Fast Model Selection for MaxMinOver-based Training of Support Vector Machines

Fabian Timm Sascha Klement Thomas Martinetz
Institute for Neuro- and Bioinformatics, University of Lübeck,
Ratzeburger Allee 160, 23538 Lübeck, Germany

Abstract

OneClassMaxMinOver (OMMO) is a simple incremental algorithm for one-class support vector classification. We propose several enhancements and heuristics for improving model selection, including the adaptation of well-known techniques such as kernel caching and the evaluation of the feasibility gap. Furthermore, we provide a framework for optimising grid search based model selection that compromises of preinitialisation, cache reuse, and optimal path selection.

Finally, we derive simple heuristics for choosing the optimal grid search path based on common benchmark datasets. In total, the proposed modifications improve the runtime of model selection significantly while they are still simple and adaptable to a wide range of incremental support vector algorithms.

1. Introduction

The support vector machine [10] has become a standard approach for dealing with pattern recognition tasks in various areas. By now, several techniques for solving the quadratic programming problem are available, such as Sequential Minimal Optimisation (SMO) [5] or MaxMinOver [4]. In practice, we need – besides these core algorithms – supplementary techniques to increase efficiency on large datasets or for fast parameter validation.

In the present work, we describe several enhancements to MaxMinOver-based approaches for improving parameter validation, including a more complex stopping criterion, kernel caching, preinitialisation of the Lagrangian variables and the kernel cache as well as an optimised grid search sequence for model selection. We focus on OneClassMaxMinOver (OMMO) [3] due to its simplicity, although these techniques can be applied to all MaxMinOver-based approaches and – with minor changes – even to a wider range of training algorithms.

1.1. The Support Vector Framework

In the following, we make use of the common support vector framework restricted to one-class problems consisting of training samples $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$, a transformation $\phi(\mathbf{x}_i)$ into a higher dimensional kernel space, a kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, a regularisation parameter C for soft-margin classification, and slack variables ξ_i for non-separable cases.

Whereas several approaches to one-class support vector classification rely on a 1-norm slack term [9, 7], also a 2-norm slack term is reasonable [3]. In this case the optimisation problem can be expressed as

$$\min_{\mathbf{w},\xi} \left(\frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\|\xi\|_2^2 \right) \text{ s.t.} \quad (1)$$
$$\forall i : \mathbf{w}^T \phi(\mathbf{x}_i) \geq 1 - \xi_i \ , \quad \text{and} \quad \xi_i \geq 0 \ .$$

By setting the partial derivatives

$$\mathbf{w} = \sum_i \alpha_i \, \phi(\mathbf{x}_i) \quad \text{and} \quad \xi_i = \frac{\alpha_i}{C} \quad (2)$$

of the corresponding Lagrangian to zero and resubstituting (2) into (1), we obtain the dual optimisation problem

$$\min_{\alpha} \left(\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K^*(\mathbf{x}_i, \mathbf{x}_j) \right) \quad (3)$$
$$\text{s.t.} \quad \forall i : \alpha_i \geq 0 \ .$$

Here, $K^*(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C}$ defines the *modified* kernel while δ_{ij} is Kronecker's delta. Henceforth, the original and the modified kernel will be denoted by $\mathbf{K}$ and $\mathbf{K}^*$ respectively, i.e. $\mathbf{K}_{ij}^{(*)} = K^{(*)}(\mathbf{x}_i, \mathbf{x}_j)$.

The ξ_i need to be evaluated explicitly to derive a proper stopping criterion of the optimisation via the

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

Karush-Kuhn-Tucker (KKT) complementarity conditions:

$$\forall i \,:\, \alpha_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) - 1 + \xi_i \right) = 0 \ . \qquad (4)$$

In the optimum

$$\xi_i = \begin{cases} 1 - \mathbf{w}^T \phi(\mathbf{x}_i) & \text{if} \quad \alpha_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

holds due to (2) and (4). For intermediate solutions during optimisation, we need to set

$$\begin{aligned} \xi_i &= \max\left(0,\ 1 - \mathbf{w}^T \phi(\mathbf{x}_i)\right) \\ &= \max\left(0,\ 1 - \sum_j \alpha_j \, K(\mathbf{x}_i, \mathbf{x}_j)\right) \end{aligned}$$

to fulfil the constraints of the primal problem. For simplicity we set

$$\mathbf{s} = (s_1, \ldots, s_N)^T = \mathbf{K}^* \alpha = \mathbf{K}\alpha + \frac{\alpha}{C}$$

such that $\xi_i = \max(0, 1 - s_i - \frac{\alpha_i}{C})$.

