GenSVM: A Generalized Multiclass Support Vector Machine*

G.J.J. van den Burg† and P.J.F. Groenen‡

Econometric Institute
Erasmus University Rotterdam
P.O. Box 1738
3000 DR Rotterdam
The Netherlands

Econometric Institute Report EI 2014-33

December 18, 2014

Abstract

Traditional extensions of the binary support vector machine (SVM) to multiclass problems are either heuristics or require solving a large dual optimization problem. Here, a generalized multiclass SVM called GenSVM is proposed, which can be used for classification problems where the number of classes K is larger than or equal to 2. In the proposed method, classification boundaries are constructed in a $K-1$ dimensional space. The method is based on a convex loss function, which is flexible due to several different weightings. An iterative majorization algorithm is derived that solves the optimization problem without the need of a dual formulation. The method is compared to seven other multiclass SVM approaches on a large number of datasets. These comparisons show that the proposed method is competitive with existing methods in both predictive accuracy and training time, and that it significantly outperforms several existing methods on these criteria.

Keywords: Support Vector Machines (SVMs), Multiclass Classification, Iterative Majorization, MM Algorithm, Classifier Comparison

*This report is a preprint. It is not a formal publication in any way, and it will be published elsewhere.

†Email: burg@ese.eur.nl.

‡Email: groenen@ese.eur.nl.

1. Introduction

For binary classification, the support vector machine has shown to be very successful (Cortes and Vapnik, 1995). The SVM efficiently constructs linear or nonlinear classification boundaries and is able to yield a sparse solution through the so-called support vectors, that is, through those observations that are either not perfectly classified or are on the classification boundary. In addition, by regularizing the loss function, the overfitting of the training dataset is curbed.

Due to its desirable characteristics several attempts have been made to extend the SVM to classification problems where the number of outcomes K is larger than two. Overall, these extensions differ considerably in the approach taken to include multiple classes. Three types of approaches for multiclass SVMs can be distinguished. First, there are heuristic approaches that use the binary SVM as an underlying classifier and decompose the K-class problem into multiple binary problems. The most commonly used heuristic is the one-vs-one (OvO) method where decision boundaries are constructed between each pair of classes (Kreßel, 1999). OvO requires solving $K(K-1)$ binary SVM problems, which can be substantial if the number of classes is large. An advantage of OvO is that the problems to be solved are smaller in size. On the other hand, the one-vs-all (OvA) heuristic constructs K classification boundaries, one separating each class from all the other classes (Vapnik, 1998). Although OvA requires fewer binary SVMs to be estimated, the complete dataset is used for each classifier, which can create a high computational burden. Another heuristic approach is the directed acyclic graph (DAG) SVM proposed by Platt et al. (2000). DAGSVM is similar to the OvO approach except that the class prediction is done by successively voting away unlikely classes until only one remains. Thus, the DAGSVM classifier uses a more efficient path to predict the class label than the OvO method does, which uses a voting strategy.

In practice, heuristic methods such as the OvO and OvA approaches are used more often than other multiclass SVM implementations. One of the reasons for this is that there are several software packages that efficiently solve the binary SVM, such as LibSVM (Chang and Lin, 2011). This package implements the sequential minimal optimization algorithm of Platt (1999). Implementations of other multiclass SVMs in high-level (statistical) programming languages are lacking, which reduces their use in practice[1].

The second type of extension of the binary SVM are the error correcting codes. Here, the problem is decomposed into multiple binary classification problems based on a constructed coding matrix which determines the grouping of the classes in a specific binary problem (Dietterich and Bakiri, 1995; Allwein et al., 2001; Crammer and Singer, 2002a). Error correcting code SVMs can therefore be seen as a generalization of OvO and OvA. In Dietterich and Bakiri and the generalization by Allwein et al., a coding matrix is constructed that determines which class instances are paired against each other for each binary SVM. Both approaches require that the coding matrix is determined beforehand. However, it is a priori unclear how such a coding matrix should be chosen. In fact, as Crammer and Singer (2002b) show, finding the optimal coding matrix is an NP-complete problem.

[1] An exception to this is the method of Lee et al. (2004), for which an `R` implementation exists. See `http://www.stat.osu.edu/~yklee/software.html`.

The third type of approaches are those that optimize one loss function to estimate all class boundaries simultaneously, the so-called single machine approaches. In the literature, such methods have been proposed by Weston and Watkins (1998), Bredensteiner and Bennett (1999), Crammer and Singer (2002a), Lee et al. (2004), and Guermeur and Monfrini (2011). The method of Weston and Watkins yields a fairly large quadratic problem with a large number of slack variables, that is, $K-1$ slack variables for each observation. The method of Crammer and Singer reduces this number of slack variables by only penalizing the largest misclassification error. In addition, their method does not include a bias term in the decision boundaries, which is advantageous for solving the dual problem. Interestingly, this approach does not reduce to the same number of parameters as the binary SVM for $K = 2$. The method of Lee et al. uses a sum-to-zero constraint on the decision functions to reduce the dimensionality of the problem. This constraint effectively means that the solution of the multiclass SVM lies in a $K-1$ dimensional subspace of the full K dimensions considered. The size of the margins is reduced according to the number of classes, such that asymptotic convergence is obtained to the Bayes optimal decision boundary when the regularization term is ignored (Rifkin and Klautau, 2004). Finally, the method of Guermeur and Monfrini is a quadratic extension of the method developed by Lee et al. This extension keeps the sum-to-zero constraint on the decision functions, drops the nonnegativity constraint on the slack variables, and adds a quadratic function of the slack variables to the loss function. This means that at the optimum the slack variables are only positive on average, which is different from common SVM formulations.

The existing approaches to multiclass SVMs suffer from several problems. All current single machine multiclass extensions of the binary SVM rely on solving a potentially large dual optimization problem. However, dualization can be disadvantageous when a solution has to be found in a small amount of time. By iteratively improving the dual solution it is not guaranteed that the primal solution is improved as well. Thus, stopping early can lead to poor predictive performance. In addition, the dual of such single machine approaches should be solvable quickly in order to compete with existing heuristic approaches.

Almost all single machine approaches rely on misclassifications of the observed class with each of the other classes. By simply summing these misclassification errors (as in Lee et al., 2004) observations with multiple errors contribute more than those with a single misclassification do. Consequently, observations with multiple misclassifications have a stronger influence on the solution than those with a single misclassification, which is not a desirable property for a multiclass SVM, as it overemphasises objects that are misclassified with respect to multiple classes. Here, it is argued that there is no reason to penalize certain misclassification regions more than others.

Single machine approaches are preferred for their ability to capture the multiclass classification problem in a single model and optimize over it. A parallel can be drawn here with multinomial regression and logistic regression. In this case, multinomial regression reduces exactly to the binary logistic regression method when $K = 2$, both techniques are single machine approaches, and many of the properties of logistic regression extend to multinomial regression. Therefore, it can be considered natural to use a single machine approach for the multiclass SVM which reduces parsimoniously to the binary SVM when presented with a two-class problem.

The idea of casting the multiclass SVM problem to $K-1$ dimensions is appealing, since it reduces the dimensionality of the problem and is also present in other multiclass classification methods such as multinomial regression and linear discriminant analysis. However, the sum-to-zero constraint employed by Lee et al. (2004) creates an additional burden on the dual optimization problem (Dogan et al., 2011). Therefore, it would be desirable to cast the problem to $K-1$ dimensions in another manner, as will be proposed below. The low dimensional projection also has advantages for understanding the method, since it allows for a geometric interpretation. The geometric interpretation of existing single machine multiclass SVMs is often difficult since most are derived based on a dual optimization approach with little attention for a primal problem based on hinge errors.

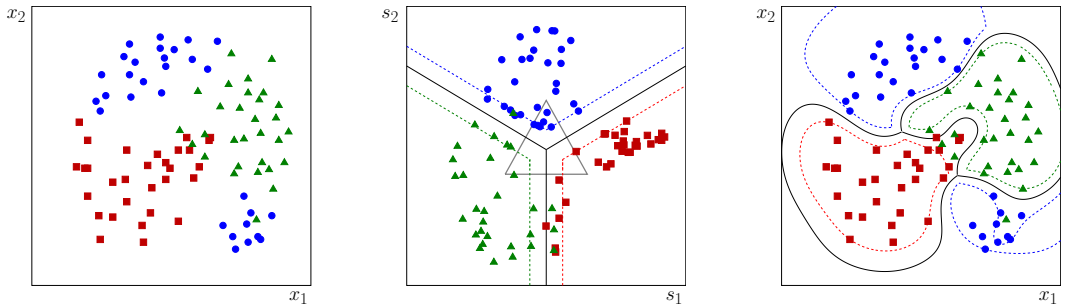
A new flexible and general multiclass SVM is proposed, which it called GenSVM. This single machine multiclass SVM reduces to the binary SVM if there are only two classes. It is based on similar ideas as the binary SVM and can be geometrically well understood in the $K-1$ simplex space of the K classes. In the linear variant, $K-1$ linear combinations of the predictor attributes are estimated, including the bias terms. A nonlinear version is formed by using kernels in a similar manner as can be done for binary SVMs. The associated loss function of GenSVM is convex in the parameters to be estimated. GenSVM is general in the sense that it subsumes multiclass SVMs from the literature that use a sum of the classical hinge functions, it allows different hinge functions such as quadratic hinge errors, and can be set to use the Euclidean distance to the boundary of the class region as an error. A novel iterative majorization (IM) algorithm is provided, which guarantees descent in each iteration towards a global minimum. The iterative improvements make this algorithm competitive in speed for reasonably sized problems, in particular in combination with cross validation.

To evaluate GenSVM, it is compared to seven other multiclass SVMs mentioned above. Existing comparisons of multiclass SVMs fail to determine any statistically significant differences in performance between classifiers, and resort to tables of accuracy rates for the comparisons (e.g. Hsu and Lin, 2002). Using suggestions from the literature, performance and training time of all classifiers is compared using performance profiles and rank tests. The rank tests can be used to uncover statistically significant differences between classifiers.

This paper is organized as follows. Section 2 introduces the novel generalized multiclass SVM. In Section 3, the iterative majorization theory is reviewed and a number of useful properties are highlighted. Section 4 derives the IM algorithm for GenSVM, and presents pseudocode for the algorithm. Extensions of GenSVM to nonlinear classification boundaries are discussed in Section 5. A numerical comparison of GenSVM with existing multiclass SVMs on empirical datasets is done in Section 6. Section 7 concludes the paper.

