patterns with Lagrange multipliers at the upper bound C,

$$\mathcal{J} = \{i \quad : \quad \alpha_i^0 = C\}.$$

Equation 1 can then be written as an expansion over support vectors,

$$f(\boldsymbol{x}) = \left\{ \sum_{i \in \{\mathcal{I}, \mathcal{J}\}} \alpha_i^0 y_i k(\boldsymbol{x}_i, \boldsymbol{x}) \right\} - b. \tag{8}$$

For a full exposition of the support vector method, see the excellent books by Vapnik [3, 4] or Cristianini and Shawe-Taylor [5].

3 Sequential Minimal Optimisation

The sequential minimal optimisation (SMO) algorithm (Platt [6]) implements an extreme form of the decomposition method of Osuna *et al.* [7]. SMO iteratively solves the constrained quadratic optimisation problem, given in equations 2–4, via a series of smaller optimisation problems, each involving a single pair of Lagrange multipliers, which may be solved analytically. The popularity of SMO stems from the competitive training times that can be obtained without the need for a complex numerical optimisation library. The functional given in equation 2 can be written in terms of a pair of Lagrange multipliers, say α_1 and α_2, as

$$\begin{aligned} W(\alpha_1, \alpha_2) &= \alpha_1 + \alpha_2 - \alpha_1 y_1 v_1 - \alpha_2 y_2 v_2 \\ &- \frac{1}{2} K_{11} \alpha_1^2 - \frac{1}{2} K_{22} \alpha_2^2 - s K_{12} \alpha_1 \alpha_2 \\ &+ W_{\text{constant}}, \end{aligned} \tag{9}$$

where

$$v_i = \sum_{j \neq \{1,2\}} \alpha_j y_j k(\boldsymbol{x}_j, \boldsymbol{x}_i),$$

and

$$K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

The linear equality constraint can be expressed in terms of α_1 and α_2 alone as

$$\alpha_1 + s\alpha_2 = \gamma. \tag{10}$$

Each iteration of the sequential minimal optimisation algorithm begins with the selection of a pair of Lagrange multipliers to jointly optimise. The second multiplier, α_2, is then updated to minimise the functional (9), within the feasible region defined by the linear equality and box constraints (10, 3). The first multiplier, α_1, is then updated, substituting the the new value for the second multiplier into the linear equality constraint (10). All that remains is to update the bias term, b, preserving the second Karush-Kuhn-Tucker condition (6).

The first Lagrange multiplier is selected by an outer loop, alternating between single iterations over the entire training set and multiple passes over patterns corresponding to non-bound Lagrange multipliers. A series of passes over non-bound multipliers ends if a pass is made without updating a single Lagrange multiplier. The algorithm terminates following a pass over all training patterns not resulting in a reduction in the objective function. The second multiplier is chosen according to a sequence of heuristics: The first heuristic selects a non-bound multiplier corresponding to the error maximising $|E_1 - E_2|$, where E_i is the error for the i^{th} training pattern, as this is likely to increase the magnitude of the update to the second multiplier. If no progress is made using the first choice, the algorithm iterates over the remaining non-bound multipliers, starting from a random location. If this also fails the algorithm iterates over all bound multipliers, again starting from a random location.

4 Method

Platt reports that the run-time of the sequential minimal optimisation algorithm is dominated by evaluation of the kernel function. Evaluation of the kernel function, involving a training pattern corresponding to a Lagrange multiplier at the upper bound ($\alpha_i = C, i \in \mathcal{J}$), arises in two situations; firstly during joint optimisation of the Lagrange multipliers associated with such patterns, but also whenever the output of the support vector machine is evaluated. The former situation is

unavoidable, and is most likely to occur only during the intermittent passes through the entire training set made by the outer loop of the SMO algorithm. The second situation, however is likely to result in a significant proportion of kernel function evaluations, simply because each joint optimisation involving a Lagrange multiplier at the upper or lower bound will result in at least one evaluation of the output function. The support vector expansion given by (8) can be written as

$$f(\boldsymbol{x}) = -b + \sum_{i \in \mathcal{I}} \alpha_i^0 y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + C \sum_{i \in \mathcal{J}} y_i k(\boldsymbol{x}_i, \boldsymbol{x}).$$

Note that the value of the second summation is likely to remain constant for much of the time during the training procedure. We therefore cache the value of this term for each training pattern, updated only when a Lagrange multiplier reaches or leaves the upper bound, C, during a joint optimisation (note this involves evaluation of only a single column of the kernel matrix). This is an example of a time-space optimisation, improving speed at the expense of additional storage requirements, however the costs are low (in the author's implementation, an additional double precision floating point variable for each training pattern).