1.2. OneClassMaxMinOver

The OMMO algorithm (see Alg. 1) is inspired by the MaxMinOver algorithm for two-class classification proposed in [4]. Each training sample $\mathbf{x}_i$ is associated with a counter variable α_i that can be interpreted as a Lagrangian variable in (3). Within each iteration step a variable is increased if its associated sample is closest to the decision border in terms of the distance measure s. On the other hand, a variable is decreased if it is non-zero and its associated sample is the farthest away from the decision border. Since we need the correct ξ_i for deriving a stopping criterion we have to scale the α_i such that the smallest distance of a point in the modified kernel space to the origin is one.

OMMO converges to the support vector solution with at least $\mathcal{O}(1/\sqrt{t})$ as shown in [3]. A new sample $\mathbf{x}$ is classified by $f(\mathbf{x}) = \text{sign}\left(\sum_i \alpha_i \, K(\mathbf{x}, \mathbf{x}_i) - 1\right)$.

Algorithm 1: OneClassMaxMinOver

$\alpha \leftarrow \mathbf{0}$
for $t \leftarrow 0, \ldots, t_{\max}$ **do**
  $\mathbf{s} \leftarrow \mathbf{K}^* \alpha$
  $i_{\min} \leftarrow \arg\min_i s_i, \quad i_{\max} \leftarrow \arg\max_{i,\, \alpha_i > 0} s_i$
  $\alpha_{i_{\min}} \leftarrow \alpha_{i_{\min}} + 2, \quad \alpha_{i_{\max}} \leftarrow \alpha_{i_{\max}} - 1$
end
$\mathbf{s} \leftarrow \mathbf{K}^* \alpha, \quad \rho \leftarrow \min_i s_i, \quad \alpha \leftarrow \frac{\alpha}{\rho}$

1.3. Stopping Criteria

Several ways for defining stopping criteria of an iterative support vector approach such as OMMO have been proposed in the literature, e.g. monitoring the growth of the dual objective function (3), monitoring the KKT conditions for the primal problem (4), or measuring the feasibility gap. See chapter 7 of [1] and chapter 10 of [8] for a more detailed discussion on stopping criteria.

We will focus on the so-called feasibility gap which is defined as the difference between the values of the primal (1) and the dual objective function (3):

$$\begin{aligned} & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\xi^T\xi - \mathbf{e}^T\alpha + \frac{1}{2}\alpha^T\mathbf{K}^*\alpha \\ = \ & \alpha^T\mathbf{s} - \frac{1}{2\,C}\alpha^T\alpha + \frac{C}{2}\xi^T\xi - \mathbf{e}^T\alpha \ , \end{aligned}$$

with $\mathbf{e} = (1, ..., 1)^T$. According to [1] a useful measure of progress is

$$\begin{aligned} & \frac{\text{primal} - \text{dual}}{\text{primal} + 1} \\ = \ & \frac{2\,\alpha^T\mathbf{s} - \frac{1}{C}\alpha^T\alpha + C\xi^T\xi - 2\,\mathbf{e}^T\alpha}{\alpha^T\mathbf{s} + C\xi^T\xi + 2} \end{aligned} \qquad (5)$$

In the following two sections we describe the OMMO algorithm with several enhancements and show that evaluating the feasibility gap comes almost without any extra cost in time complexity.

2. Optimising OMMO

First of all, OMMO simply provides a way for deriving the support vector solution of a single parameter combination. We will now wrap OMMO into a framework for speeding up parameter selection while still preserving its simplicity (see Alg. 2). First, incremental evaluation and a kernel cache are well-known techniques for improving time complexity of the learning algorithm itself. Second, the algorithm terminates as soon as the feasibility gap falls below a threshold. Finally, we initialise the α_i and the kernel cache with the slightly modified results of a previous parameter combination and choose a grid search path which minimises a specific cost function.

In detail, we use the following techniques to optimise runtime performance of the model selection.

Kernel Caching and Incremental Evaluation The kernel evaluations are normally the most time consuming parts of naive implementations. Therefore, the kernel cache stores frequently used values for later usage.

Algorithm 2: OneClassMaxMinOver2

while $\mathrm{gap}_{\mathrm{curr}} > \mathrm{gap}_{\mathrm{stop}}$ **do**

$\mathbf{s} \leftarrow \mathbf{K}^* \alpha$

$i_{\min} \leftarrow \arg\min_i s_i, \quad i_{\max} \leftarrow \arg\max_{i,\, \alpha_i > 0} s_i$

$\alpha_{i_{\min}} \leftarrow \alpha_{i_{\min}} + 2, \quad \alpha_{i_{\max}} \leftarrow \alpha_{i_{\max}} - 1$