2. GenSVM

Before introducing GenSVM formally, consider a small illustrative example of a hypothetical data set of 90 objects with $K = 3$ classes and $m = 2$ attributes. Figure 1(a) shows the dataset in the space of these two predictor attributes, x_1 and x_2. Then, Figure 1(b) gives a linear combination of the kernel space of the two predictor attributes, denoted s_1 and s_2, which are formed by some linear combination of the kernel mapping of x_1 and x_2. In the middle, a triangle is drawn that corresponds to a regular K simplex in $K-1$ dimensions.



(a) Input space **(b)** Simplex space **(c)** Input space with boundaries

Figure 1: Illustration of the novel multiclass SVM for a 2D dataset with three classes. In (a) the original data is shown. Figure (b) shows the same data in the simplex space, where the optimal decision boundaries have been determined (using an RBF kernel). Figure (c) contains the same data as (a), but also shows the optimal decision boundaries. In (b) and (c) the dashed lines show the margins of the SVM solution.

The decision boundaries are given by the perpendicular bisectors to the faces of this K-simplex, as illustrated in Figure 1(b). This $K-1$ dimensional space will be referred to as the *simplex space* throughout. The optimal weight matrix is estimated by minimizing the misclassification errors, which are calculated by projecting the mapped object vectors on the decision boundaries in the simplex space. Predicting unknown classes of instances in the test set can be done simply by mapping the object vectors to the simplex space and finding the nearest simplex vertex. Figure 1(c) shows the decision boundaries in the original space of the two predictor attributes. It can also be seen that in both Figure 1(b) and Figure 1(c) the support vectors can be identified. The support vectors are those objects that are located on the margins or at the other side of the margin as seen from the vertex of the associated class.

The misclassification errors can be formally defined as follows. Let $\mathbf{x}_i'$ be an $m \times 1$ object vector corresponding to m attributes, and let y_i denote the class label of object i with $y_i \in \{1, \ldots, K\}$, for $i \in \{1, \ldots, n\}$. Furthermore, let $\mathbf{W}$ be an $m \times (K-1)$ dimensional weight matrix, and define a $K-1$ translation vector $\mathbf{t}$ (the bias terms). Then, object i is represented in the $K-1$ dimensional simplex space by $\mathbf{s}_i' = \mathbf{x}_i'\mathbf{W} + \mathbf{t}'$. Note that the *linear* multiclass SVM is described here, whereas nonlinearity is described in Section 5.

To obtain the misclassification error of an object, the corresponding simplex space vector is projected on each of the decision boundaries that separate the observed class of an object from the other classes. For the errors to be proportional with the distance to the decision boundaries, a regular K-simplex is used with distance 1 between each pair of vertices. Let $\mathbf{U}$ be the $K \times (K-1)$ coordinate matrix of this simplex, where a row $\mathbf{u}_k'$ of $\mathbf{U}$ gives the coordinates of a single vertex k. Then, it follows that with $k \in \{1, \ldots, K\}$ and

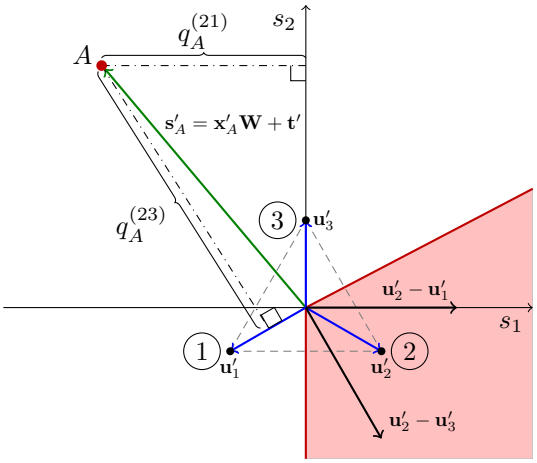


Figure 2: Graphical illustration of the distance $q_i^{(y_A j)}$ for an $y_A = 2$ and $K = 3$. The figure shows the situation in the $(K-1)$-dimensional space. The distance $q_A^{(21)}$ is calculated by projecting $\mathbf{s}'_A = \mathbf{x}'_A\mathbf{W}+\mathbf{t}'$ on $\mathbf{u}_2-\mathbf{u}_1$, and the distance $q_A^{(23)}$ is found by projecting $\mathbf{s}'_A$ on $\mathbf{u}_2 - \mathbf{u}_3$. The boundary between the class 1 and class 3 regions has been omitted for clarity, but lies along $\mathbf{u}_2$.

$l \in \{1, \ldots, K-1\}$ the elements of $\mathbf{U}$ are given by

$$u_{kl} = \begin{cases} -\frac{1}{\sqrt{2(l^2+l)}} & \text{if } k \leq l \\ \frac{l}{\sqrt{2(l^2+l)}} & \text{if } k = l+1 \\ 0 & \text{if } k > l+1. \end{cases} \tag{1}$$

Figure 2 shows an illustration of how the errors are computed. Consider object A with true class $y_A = 2$. It is clear to see that object A is misclassified as it is not located in the shaded area that has Vertex 2 as the nearest vertex. The error is computed using the perpendicular bisector of the edge connecting Vertexes 1 and 2 (here coinciding with the axis of the second dimension) and the perpendicular bisector of the edge connecting Vertexes 2 and 3, which are exactly the decision boundaries. Now, $q_A^{(21)}$ and $q_A^{(23)}$ are the distances to the respective class boundaries, which are obtained by projecting $\mathbf{s}'_A = \mathbf{x}'_A\mathbf{W}+\mathbf{t}'$ on $\mathbf{u}_2-\mathbf{u}_1$ and $\mathbf{u}_2 - \mathbf{u}_3$, respectively.

Generalizing the above, scalars $q_i^{(kj)}$ are defined to measure the distance of object i to the boundary between class k and j in the simplex space, as

$$q_i^{(kj)} = (\mathbf{x}'_i\mathbf{W} + \mathbf{t}')(\mathbf{u}_k - \mathbf{u}_j). \tag{2}$$

It is required that the GenSVM loss function is both general and flexible, such that it can easily be tuned for the specific dataset at hand. To achieve this, a loss function is constructed with a number of different weightings, each with a specific effect on the object distances $q_i^{(kj)}$.

As is customary for SVMs a hinge loss is used to ensure that instances which do not cross their class margin will yield zero error. Here, the flexible and continuous Huber hinge

loss is used (after the Huber error in robust statistics, see Huber, 1964), which is defined as

$$h(q) = \begin{cases} 1 - q - \frac{\kappa+1}{2} & \text{if } q \leq -\kappa \\ \frac{1}{2(\kappa+1)}(1-q)^2 & \text{if } q \in (-\kappa, 1] \\ 0 & \text{if } q > 1, \end{cases} \tag{3}$$

with $\kappa > -1$. The Huber hinge loss has been independently introduced in Chapelle (2007), Rosset and Zhu (2007), and Groenen et al. (2008). This hinge error is again zero when an instance is classified correctly beyond its class margin. However, in contrast to the absolute hinge error, it is continuous due to a quadratic region in the interval $(-\kappa, 1]$. This quadratic region allows for a stronger weighting of objects close to the decision boundary. Additionally, the continuity of the Huber hinge error is a desirable property for the iterative majorization algorithm derived in Section 4.1. Note that the Huber hinge error approaches the absolute hinge for $\kappa \downarrow -1$, and the quadratic hinge for $\kappa \to \infty$.

The Huber hinge error is applied to each of the distances $q_i^{(y_i j)}$, for $j \neq y_i$. Thus, no error is counted when the object is correctly classified. For each of the objects, errors with respect to the other classes are summed using an ℓ_p norm to obtain the total object error,

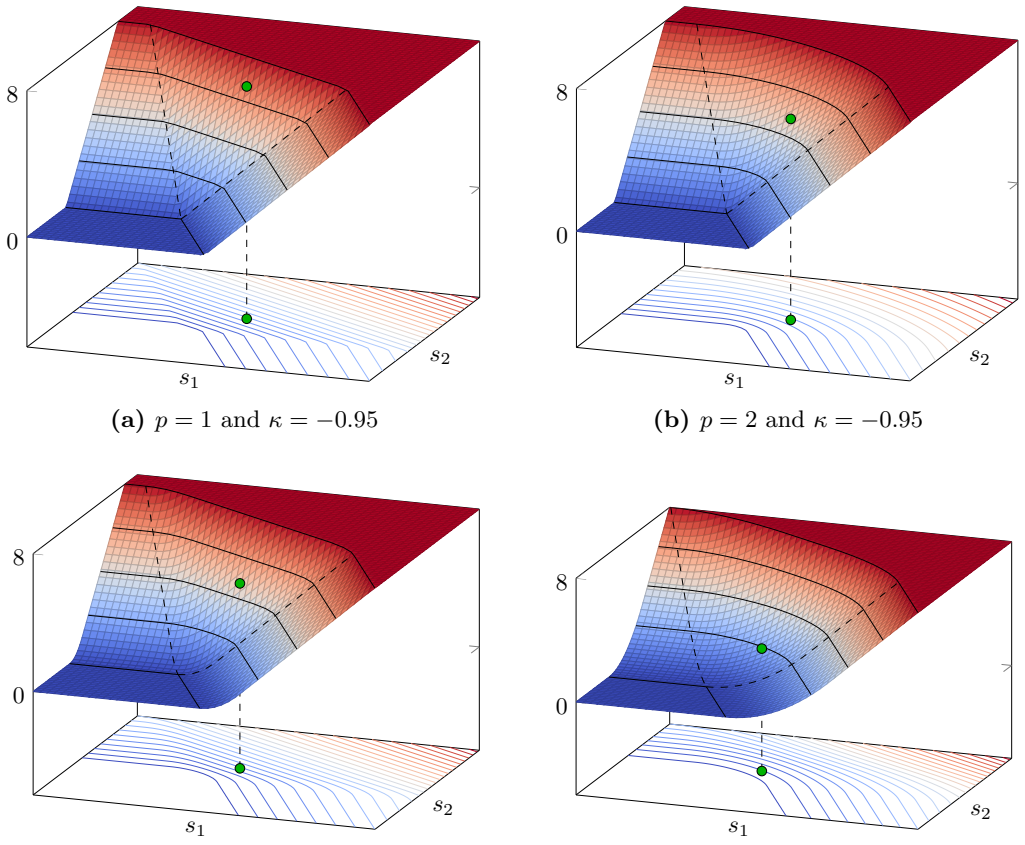
$$\left(\sum_{\substack{j=1 \\ j \neq y_i}}^{K} h^p \left(q_i^{(y_i j)} \right) \right)^{1/p} . \tag{4}$$

In the formulation of the loss function the norm parameter is restricted to the domain $p \in [1, 2]$, for algorithmic reasons. The ℓ_p norm is added to provide a form of regularization on Huber weighted errors for instances that are misclassified with respect to multiple classes.