5 Results

In this section we present results obtained using the modified and standard sequential minimal optimisation algorithms on a publically available bench mark data set. Both the standard sequential minimal optimisation algorithm (Platt [6]) and the enhanced SMO algorithm, described in the previous section, were implemented in the Java programming language (Arnold *et al.* [8]). Figure 1 shows a graph of training time as a function of the number of training patterns for support vector classification networks trained on the Adult benchmark problem. A Gaussian radial basis kernel was used in each case, with a variance of 10, and with the regularisation parameter, C, set at unity, conforming to the experimental conditions used in (Platt [6]). Ten networks were trained for each of the eight partitions of the data, the mean training times for each partition are shown in Figure 1.

It can easily be seen that the enhanced sequential minimal optimisation algorithm is significantly faster than the standard algorithm for all partitions of the data, the improvement in performance increasing slightly as the number of training patters increases. The variability in training time using the enhanced algorithm also appears to be reduced. It

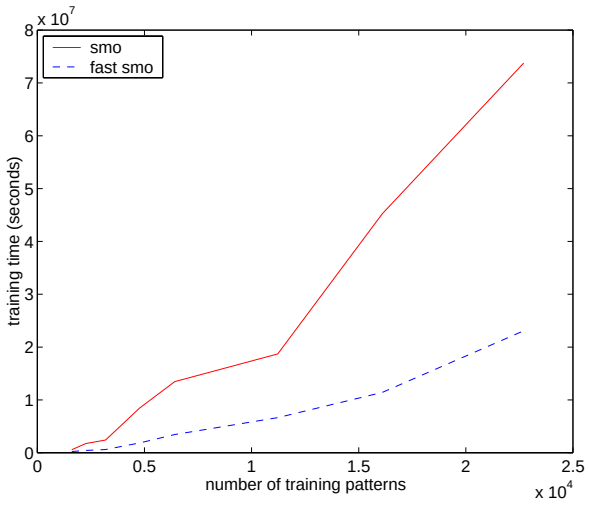

Fig. 1. Mean training time for 10 support vector machines trained on the Adult bench mark data set, as a function of the number of training patterns used.

appears that the sequential minimal optimisation algorithm sometimes experiences difficulty converging to the optimal solution, resulting in a large number of passes through the entire data set. The proposed modification is most useful in such circumstances as it reduces the cost of evaluations of the support vector expansion.

6 Summary

This paper describes a simple method improving the efficiency of the sequential minimal optimisation algorithm for support vector classification. This technique is likely to be most useful for difficult pattern recognition tasks, with a high degree of overlap in the distribution of patterns belonging to each class and therefore a large number of bound support vectors. Naturally the improvement in training time will be greatest for computationally expensive kernel functions, and smallest (i.e. nil) in the case of linear support vector machines, where in principle the output function can be computed without evaluating the kernel function [6].

7 Acknowledgements

The author would like to thank Rob Foxall for his helpful comments on previous drafts of this manuscript.

References

[1] B. Boser, I. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop*

on computational learning theory, (Pittsburgh), pp. 144–152, ACM, 1992.

[2] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 1–25, 1995.

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

[4] V. N. Vapnik, *Statistical Learning Theory*. Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications and Control, New York: Wiley, 1998.

[5] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge, U.K.: Cambridge University Press, 2000.

[6] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning* (B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds.), ch. 12, pp. 185–208, Cambridge, Massachusetts: MIT Press, 1999.

[7] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Neural Networks for Signal Processing VII - Proceedings of the 1997 IEEE Workshop* (J. Principe, L. Gile, N. Morgan, and E. Wilson, eds.), (New York), pp. 276–285, IEEE, 1997.

[8] K. Arnold, J. Gosling, and D. Holmes, *The Java Programming Language*. Addison-Wesley, third ed., 2000.