A $\quad \rho \leftarrow \min_i s_i$

B $\quad \xi = \max(\mathbf{0}, \mathbf{1} - (\mathbf{s} - \alpha/C)/\rho)$

C $\quad \mathrm{gap}_{\mathrm{curr}} = \frac{\frac{2}{\rho^2}\alpha^T\mathbf{s} - \frac{1}{C\,\rho^2}\alpha^T\alpha + C\xi^T\xi - \frac{2}{\rho}\mathbf{e}^T\alpha}{\frac{1}{\rho^2}\alpha^T\mathbf{s} + C\xi^T\xi + 0.5}$

end

$\mathbf{s} \leftarrow \mathbf{K}^* \alpha, \quad \rho \leftarrow \min_i s_i, \quad \alpha \leftarrow \frac{\alpha}{\rho}$

Additionally, $\mathbf{s}$ can be evaluated incrementally by

$$s_i^{new} = s_i^{old} + 2K(\mathbf{x}_{i_{\min}}, \mathbf{x}_i) - K(\mathbf{x}_{i_{\max}}, \mathbf{x}_i) \ .$$

The size of the cache is crucial depending on the number of support vectors of the final solution. As this incremental summation is not numerically stable, a non-incremental recomputation of $\mathbf{s}$ should be done after a fixed number of iterations t_{re}. The extra cost in time complexity of this recomputation can be neglected as a good choice for t_{re} is 1000.

Evaluating the feasibility gap (lines A–C in Alg. 2) takes time $\mathcal{O}(N)$ due to the dot products. However, it is sufficient to compute the gap in regular intervals, e.g. every t_{re} steps, so that the runtime of the whole algorithm does not change significantly.

Preinitialisation With grid search one evaluates the performance of a machine learning algorithm for different parameter sets sampled on a regular grid. An optimal parameter set is chosen according to common performance measures, such as the receiver-operator-characteristic or the leave-one-out error. Normally, adjacent nodes in the grid yield similar support vector solutions. So, we initialise OMMO with the solution of a previous node, in this case with the unscaled α_i. Furthermore, if a transformation between kernel values of different parameter exists and if it has lower time complexity than a complete recalculation, then the kernel cache can also be reused. For Gaussian kernels

$$\tilde{\mathbf{K}}_{ij} = \left(\mathbf{K}_{ij} - \frac{\delta_{ij}}{C_1}\right)^{\sigma_1^2 \,/\, \sigma_2^2} + \frac{\delta_{ij}}{C_2} \qquad (6)$$

transforms the kernel values of the parameter tuple (C_1, σ_1) into those of the tuple (C_2, σ_2).

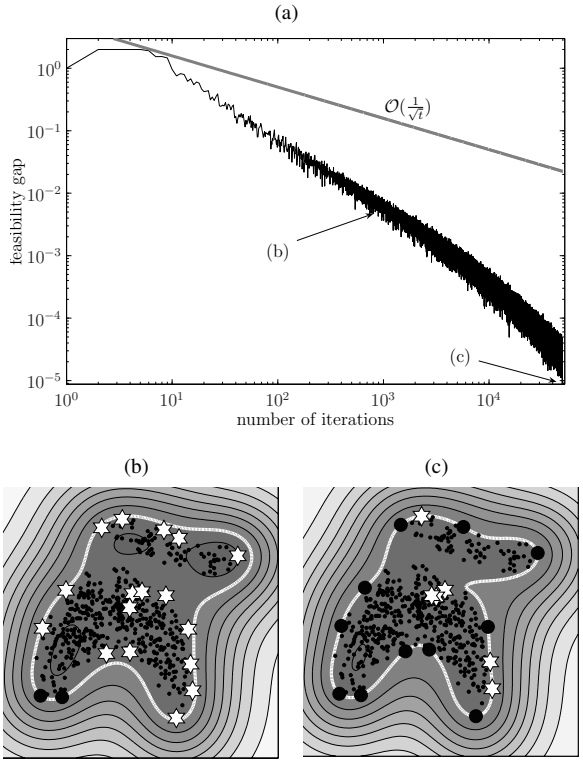


Figure 1. Feasibility gap for the modified banana dataset (a) and two intermediate solutions (b) and (c).

3. Experiments and Results

3.1. Feasibility Gap

We show the decrease of the feasibility gap (5) exemplarily for the well-know banana dataset [6], limited to 500 randomly chosen samples from class $+1$. In this scenario, we set $C = 1000$, $\sigma = 0.4$, $\mathrm{gap}_{\mathrm{stop}} = 10^{-5}$ and $t_{\mathrm{re}} = 1$ in order to compute the feasibility gap within each iteration.

Figure 1 shows the feasibility gap and two intermediate solutions (white line: class boundary; large dark dots: support vectors outside; stars: support vectors inside; small dots: non-support vectors). As OMMO converges with at least $\mathcal{O}(1/\sqrt{t})$, the feasibility gap is also upper bounded by $\mathcal{O}(1/\sqrt{t})$.