To illustrate the effects of p and κ on the total object error, refer to Figure 3. In Figures 3(a) and 3(b), the value of p is set to $p = 1$ and $p = 2$ respectively, while maintaining the absolute hinge error using $\kappa = -0.95$. A reference point is plotted at a fixed position in the area of the simplex space where there is a nonzero error with respect to two classes. It can be seen from this reference point that the value of the combined error is higher when $p = 1$. With $p = 2$ the combined error at the reference point approximates the Euclidean distance to the margin, when $\kappa \downarrow -1$. Figures 3(a), 3(c), and 3(d) show the effect of varying κ. It can be seen that the error near the margin becomes more quadratic with increasing κ. In fact, as κ increases, the error approaches the squared Euclidean distance to the margin, which can be used to obtain a quadratic hinge multiclass SVM. Both of these effects will become stronger when the number of classes increases, as increasingly more objects will have errors with respect to more than one class.

Next, let $\rho_i \geq 0$ denote optional object weights, which are introduced to allow flexibility in the way individual objects contribute to the total loss function. With these individual weights it is possible to correct for different group sizes, or to give additional weights to misclassifications of certain classes. When correcting for group sizes, the weights can be chosen as

$$\rho_i = \frac{n}{n_k K}, \quad i \in G_k, \tag{5}$$



(a) $p = 1$ and $\kappa = -0.95$ **(b)** $p = 2$ and $\kappa = -0.95$

(c) $p = 1$ and $\kappa = 1.0$ **(d)** $p = 1$ and $\kappa = 5.0$

Figure 3: Illustration of the ℓ_p norm of the Huber weighted errors. Comparing figures (a) and (b) shows the effect of the ℓ_p norm. With $p = 1$ objects that have errors w.r.t. both classes are penalized more strongly than those with only one error, whereas with $p = 2$ this is not the case. Figures (a), (c), and (d) compare the effect of the κ parameter, with $p = 1$. This shows that with a large value of κ, the errors close to the boundary are weighted quadratically. Note that s_1 and s_2 indicate the dimensions of the simplex space.

where $G_k = \{i : y_i = k\}$ is the set of objects belonging to class k, and $n_k = |G_k|$. The complete multiclass SVM loss function combining all n objects can now be formulated as

$$L_{\text{MSVM}}(\mathbf{W}, \mathbf{t}) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in G_k} \rho_i \left(\sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right)^{1/p} + \lambda \operatorname{tr} \mathbf{W}'\mathbf{W}, \tag{6}$$

where $\lambda \operatorname{tr} \mathbf{W}'\mathbf{W}$ is the penalty term to avoid overfitting, and $\lambda > 0$ is the regularization parameter. Note that for the case where $K = 2$, the above loss function reduces to the loss function for binary SVM given in Groenen et al. (2008), with Huber hinge errors.

The outline of a proof for the convexity of the loss function in (6) is given. First, note that the distances $q_i^{(kj)}$ in the loss function are affine in $\mathbf{W}$ and $\mathbf{t}$. Hence, if the loss function is convex in $q_i^{(kj)}$ it is convex in $\mathbf{W}$ and $\mathbf{t}$ as well. Second, the Huber hinge function

is trivially convex in $q_i^{(kj)}$, since each separate piece of the function is convex, and the Huber hinge is continuous. Third, the ℓ_p norm is a convex function by the Minkowski inequality, and it is monotonically increasing by definition. Thus, it follows that the ℓ_p norm of the Huber weighted instance errors is convex (see for instance Rockafellar, 1997). Next, since it is required that the weights ρ_i are non-negative, the sum in the first term of (6) is a convex combination. Finally, the penalty term can also be shown to be convex, since tr $\mathbf{W}'\mathbf{W}$ is the square of the Frobenius norm of $\mathbf{W}$, and it is required that $\lambda > 0$. Thus, it holds that (6) is convex in $\mathbf{W}$ and $\mathbf{t}$.

Predicting class labels in GenSVM can be done as follows. Let $(\mathbf{W}^*, \mathbf{t}^*)$ denote the parameters that minimize the loss function. Predicting the class label of an unseen sample $\mathbf{x}'_{n+1}$ can then be done by first mapping it to the simplex space, using the optimal projection: $\mathbf{s}'_{n+1} = \mathbf{x}'_{n+1}\mathbf{W}^* + \mathbf{t}^*$. The predicted class label is then simply the label corresponding to the nearest simplex vertex as measured by the squared Euclidean norm, or

$$\hat{y}_{n+1} = \arg\min_k \|\mathbf{s}'_{n+1} - \mathbf{u}'_k\|^2, \quad \text{for } k = 1, \ldots, K. \tag{7}$$

3. Iterative Majorization

To minimize the loss function given in (6), an iterative majorization algorithm will be derived. Iterative majorization was first described by Weiszfeld (1937), however the first application of the algorithm in the context of a line search comes from Ortega and Rheinboldt (1970, p. 253 – 255). During the late 1970s, the method was independently developed by De Leeuw (1977) as part of the SMACOF algorithm for multidimensional scaling, and by Voss and Eckhardt (1980) as a general minimization method. It is the description given in the latter that will be presented here[2].

Given a continuous function $f : \mathcal{X} \to \mathbb{R}$ with $\mathcal{X} \subseteq \mathbb{R}^d$, construct a *majorization function* $g(x, \overline{x})$ such that

$$f(\overline{x}) = g(\overline{x}, \overline{x}), \tag{8}$$

$$f(x) \leq g(x, \overline{x}) \text{ for all } x \in \mathcal{X}, \tag{9}$$

with $\overline{x} \in \mathcal{X}$ a so-called *supporting point*. In general, the majorization function is constructed such that its minimum can easily be found, for instance by choosing it to be quadratic in x. If $f(x)$ is differentiable at the supporting point, the above conditions imply $\nabla f(\overline{x}) = \nabla g(\overline{x}, \overline{x})$. The following procedure can now be used to find a stationary point of $f(x)$,

1. Let $\overline{x} = x_0$, with x_0 a random starting point.
2. Minimize $g(x, \overline{x})$ with respect to x, such that $x^+ = \arg\min g(x, \overline{x})$.
3. If $f(\overline{x}) - f(x^+) < \epsilon f(x^+)$ stop, otherwise let $\overline{x} = x^+$ and go to step 2.

[2]The IM algorithm is also known as the MM algorithm, where MM is short for minimization by majorization, or maximization by minorization. See, for example, Hunter and Lange (2004).

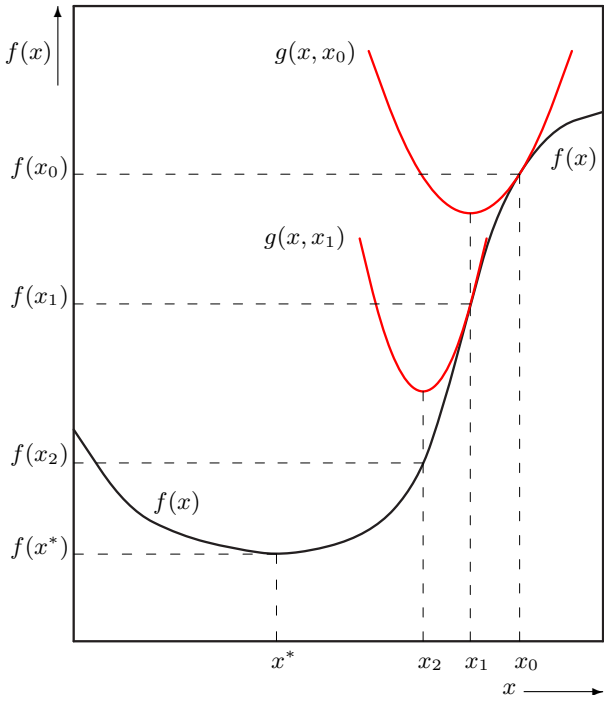


Figure 4: One-dimensional graphical illustration of the iterative majorization algorithm, adapted from De Leeuw (1988). The minimum of a majorization function $g(x, x_r)$ provides the supporting point for the next majorization function $g(x, x_{r+1})$. The sequence of supporting points $\{x_r\}$ converges towards the stationary point x^* if $f(x)$ is bounded from below, as is the case here.

In this algorithm ϵ is a small constant. Note that $f(x)$ must be bounded from below on $\mathcal{X}$ for the algorithm to converge. In fact, the following *sandwich inequality* can be derived (De Leeuw, 1993),

$$f(x^+) \leq g(x^+, \overline{x}) \leq g(\overline{x}, \overline{x}) = f(\overline{x}). \tag{10}$$

This inequality shows that if $f(x)$ is bounded from below the iterative majorization algorithm achieves global convergence to a stationary point of the function (Voss and Eckhardt, 1980). The iterative majorization algorithm is illustrated in Figure 4, where the majorization functions are shown as a quadratic function. As can be seen from the illustration, the sequence of supporting points $\{x_r\}$ converges to the stationary point x^* of the function $f(x)$. In practical situations, this convergence is to a local minimum of $f(x)$.

The asymptotic convergence rate of the IM algorithm is linear, and is thus smaller than that of the Newton-Raphson algorithm (De Leeuw, 1994). However, the largest improvements in the loss function will occur in the first few steps of the iterative majorization algorithm, where the asymptotic linear rate does not apply (Havel, 1991).

For quadratic majorization the number of iterations can often be reduced by using a technique known as *step doubling* (De Leeuw and Heiser, 1980). Step doubling reduces the number of iterations by using $\overline{x} = x_{r+1} = 2x^+ - x_r$ as the next supporting point in Step 3

of the algorithm, instead of $\overline{x} = x_{r+1} = x^+$. Intuitively, step doubling can be understood as stepping over the minimum of the majorization function to the point lying directly "opposite" the supporting point $\overline{x}$ (see also Figure 4). Note that the guaranteed descent of the IM algorithm still holds when using step doubling, since $f(2x^+ - \overline{x}) \leq g(2x^+ - \overline{x}, \overline{x}) = g(\overline{x}, \overline{x}) = f(\overline{x})$. In practice, step doubling reduces the number of iterations by half. A caveat of using step doubling is that the distance to the stationary point can be increased if the initial point is far from this point. Therefore, in practical applications, a burn-in should be used before step doubling is applied.

There is no straightforward technique for deriving the majorization function for any given convex function. However, in the next section the derivation of the majorization function for the GenSVM loss function is presented, using an "outside-in" approach. In this approach each function that constitutes the loss function is majorized separately and the majorization functions are combined. Two properties of majorization functions which are useful for this derivation are now formally defined.

P1. Let $f_1 : \mathcal{Y} \to \mathcal{Z}$, $f_2 : \mathcal{X} \to \mathcal{Y}$, and define $f = f_1 \circ f_2 : \mathcal{X} \to \mathcal{Z}$, such that for $x \in \mathcal{X}$, $f(x) = f_1(f_2(x))$. If $g_1 : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Z}$ is a majorization function of f_1, then $g : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$ defined as $g = g_1 \circ f_2$ is a majorization function of f. Thus for $x, \overline{x} \in \mathcal{X}$ it holds that $g(x, \overline{x}) = g_1(f_2(x), f_2(\overline{x}))$ is a majorization function of $f(x)$ at $\overline{x}$.

P2. Let $f_i : \mathcal{X} \to \mathcal{Z}$ and define $f : \mathcal{X} \to \mathcal{Z}$ such that $f(x) = \sum_i a_i f_i(x)$ for $x \in \mathcal{X}$, with $a_i \geq 0$ for all i. If $g_i : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$ is a majorization function for f_i at a point $\overline{x} \in \mathcal{X}$, then $g : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$ given by $g(x, \overline{x}) = \sum_i a_i g_i(x, \overline{x})$ is a majorization function of f.

Proofs of these properties are omitted, as they follow directly from the requirements for a majorization function in (8), and (9). The first property allows for the use of the "outside-in" approach to majorization, as will be illustrated in the next section.

4. GenSVM Optimization and Implementation

In this section, the IM algorithm for GenSVM will be derived using the "outside-in" strategy of majorization. Pseudocode for the derived algorithm will be presented, as well as an analysis of the computational complexity of the algorithm. A note on smart initialization of the algorithm is also given.

4.1. Majorization Derivation

To shorten the notation, define

$$\mathbf{V} = [\mathbf{t}\ \mathbf{W}']', \tag{11}$$

$$\mathbf{z}_i' = [1\ \mathbf{x}_i'], \tag{12}$$

$$\boldsymbol{\delta}_{kj} = \mathbf{u}_k - \mathbf{u}_j, \tag{13}$$

such that $q_i^{(kj)} = \mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}$. With this notation it becomes sufficient to optimize the loss function with respect to $\mathbf{V}$. Formulated in this manner (6) becomes

$$L_{\text{MSVM}}(\mathbf{V}) = \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k}\rho_i\left(\sum_{j\neq k}h^p\left(q_i^{(kj)}\right)\right)^{1/p} + \lambda\,\text{tr}\;\mathbf{V}'\mathbf{J}\mathbf{V}, \tag{14}$$

where $\mathbf{J}$ is an $m+1$ diagonal matrix with $\mathbf{J}_{i,i} = 1$ for $i > 1$ and zero elsewhere. To derive a majorization function for this expression the "outside-in" approach will be used, together with the properties of majorization functions. In what follows, variables with a bar denote supporting points for the IM algorithm. The goal of the derivation is to find a quadratic majorization function in $\mathbf{V}$ such that

$$L_{\text{MSVM}}(\mathbf{V}) \leq \text{tr}\;\mathbf{V}'\mathbf{Z}'\mathbf{A}\mathbf{Z}'\mathbf{V} - 2\,\text{tr}\;\mathbf{V}'\mathbf{Z}'\mathbf{B} + C, \tag{15}$$

where $\mathbf{A}$, $\mathbf{B}$, and C are coefficients of the majorization depending on $\overline{\mathbf{V}}$. The matrix $\mathbf{Z}$ is simply the $n \times (m+1)$ matrix with rows $\mathbf{z}_i'$.

Property P2 above means that the summation over instances in the loss function can be ignored. In addition, the regularization term is quadratic in $\mathbf{V}$, and thus requires no majorization. The outermost function for which a majorization function has to be found is thus the ℓ_p norm of the Huber hinge errors. Hence it is possible to consider the function $f(\mathbf{x}) = \|\mathbf{x}\|_p$ for majorization. A majorization function for $f(\mathbf{x})$ can be constructed, but a discontinuity at $\mathbf{x} = \mathbf{0}$ will remain (Tsutsu and Morikawa, 2012).

To avoid the discontinuity of the ℓ_p norm, the following inequality is needed (Hardy et al., 1934, eq. 2.10.3)

$$\left(\sum_{j\neq k}h^p\left(q_i^{(kj)}\right)\right)^{1/p} \leq \sum_{j\neq k}h\left(q_i^{(kj)}\right). \tag{16}$$

This inequality can be used as a majorization function only if equality holds at the supporting point,

$$\left(\sum_{j\neq k}h^p\left(\overline{q}_i^{(kj)}\right)\right)^{1/p} = \sum_{j\neq k}h\left(\overline{q}_i^{(kj)}\right). \tag{17}$$

It is not difficult to see that this only holds if at most one of the $h\left(\overline{q}_i^{(kj)}\right)$ errors is nonzero for $j \neq k$. Thus an indicator variable ε_i is introduced which is 1 if at most one of these errors is nonzero, and 0 otherwise. Then it follows that

$$\begin{aligned} L_{\text{MSVM}}(\mathbf{V}) \leq & \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k}\rho_i\left[\varepsilon_i\sum_{j\neq k}h\left(q_i^{(kj)}\right) + (1-\varepsilon_i)\left(\sum_{j\neq k}h^p\left(q_i^{(kj)}\right)\right)^{1/p}\right] \\ & + \lambda\,\text{tr}\;\mathbf{V}'\mathbf{J}\mathbf{V}. \end{aligned} \tag{18}$$

Now, the next function for which a majorization needs to be found is $f_1(x) = x^{1/p}$. From the inequality $a^\alpha b^\beta < \alpha a + \beta b$, with $\alpha + \beta = 1$ (Hardy et al., 1934, Theorem 37), a

linear majorization inequality can be constructed for this function by substituting $a = x$, $b = \overline{x}$, $\alpha = 1/p$ and $\beta = 1 - 1/p$ (Groenen and Heiser, 1996). This yields

$$f_1(x) = x^{1/p} \leq \frac{1}{p}\overline{x}^{1/p-1}x + \left(1 - \frac{1}{p}\right)\overline{x}^{1/p} = g_1(x, \overline{x}). \tag{19}$$

Applying this majorization and using property P1 gives

$$\begin{aligned}\left(\sum_{j\neq k} h^p\left(q_i^{(kj)}\right)\right)^{1/p} \leq \frac{1}{p}\left(\sum_{j\neq k} h^p\left(\overline{q}_i^{(kj)}\right)\right)^{1/p-1}\left(\sum_{j\neq k} h^p\left(q_i^{(kj)}\right)\right) \\ + \left(1 - \frac{1}{p}\right)\left(\sum_{j\neq k} h^p\left(\overline{q}_i^{(kj)}\right)\right)^{1/p}.\end{aligned} \tag{20}$$

Plugging this into (18) and collecting terms yields,

$$\begin{aligned}L_{\text{MSVM}}(\mathbf{V}) \leq \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k}\rho_i\left[\varepsilon_i\sum_{j\neq k} h\left(q_i^{(kj)}\right) + (1-\varepsilon_i)\omega_i\sum_{j\neq k} h^p\left(q_i^{(kj)}\right)\right] \\ + \Gamma^{(1)} + \lambda\,\text{tr}\;\mathbf{V}'\mathbf{J}\mathbf{V},\end{aligned} \tag{21}$$

with

$$\omega_i = \frac{1}{p}\left(\sum_{j\neq k} h^p\left(\overline{q}_i^{(kj)}\right)\right)^{1/p-1}. \tag{22}$$

The constant $\Gamma^{(1)}$ contains all terms that only depend on previous errors $\overline{q}_i^{(kj)}$. The next majorization step by the "outside-in" approach is to find a quadratic majorization function for $f_2(x) = h^p(x)$, of the form

$$f_2(x) = h^p(x) \leq a(\overline{x}, p)x^2 - 2b(\overline{x}, p)x + c(\overline{x}, p) = g_2(x, \overline{x}). \tag{23}$$