3.2. Validation of Hyperparameters

An optimal grid search should make intensive use of preinitialisation and kernel reuse. So, we want to find a directed spanning tree of the grid which has min-

imum runtime if we travel from the root along the edges. For evaluating the reliability of this method we used 13 benchmark datasets [6] containing artificial as well as real world data. Each dataset was separated class-specifically and scaled to unit mean norm. A logarithmically scaled parameter grid with 20×20 nodes ($C \in [1, 10^5]$, $\sigma \in [0.1, 5]$, $\mathrm{gap}_{\mathrm{stop}} = 10^{-4}$) representing maximum uncertainty was used throughout the experiment.

A reasonable cost function for describing the computational effort to travel between solutions of different parameter sets is

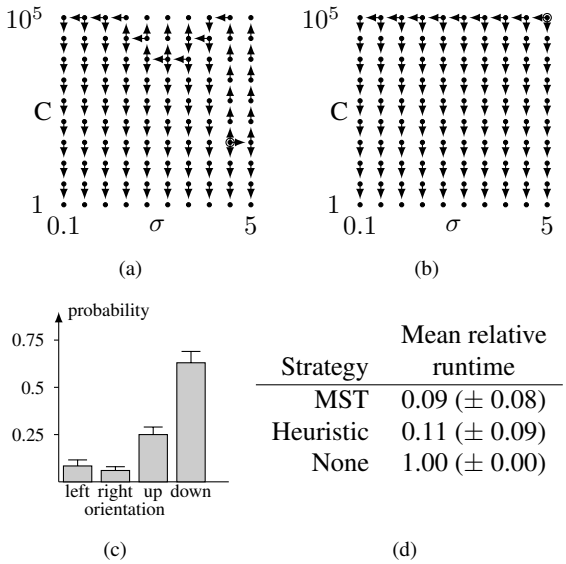
$$c(i,j) = \begin{cases} \mathrm{time}_i(j) & \text{if nodes } i \text{ and } j \text{ are neighbours} \\ \infty & \text{otherwise} \end{cases}$$

where $\mathrm{time}_i(j)$ describes the runtime of OMMO for grid node i, preinitialised with the results of node j, which itself was trained without preinitialisation.

Via the cost matrix, the minimum spanning tree (MST) was determined for every dataset by Edmonds' algorithm [2] with each node as a root node. Figure 2(a) shows exemplarily the MST of the banana dataset. Obviously, vertical edges are preferred, i.e. support vector solutions for adjacent kernel widths differ more than those of adjacent softness parameters. This behaviour remains the same for a variety of datasets, but could change if the grid spacing or the parameter scaling would vary by orders of magnitude. Evaluating the MSTs for different datasets indicates that the general heuristic (see Fig. 2(b)) is very close to the optimal spanning tree. Moreover, the probabilities of the four edge orientations in the MST differ significantly (see Fig. 2(c)) and make the heuristic reasonable. The best improvement is achieved when training along the MST, but according to table 2(d) the mean runtime when using the heuristic is about 11% of the runtime without preinitialisation.

4. Conclusions

Based on the OMMO algorithm we derived a modified algorithm which uses the feasibility gap as stopping criterion. Further enhancements such as kernel caching, preinitialisation, and a grid search heuristic now qualify the whole one-class framework for fast model selection so that the runtime for a complete grid search is reduced to 11%. Besides, deriving a grid search heuristic by means of a cost function and the minimum spanning tree can be applied to other learning algorithms or higher-dimensional parameter sets such as in support vector regression. Further improvements might include the online adaptation of a heuristic during grid search or to split the heuristic to enable fast parallelisation.



Strategy	Mean relative runtime
MST	0.09 ($\pm$ 0.08)
Heuristic	0.11 ($\pm$ 0.09)
None	1.00 ($\pm$ 0.00)

Figure 2. Results of grid evaluation.

References

[1] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, U.K., 2000.

[2] J. Edmonds. Optimum branchings. *J. Res. Nat. Bur. Standards*, 71B:233–240, 1967.

[3] K. Labusch, F. Timm, and T. Martinetz. Simple incremental one-class support vector classification. In *DAGM-Symposium*, Lecture Notes in Computer Science, pages 21–30. Springer, 2008.

[4] T. Martinetz. MaxMinOver: A Simple Incremental Learning Procedure for Support Vector Classification. In *IJCNN 2004*, pages 2065–2070, Budapest, Hungary, 2004.

[5] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[6] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. Technical Report NC-TR-1998-021, Department of Computer Science, University of London, Aug. 1998.

[7] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[8] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[9] D. M. J. Tax and R. P. W. Duin. Data domain description using support vectors. In *ESANN*, pages 251–256, 1999.

[10] V. N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, Heidelberg, DE, 1995.