Since this derivation is mostly an algebraic exercise it has been moved to Appendix A. In the remainder of this derivation $a_{ijk}^{(p)}$ will be used to abbreviate $a(\overline{q}_i^{(kj)}, p)$, with similar abbreviations for b and c. Using these majorizations and making the dependence on $\mathbf{V}$ explicit by substituting $q_i^{(kj)} = \mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}$ gives

$$\begin{aligned}L_{\text{MSVM}}(\mathbf{V}) \leq \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k}\rho_i\varepsilon_i\sum_{j\neq k}\left[a_{ijk}^{(1)}\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}'\mathbf{V}'\mathbf{z}_i - 2b_{ijk}^{(1)}\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\right] \\ + \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k}\rho_i(1-\varepsilon_i)\omega_i\sum_{j\neq k}\left[a_{ijk}^{(p)}\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}'\mathbf{V}'\mathbf{z}_i - 2b_{ijk}^{(p)}\mathbf{z}_i'\mathbf{V}\boldsymbol{\delta}_{kj}\right] \\ + \Gamma^{(2)} + \lambda\,\text{tr}\;\mathbf{V}'\mathbf{J}\mathbf{V},\end{aligned} \tag{24}$$

where $\Gamma^{(2)}$ again contains all constant terms. Due to dependence on the matrix $\boldsymbol{\delta}_{kj}\boldsymbol{\delta}_{kj}'$, the above majorization function is not yet in the desired quadratic form of (15). However,

since the maximum eigenvalue of $\boldsymbol{\delta}_{kj}\boldsymbol{\delta}'_{kj}$ is 1 by definition of the simplex coordinates, it follows that the matrix $\boldsymbol{\delta}_{kj}\boldsymbol{\delta}'_{kj} - \mathbf{I}$ is negative semidefinite. Hence, it can be shown that the inequality $\mathbf{z}'_i(\mathbf{V} - \overline{\mathbf{V}})(\boldsymbol{\delta}_{kj}\boldsymbol{\delta}'_{kj} - \mathbf{I})(\mathbf{V} - \overline{\mathbf{V}})'\mathbf{z}_i \leq 0$ holds (Bijleveld and De Leeuw, 1991, Theorem 4). Rewriting this gives the majorization inequality

$$\mathbf{z}'_i\mathbf{V}\boldsymbol{\delta}_{kj}\boldsymbol{\delta}'_{kj}\mathbf{V}'\mathbf{z}_i \leq \mathbf{z}'_i\mathbf{V}\mathbf{V}'\mathbf{z}_i - 2\mathbf{z}'_i\mathbf{V}(\mathbf{I} - \boldsymbol{\delta}_{kj}\boldsymbol{\delta}'_{kj})\overline{\mathbf{V}}\mathbf{z}_i + \mathbf{z}'_i\overline{\mathbf{V}}(\mathbf{I} - \boldsymbol{\delta}_{kj}\boldsymbol{\delta}'_{kj})\overline{\mathbf{V}}'\mathbf{z}_i. \tag{25}$$

With this inequality the majorization inequality becomes

$$\begin{aligned} L_{\text{MSVM}}(\mathbf{V}) \leq \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in G_k}\rho_i\mathbf{z}'_i\mathbf{V}(\mathbf{V}' - 2\overline{\mathbf{V}}')\mathbf{z}_i\sum_{j\neq k}\left[\varepsilon_i a_{ijk}^{(p)} + (1-\varepsilon_i)\omega_i a_{ijk}^{(p)}\right] \\ - \frac{2}{n}\sum_{k=1}^{K}\sum_{i\in G_k}\rho_i\mathbf{z}'_i\mathbf{V}\sum_{j\neq k}\Big[\varepsilon_i\left(b_{ijk}^{(1)} - a_{ijk}^{(1)}\overline{q}_i^{(kj)}\right) \\ +(1-\varepsilon_i)\omega_i\left(b_{ijk}^{(p)} - a_{ijk}^{(p)}\overline{q}_i^{(kj)}\right)\Big]\boldsymbol{\delta}_{kj} \\ + \Gamma^{(3)} + \lambda\,\text{tr}\;\mathbf{V}'\mathbf{J}\mathbf{V}, \end{aligned} \tag{26}$$

where $\overline{q}_i^{(kj)} = \mathbf{z}'_i\overline{\mathbf{V}}\boldsymbol{\delta}_{kj}$. This majorization function is quadratic in $\mathbf{V}$ and can thus be used in the IM algorithm. To derive the first-order condition used in Step 2 of the IM algorithm matrix notation for the above expression is introduced. Let $\mathbf{A}$ be an $n \times n$ diagonal matrix with elements α_i, and let $\mathbf{B}$ be an $n \times (K-1)$ matrix with rows $\boldsymbol{\beta}'_i$, where

$$\alpha_i = \frac{1}{n}\rho_i\sum_{j\neq k}\left[\varepsilon_i a_{ijk}^{(p)} + (1-\varepsilon_i)\omega_i a_{ijk}^{(p)}\right], \tag{27}$$

$$\boldsymbol{\beta}'_i = \frac{1}{n}\rho_i\sum_{j\neq k}\left[\varepsilon_i\left(b_{ijk}^{(1)} - a_{ijk}^{(1)}\overline{q}_i^{(kj)}\right) + (1-\varepsilon_i)\omega_i\left(b_{ijk}^{(p)} - a_{ijk}^{(p)}\overline{q}_i^{(kj)}\right)\right]\boldsymbol{\delta}'_{kj}. \tag{28}$$

Then the majorization function of $L_{\text{MSVM}}(\mathbf{V})$ given in (26) can be written as

$$L_{\text{MSVM}}(\mathbf{V}) \leq \text{tr}\;(\mathbf{V} - 2\overline{\mathbf{V}})'\mathbf{Z}'\mathbf{A}\mathbf{Z}\mathbf{V} - 2\,\text{tr}\;\mathbf{B}'\mathbf{Z}\mathbf{V} + \Gamma^{(3)} + \lambda\,\text{tr}\;\mathbf{V}'\mathbf{J}\mathbf{V} \tag{29}$$

$$= \text{tr}\;\mathbf{V}'(\mathbf{Z}'\mathbf{A}\mathbf{Z} + \lambda\mathbf{J})\mathbf{V} - 2\,\text{tr}\;(\overline{\mathbf{V}}'\mathbf{Z}'\mathbf{A} + \mathbf{B}')\mathbf{Z}\mathbf{V} + \Gamma^{(3)}. \tag{30}$$

This majorization function has the desired functional form described in (15). Differentiation with respect to $\mathbf{V}$ and equating to zero yields the linear system

$$(\mathbf{Z}'\mathbf{A}\mathbf{Z} + \lambda\mathbf{J})\mathbf{V} = (\mathbf{Z}'\mathbf{A}\mathbf{Z}\overline{\mathbf{V}} + \mathbf{Z}'\mathbf{B}). \tag{31}$$

The update $\mathbf{V}^+$ which solves this system can then be calculated efficiently by Gaussian elimination.

4.2. Algorithm Implementation and Complexity

Pseudocode for GenSVM is given in Algorithm 1. As can be seen, the algorithm simply updates all instance weights at each iteration, starting by determining the indicator variable ε_i. In practice, some calculations can be done efficiently for all instances by using matrix

Algorithm 1: GenSVM Algorithm

Input: $\mathbf{X}, \mathbf{y}, \boldsymbol{\rho}, p, \kappa, \lambda, \epsilon$
Output: $\mathbf{V}^*$

$K \leftarrow \max(\mathbf{y})$
$t \leftarrow 1$
$\mathbf{Z} \leftarrow [\mathbf{1}\ \mathbf{X}]$
Let $\overline{\mathbf{V}} \leftarrow \mathbf{V}_0$
Generate $\mathbf{J}$ and $\mathbf{U}$
$L_t = L_{\text{MSVM}}(\overline{\mathbf{V}})$
$L_{t-1} = (1 + 2\epsilon)L_t$
while $(L_{t-1} - L_t)/L_t > \epsilon$ **do**
    for $i \leftarrow 1$ **to** n **do**
        Compute $\overline{q}_i^{(y_i j)} = \mathbf{z}_i' \overline{\mathbf{V}} \boldsymbol{\delta}_{y_i j}$ for all $j \neq y_i$
        Compute $h\left(\overline{q}_i^{(y_i j)}\right)$ for all $j \neq y_i$ by (3)
        if $\varepsilon_i = 1$ **then**
            Compute $a_{ijy_i}^{(1)}$ for all $j \neq y_i$ following Appendix A
            Compute $b_{ijy_i}^{(1)}$ for all $j \neq y_i$ following Appendix A
        else
            Compute ω_i following (22)
            Compute $a_{ijy_i}^{(p)}$ for all $j \neq y_i$ following Appendix A
            Compute $b_{ijy_i}^{(p)}$ for all $j \neq y_i$ following Appendix A
        end
        Compute α_i by (27)
        Compute $\boldsymbol{\beta}_i$ by (28)
    end
    Construct $\mathbf{A}$ from α_i
    Construct $\mathbf{B}$ from $\boldsymbol{\beta}_i$
    Find $\mathbf{V}^+$ that solves (31)
    $\overline{\mathbf{V}} \leftarrow \mathbf{V}$
    $\mathbf{V} \leftarrow \mathbf{V}^+$
    $L_{t-1} \leftarrow L_t$
    $L_t \leftarrow L_{\text{MSVM}}(\mathbf{V})$
    $t \leftarrow t + 1$
end

algebra. When step doubling is applied in the majorization algorithm, line 27 is replaced by $\mathbf{V} \leftarrow 2\mathbf{V}^+ - \overline{\mathbf{V}}$. In the implementation step doubling is applied after a burn-in of 50 iterations. The implementation used in the experiments described in Section 6 is written in `C`, using the BLAS and LAPACK libraries. The source code for this `C` library is available under the open source GNU GPL license, through an online repository. A thorough description of the implementation is available in the package documentation.

The complexity of a single iteration of the IM algorithm is $O(n(m+1)^2)$ assuming that $n > m > K$. As noted earlier, the convergence rate of the general IM algorithm is linear.

4.3. Smart Initialization

When training machine learning algorithms to determine the optimal hyperparameters, it is common to use cross validation (CV). First, the data is split randomly in 10 parts of roughly equal size. Then, each of these parts is used once as a test set, where the remaining parts are combined in a training set. The algorithm is then trained on the training set and tested on the test set, which is repeated 10 times for each fold. With GenSVM it is possible to initialize the matrix $\overline{\mathbf{V}}$ such that the final result of a fold is used as the initial value for $\mathbf{V}_0$ for the next fold. Such warm-start initialization greatly reduces the time needed to perform cross validation with GenSVM. It is important to note here that this smart initialization is not easily possible with dual optimization approaches. Therefore, the ability to use smart initialization can be seen as an advantage of solving the GenSVM optimization problem in the primal.

5. Nonlinearity

One possible method to include nonlinearity in a classifier is through the use of spline transformations (see e.g. Hastie et al., 2009). With spline transformations each attribute vector $\mathbf{x}_j$ is transformed to a spline basis $\mathbf{N}_j$, for $j = 1, \ldots, m$. The transformed input matrix $\mathbf{N} = [\mathbf{N}_1, \ldots, \mathbf{N}_m]$ is then of size $n \times l$, where l depends on the degree of the spline transformation and the number of interior knots chosen. An application of spline transformations to the binary SVM can be found in Groenen et al. (2007).

A more common way to include nonlinearity in machine learning methods is through the use of the kernel trick, attributed to Aizerman et al. (1964). With the kernel trick, the dot product of two instance vectors in the dual optimization problem is replaced by the dot product of the same vectors in a high dimensional feature space. Since no dot products appear in the primal formulation of GenSVM, a different method is used here. By applying a pre-processing step on the kernel matrix, nonlinearity can be included using the same algorithm as the one presented for the linear case. Furthermore, predicting class labels requires a post-processing step on the obtained matrix $\mathbf{V}^*$. A full derivation is given in Appendix B.

6. Experiments

To assess the performance of the proposed GenSVM classifier, comparisons were done with seven existing multiclass SVMs on 13 datasets. In addition to predictive performance, the total training time of each classifier is compared. These quantities are compared using both performance profiles and rank tests.

6.1. Setup

Implementations of the heuristic multiclass SVMs (OvO, OvA, and DAGSVM) were done through LibSVM (v. 3.16, Chang and Lin, 2011). LibSVM is a `C++` library for binary SVMs which implements the SMO algorithm of Platt (1999). The OvO and DAGSVM methods are implemented in this package, and a `C` implementation of OvA using LibSVM was created

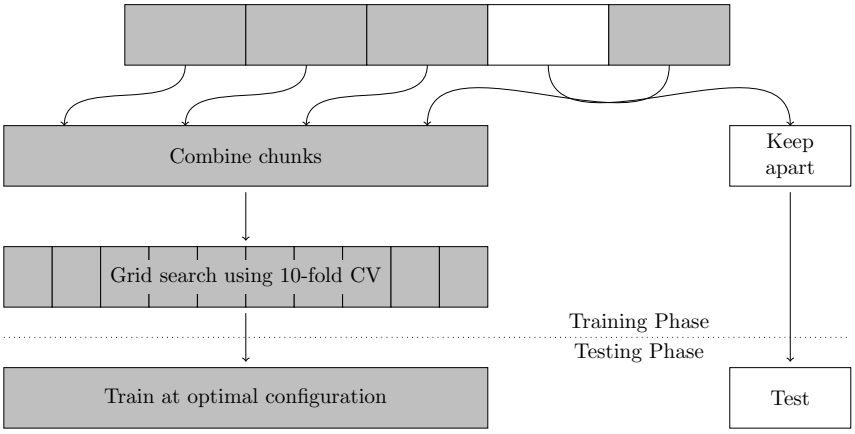


Figure 5: An illustration of nested cross validation. A dataset is initially split in five *chunks*. Each chunk is kept apart once, while a grid search using 10-fold CV is applied to the combined data from the remaining 4 chunks. The optimal parameters obtained there are then used to train the model one last time, and predict the chunk that was kept apart.

for these experiments[3]. For the single-machine approaches the MSVMpack package was used (v. 1.3, Lauer and Guermeur, 2011), which is written in `C`. This package implements the methods of Weston and Watkins (W&W, 1998), Crammer and Singer (C&S, 2002a), Lee et al. (LLW, 2004), and Guermeur and Monfrini (MSVM^2, 2011).

To compare the classification methods properly, it is desirable to remove any bias that could occur when using cross validation. Therefore, a process which is known as nested cross validation is used, as illustrated in Figure 5. In nested CV, a dataset is randomly split in a number of *chunks* (or superfolds). Each of these chunks is kept apart from the remaining chunks once, while the remaining chunks are combined to form a single dataset. A grid search is then applied to this dataset to find the optimal hyperparameters with which to predict the test chunk. This process is then repeated for each of the chunks. The predictions of the test chunk will be unbiased since it was not included in the grid search. For this reason, it is argued that this approach is to be preferred over approaches which simply report maximum accuracy rates obtained during the grid search.

For the experiments 13 datasets were selected from the UCI repository (Bache and Lichman, 2013). The selected datasets and their relevant statistics are shown in Table 1. All attributes were rescaled to the interval $[-1, 1]$. The `image segmentation` and `vowel` datasets have a predetermined train and test set, and were thus not used in the nested CV procedure. Instead, a grid search was done on the provided training set for each classifier, and the provided test set was predicted at the optimal hyperparameters obtained. For the datasets without a predetermined train/test split, nested CV was used with 5 initial chunks. Hence, $5 \cdot 11 + 2 = 57$ pairs of independent train and test datasets are obtained.

While running the grid search, it is desirable to remove any fluctuations that may result in an unfair comparison. Therefore, it was ensured that all methods had the same CV split

[3]The LibSVM code used for DAGSVM is the same code as was used in Hsu and Lin (2002) and is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools`.

Table 1: Dataset summary statistics. Datasets with an asterisk have a predetermined test dataset. For these datasets, the number of training instances is denoted for the train and test datasets respectively.

Dataset	**Training instances** (n)	**Attributes** (m)	**Classes** (K)	**Smallest Class**	**Largest Class**
breast tissue	106	9	6	14	22
iris	150	4	3	50	50
wine	178	13	3	48	71
image segmentation*	210/2100	18	7	30	30
glass	214	9	6	9	76
vertebral	310	6	3	60	150
ecoli	336	8	8	2	143
vowel*	528/462	10	11	48	48
balancescale	625	4	3	49	288
vehicle	846	18	4	199	218
contraception	1473	9	3	333	629
yeast	1484	8	10	5	463
car	1728	6	4	65	1210

of the training data for the same hyperparameter configuration (specifically, the value of the regularization parameter). In practice, it can occur that a specific CV split is advantageous for one classifier but not for others (either in time or performance). Thus, ideally the grid search would be repeated a number of times with different CV splits, to remove this variation. However, due to the size of the grid search this is considered to be not feasible. Finally, it should be noted here that during the grid search 10-fold cross validation was applied in a non-stratified manner, that is without resampling of small classes.

Due to the large number of datasets and methods, training was done only for the linear kernel. The regularization parameter was varied on a grid with $C, \lambda \in \{2^{-18}, 2^{-16}, \ldots, 2^{18}\}$. For GenSVM the grid search was extended with the parameters $\kappa \in \{-0.9, 0.5, 5.0\}$ and $p \in \{1.0, 1.5, 2.0\}$. The stopping parameter for the GenSVM majorization algorithm was set at $\epsilon = 10^{-6}$. In addition, two different weight specifications were used for GenSVM, the unit weights with $\rho_i = 1, \forall i$, as well as the group-size correction weights introduced in (5). Thus, the grid search consists of 342 configurations for GenSVM, and 19 configurations for the other methods. Since nested CV is used for most datasets, it is required to run 10-fold cross validation on a total of 27075 hyperparameter configurations. To enhance the reproducibility of these experiments, the exact predictions made by each classifier for each configuration were stored in a text file.

To run all computations in a reasonable amount of time, the computations were performed on the Dutch National LISA Compute Cluster. A master-worker program was developed using the message passing interface in Python. This allows for efficient use of multiple nodes by successively sending out tasks to worker threads from a single master thread. Since the total training time of a classifier is also of interest, it was ensured that

all computations were done on the exact same core type[4]. Furthermore, training time was measured from within the `C` programs, to ensure that only the time needed for the cross validation routine was measured. The total computation time needed to obtain the presented results was about 228 days, using the LISA Cluster this was done in five and a half days wall-clock time.

During the training phase it showed that several of the single machine methods implemented through MSVMpack could not converge to an optimal solution within reasonable amount of time[5]. Instead of limiting the maximum number of iterations of the method, MSVMpack was modified to stop after a maximum of 2 hours of training time per configuration. This results in 12 minutes of training time per cross validation fold. The solution found after this amount of training time was used for prediction during cross validation. Whenever training was stopped prematurely, this was recorded[6]. Of the 57 training sets, 24 optimal configurations had prematurely stopped training in one or more CV splits for the LLW method, versus 19 for W&W, 9 for MSVM^2, and 2 for C&S. For the LibSVM methods, 13 optimal configurations for OvA reached the default maximum number of iterations in one or more CV folds, versus 9 for DAGSVM, and 3 for OvO.

Determining the optimal hyperparameters requires a performance measure on the obtained predictions. For binary classifiers it is common to use either the hitrate or the area under the ROC curve as a measure of classifier performance. The hitrate measures the percentage of correct predictions of a classifier, and has the well known problem that no correction is made for group sizes. For instance, if 90% of the observations of a test set belong to one class, a classifier that always predicts this class has a high hitrate, regardless of its discriminatory power. Therefore, the adjusted rand index (ARI) is used as a performance measure (Hubert and Arabie, 1985). The ARI corrects for chance and can therefore more accurately measure discriminatory power of a classifier than the hitrate can. Using the ARI for evaluating supervised learning algorithms has previously been proposed by Santos and Embrechts (2009).

The optimal parameter configurations for each method on each dataset were chosen such that the maximum predictive performance was obtained as measured with the ARI. If multiple configurations obtained the highest performance during the grid search, the configuration with the smallest training time was chosen. The results on the training data show that during cross validation GenSVM achieved the highest classification accuracy on 39 out of 57 datasets, compared to 16 and 13 for DAGSVM and OvO, respectively. However, these are results on the training datasets and therefore can contain considerable bias. To accurately assess the out-of-sample prediction accuracy the optimal hyperparameter configurations were determined for each of the 57 training sets, and the test sets were predicted with these parameters. This was done through a newly created software library named MSVMEval, which combines LibSVM, MSVMpack, and GenSVM in a single pack-

[4]The specific type of core used is the Intel Xeon E5-2650 v2, with 16 threads at a clock speed of 2.6 GHz. At most 14 threads were used simultaneously, reserving one master thread and one for system processes.

[5]Using the default MSVMpack settings and a chunk size of 4 for all methods.

[6]For the classifiers implemented through LibSVM very long training times were only observed for the OvA method, however due to the nature of this method it is not trivial to stop the calculations after a certain amount of time. This behaviour was observed in about 1% of all configurations tested on all datasets, and is therefore considered negligible. Also, for the LibSVM methods it was recorded whenever the maximum number of iterations was reached.

age. MSVMEval allows a researcher to easily compare the 8 multiclass SVMs on a given dataset. To remove any variations due to random starts, training and predicting the test set was repeated 5 times for each classifier.

To promote reproducibility of the empirical results, all the code used for the classifier comparisons and all the obtained results will be released through an online repository.

6.2. Performance Profiles

One way to get insight in the performance of different classification methods is through *performance profiles* (Dolan and Moré, 2002). A performance profile shows the empirical cumulative distribution function of a classifier on a performance metric.

Let $\mathcal{D}$ denote the set of datasets, and $\mathcal{C}$ denote the set of classifiers. Further, let $p_{d,c}$ denote the performance of classifier $c \in \mathcal{C}$ on dataset $d \in \mathcal{D}$ as measured by the ARI. Now define the performance ratio $v_{d,c}$ as the ratio between the best performance on dataset d and the performance of classifier c on dataset d, that is

$$v_{d,c} = \frac{\max\{p_{d,c} : c \in \mathcal{C}\}}{p_{d,c}}. \tag{32}$$

Thus the performance ratio is 1 for the best performing classifier on a dataset and increases for classifiers with a lower performance. Then, the performance profile for classifier c is given by the function

$$P_c(\eta) = \frac{1}{N_D} \left|\{d \in \mathcal{D} : v_{d,c} \leq \eta\}\right|, \tag{33}$$

where $N_D = |\mathcal{D}|$ denotes the number of datasets. Thus, the performance profile estimates the probability that classifier c has a performance ratio below η. Note that $P_c(1)$ denotes the empirical probability that a classifier achieves the highest performance on a given dataset.

Figure 6 shows the performance profile for classification accuracy. Estimates of $P_c(1)$ from Figure 6 show that there is a 29.47% probability that GenSVM achieves the optimal performance, versus 28.42% for OvO and 27.72% for the DAGSVM method. Figure 6 also shows that if one is merely concerned with being within a factor of 1.13 of the optimal classification, there is little difference between these three classifiers. Finally, it shows that OvA and the methods of Lee et al. (2004), Crammer and Singer (2002a), and Guermeur and Monfrini (2011) always have a smaller probability of being within a given factor of the optimal performance than any of the other methods.

Similarly, a performance profile can be constructed for the training time necessary to do the grid search. Let $t_{d,c}$ denote the total training time for classifier c on dataset d. Next, define the performance ratio for time as

$$w_{d,c} = \frac{t_{d,c}}{\min\{t_{d,c} : c \in \mathcal{C}\}}. \tag{34}$$

Note that here the classifier with the smallest training time has preference. Therefore, comparison of classifier computation time is done with the lowest computation time achieved on a given dataset d. Again, the ratio is 1 when the lowest training time is reached, and it increases for higher computation time. Thus, the performance profile for time is defined as

$$T_c(\tau) = \frac{1}{N_D} |\{d \in \mathcal{D} : w_{d,c} \leq \tau\}|. \tag{35}$$

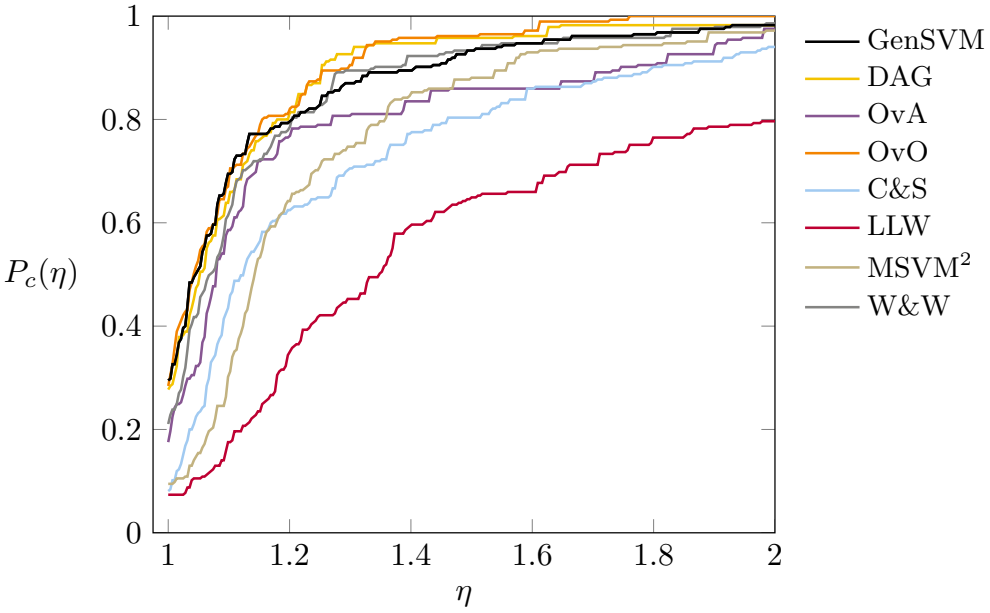


Figure 6: Performance profiles for classification accuracy created from all runs in MSVMEval. GenSVM has the highest probability of achieving the maximum classification accuracy on a given dataset. The methods OvA, C&S, MSVM^2, and LLW will always have a smaller probability of achieving maximum performance than the other methods.

Thus the performance profile for time estimates the probability that a classifier c has a time ratio below τ. Again, $T_c(1)$ denotes the fraction of datasets where classifier c achieved the smallest training time among all classifiers.

Figure 7 shows the performance profile for the time needed to do the grid search. Since large differences in training time were observed, a logarithmic scale is used for the horizontal axis. This performance profile clearly shows that all MSVMpack methods suffer from long computation times. The fastest methods are the OvO and DAGSVM methods, followed by GenSVM and OvA. From the value of $T_c(1)$ it is found that OvO achieves the smallest grid search time on 40% of the datasets, versus 37% and 23% for DAGSVM and GenSVM, respectively. The other methods never achieve the smallest grid search time. It is important to note here that the grid search for GenSVM is 18 times larger than that of the other methods.

In addition to the performance profile, the average computation time per hyperparameter configuration was also examined. Here, GenSVM has an average training time of 1.98 seconds per configuration, versus 24.84 seconds for OvO and 25.03 seconds for DAGSVM. This is a considerable difference, which can be explained in part by the use of warm starts in GenSVM (see Section 4.3). However, in the performance profiles for training time GenSVM suffers from the number of hyperparameters in the grid search. When examining the average computation time per dataset, it is found that GenSVM takes on average 676 seconds, versus 472 and 476 seconds for OvO and DAGSVM, respectively. The difference between DAGSVM and OvO is attributed to the prediction strategy used by DAGSVM.

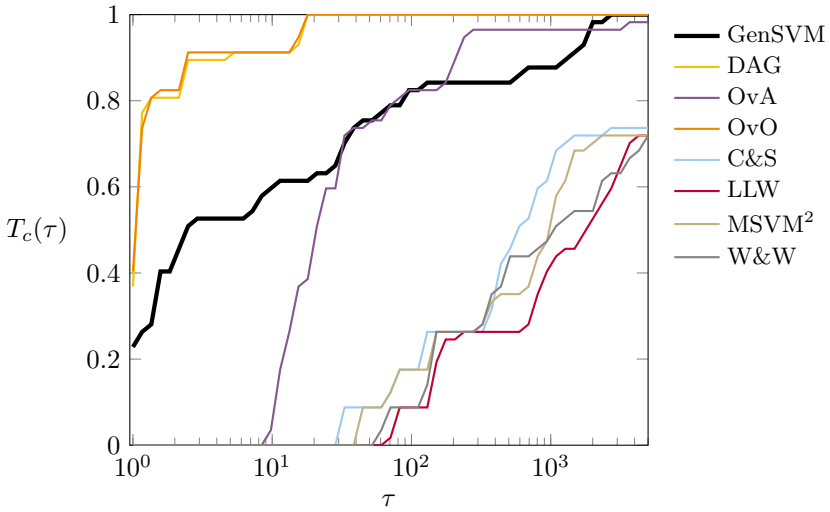


Figure 7: Performance profiles for training time. GenSVM has a priori about 23% chance of requiring the smallest time to perform the grid search on a given method. The methods implemented through MSVMpack will always be slower than any of the other methods.

6.3. Rank Tests

Following suggestions from Demšar (2006), the Friedman rank test can be used to find significant differences between classifiers (Friedman, 1937, 1940). If r_{cd} denotes the fractional rank of classifier c on dataset d, then with N_C classifiers and N_D datasets the Friedman statistic is given by

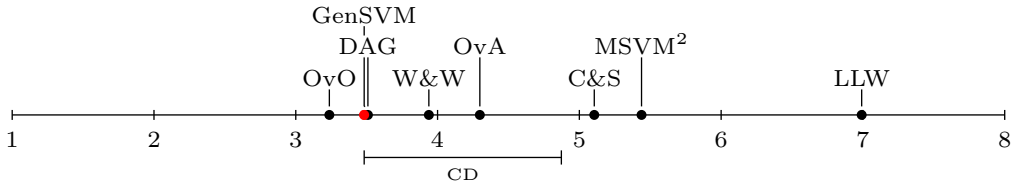
$$\chi_F^2 = \frac{12N_D}{N_C(N_C+1)}\left[\sum_c R_c^2 - \frac{N_C(N_C+1)^2}{4}\right]. \tag{36}$$

Here, $R_c = 1/N_D \sum_d r_{cd}$ denotes the average rank of classifier c. As Demšar notes, Iman and Davenport (1980) showed that the Friedman statistic is undesirably conservative and the F-statistic is to be used instead, which is given by

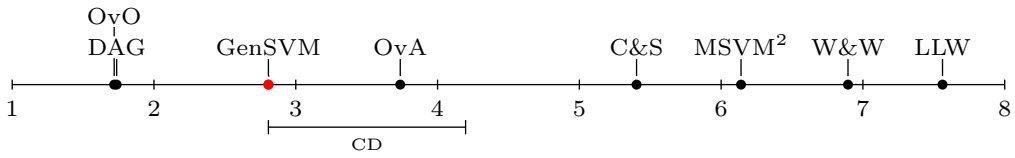
$$F_F = \frac{(N_D-1)\chi_F^2}{N_D(N_C-1)-\chi_F^2}. \tag{37}$$

Under the null hypothesis of either test there is no significant difference in the performance of any of the algorithms. Ranks are calculated for both the performance as measured by the ARI, as well as for the total training time needed to do the grid search.

Figure 8 shows the average ranks for both classification accuracy and training time for all classifiers. It can be seen that overall GenSVM ranks second in classification accuracy, and third in total training time. Demšar defines the critical difference (CD) as the minimal distance between two significantly different average ranks. The figures show this distance as measured from GenSVM onwards. This critical difference depends on the significance level used for the hypothesis testing, which is here chosen to be 5%.



(a) Performance



(b) Training time

Figure 8: Figure (a) shows the average ranks for performance, whereas (b) shows the average ranks for the total computation time needed for the grid search. It can be seen that GenSVM obtains the second overall rank in performance, and third overall rank in training time. In both figures, CD shows the critical difference. Classifiers beyond this CD differ significantly from GenSVM at the 5% significance level.

When performing the Friedman test, it is found that for classifier performance $\chi_F^2 = 108.52$ ($p = 0.0$), and $F_F = 20.92$ ($p \approx 10^{-16}$). Hence, with both tests the null hypothesis of equal classification accuracy can be rejected. Similarly, for training time the test statistics are $\chi_F^2 = 355.58$ ($p = 0.0$) and $F_F = 458.66$ ($p \approx 10^{-16}$). Therefore, the null hypothesis of equal training time can also be rejected.

When significant differences are found through the Friedman test, Demšar (2006) suggests to use Holm's step-down procedure as a post-hoc test, to find which classifiers differ significantly from a chosen reference classifier (Holm, 1979). Here, GenSVM is used as a reference classifier, since comparing GenSVM with existing methods is the main focus of these experiments.

Holm's procedure is based on testing whether the z-statistic comparing classifier i with classifier j is significant, while adjusting for the familywise error rate. Following Demšar (2006), this z-statistic is given by

$$z = (R_i - R_j)\sqrt{\frac{6N_D}{N_C(N_C+1)}}. \tag{38}$$

Subsequently, the p-values computed from this statistic are sorted in increasing order, as $p_1, p_2, \ldots, p_{N_C-1}$. Then, the null hypothesis of equal classification accuracy can be rejected if $p_i < \alpha/(N_C - i)$ for $i = 1, \ldots, N_C - 1$. If for some i the null hypothesis cannot be rejected, all subsequent tests will also fail.

Using Holm's procedure, it is found that at the 5% significance level GenSVM significantly outperforms the method of Lee et al. (2004) ($p = 10^{-14}$), the method of Guermeur and Monfrini (2011) ($p = 10^{-5}$), and that of Crammer and Singer (2002a) ($p = 0.0002$). In addition, it is found that GenSVM is significantly faster than all methods implemented

through MSVMpack (C&S, W&W, MSVM^2, and LLW) at the 5% significance level. At the 10% significance level GenSVM is also significantly faster than OvA.

7. Discussion

A generalized multiclass support vector machine has been introduced, called *GenSVM*. The method is general in the sense that it subsumes two multiclass SVMs proposed in the literature, and it is flexible due to several different weighting options. An iterative majorization algorithm has been derived to minimize the convex GenSVM loss function in the primal. This primal optimization approach has computational advantages due to the possibility to use warm starts, and because it can be intuitively understood. Computational tests show that GenSVM significantly outperforms three existing multiclass SVMs on predictive accuracy, and four on grid search training time, at the 5% significance level.

In the comparison tests, MSVMpack (Lauer and Guermeur, 2011) was used to access four single machine multiclass SVMs proposed in the literature. A big advantage of using this library is that it allows for a single straightforward `C` implementation, which greatly reduces the programming effort needed for the comparisons. However, as is noted in the MSVMpack documentation, slight differences exist between MSVMpack and method-specific implementations. For instance, on small datasets MSVMpack can be slower, due to working set selection and shrinking procedures in other implementations. However, classification performance is comparable between MSVMpack and method-specific implementations. Thus, it is argued that the results for predictive accuracy presented above are accurate regardless of implementation, but small differences can exist for training time when other implementations for single machine MSVMs are used.

Due to the large number of datasets and the long training time of some methods, only linear classification was compared. In Appendix B it is argued that kernel GenSVM can be achieved through an adapted linear GenSVM obtained by an eigendecomposition on the kernel matrix, which is a process of the order $O(n^3)$. After this preprocessing step, the computational time will be similar to linear GenSVM. In this case, GenSVM may benefit from precomputing kernels before starting the grid search, or using a larger stopping criterion in the IM algorithm. In addition, more approximations can be done by using rank approximated kernel matrices, such as those proposed by Williams and Seeger (2001). Such enhancements are considered topics for further research.

Finally, the potential of using GenSVM in an online setting is recognized. Since the solution can be found quickly when a warm-start is used, GenSVM may be useful in situations where new instances have to be predicted at a certain moment, and the true class label arrives later. Then, reestimating the GenSVM solution can be done as soon as the true class label of an object arrives, and a previously known solution can be used as a warm start. It is expected that in this scenario only a few iterations of the IM algorithm are needed to arrive at a new optimal solution. This too is considered a subject for further research.

Acknowledgements

The computational tests of this research were performed on the Dutch National LISA Compute Cluster, and supported by the Dutch National Science Foundation (NWO). The authors thank SURFsara (`www.surfsara.nl`) for the support in using the LISA cluster.

A. Huber Hinge Majorization

In this appendix the majorization function will be derived of the Huber hinge error raised to the power p. Thus, a quadratic function $g(x, \overline{x}) = ax^2 - 2bx + c$ is required, which is a majorization function of

$$f(x) = h^p(x) = \begin{cases} \left(1 - x - \frac{\kappa+1}{2}\right)^p & \text{if } x \leq -\kappa \\ \frac{1}{(2(\kappa+1))^p}(1-x)^{2p} & \text{if } x \in (-\kappa, 1] \\ 0 & \text{if } x > 1, \end{cases} \tag{39}$$

with $p \in [1, 2]$. Each piece of $f(x)$ provides a possible region for the supporting point $\overline{x}$. These regions will be treated separately, starting with $\overline{x} \in (-\kappa, 1]$.

Since the majorization function must touch $f(x)$ at the supporting point, we can solve $f(\overline{x}) = g(\overline{x}, \overline{x})$ and $f'(\overline{x}) = g'(\overline{x}, \overline{x})$ for b and c to find

$$b = a\overline{x} + \frac{p}{1-\overline{x}} \left(\frac{1-\overline{x}}{\sqrt{2(\kappa+1)}} \right)^{2p}, \tag{40}$$

$$c = a\overline{x}^2 + \left(1 + \frac{2p\overline{x}}{1-\overline{x}}\right) \left(\frac{1-\overline{x}}{\sqrt{2(\kappa+1)}} \right)^{2p}, \tag{41}$$

whenever $\overline{x} \in (-\kappa, 1]$. Note that since $p \in [1, 2]$ the function $f(x)$ can become proportional to a fourth power on the interval $x \in (-\kappa, 1]$. The upper bound of the second derivative of $f(x)$ on this interval is reached at $x = -\kappa$. Equating $f''(-\kappa)$ to $g''(-\kappa, \overline{x}) = 2a$ and solving for a yields

$$a = \tfrac{1}{4}p(2p-1) \left(\frac{\kappa+1}{2} \right)^{p-2}. \tag{42}$$

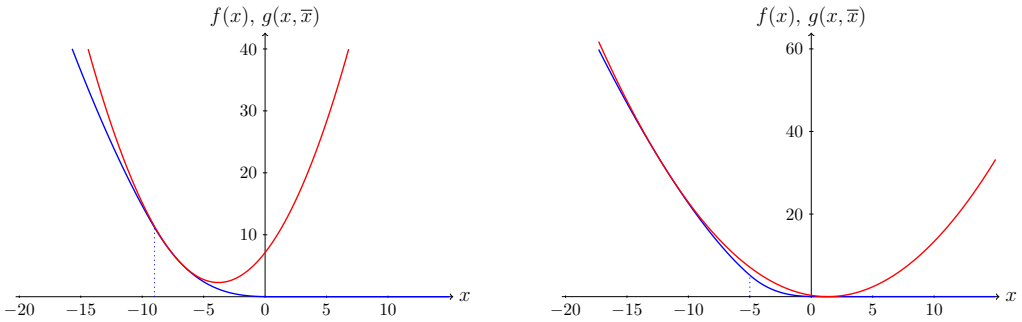
Figure 9(a) shows an illustration of the majorization function when $\overline{x} \in (-\kappa, 1]$.

For the interval $\overline{x} \leq -\kappa$ the following expressions are found for b and c using similar reasoning as above

$$b = a\overline{x} + \tfrac{1}{2}p \left(1 - \overline{x} - \frac{\kappa+1}{2} \right)^{p-1}, \tag{43}$$

$$c = a\overline{x}^2 + p\overline{x} \left(1 - \overline{x} - \frac{\kappa+1}{2} \right)^{p-1} + \left(1 - \overline{x} - \frac{\kappa+1}{2} \right)^{p}. \tag{44}$$

To obtain the largest possible majorization step it is desired that the minimum of the majorization function is located at $x \geq 1$, such that $g(x_{min}, \overline{x}) = 0$. This requirement yields



(a) **(b)**

Figure 9: Graphical illustration of the majorization of the function $f(x) = h^p(x)$. Figure (a) shows the case where $\overline{x} \in (-\kappa, 1]$, whereas (b) shows the case where $\overline{x} \leq (p+\kappa-1)/(p-2)$. In both cases $p = 1.5$. It can be seen that in (b) the minimum of the majorization function lies at $x > 1$, such that the largest possible majorization step is obtained.

$c = b^2/a$, which gives

$$a = \tfrac{1}{4}p^2\left(1-\overline{x}-\frac{\kappa+1}{2}\right)^{p-2}. \tag{45}$$

Note however that due to the requirement that $f(x) \leq g(x,\overline{x})$ for all $x \in \mathbb{R}$, this majorization is not valid for all values of $\overline{x}$. Solving the requirement for the minimum of the majorization function, $g(x_{min},\overline{x}) = 0$ for $\overline{x}$ yields

$$\overline{x} \leq \frac{p+\kappa-1}{p-2}. \tag{46}$$

Thus, if $\overline{x}$ satisfies this condition, (45) can be used for a, whereas for cases where $\overline{x} \in ((p+\kappa-1)/(p-2), -\kappa]$, the value of a given in (42) can be used. Figure 9(b) shows an illustration of the case where $\overline{x} \leq (p+\kappa-1)/(p-2)$.

Next, a majorization function for the interval $\overline{x} > 1$ is needed. Since it has been derived that for the interval $\overline{x} \leq (p+\kappa-1)/(p-2)$ the minimum of the majorization function lies at $x \geq 1$, symmetry arguments can be used to derive the majorization function for $\overline{x} > 1$, and ensure that it is also tangent at $x = (p\overline{x}+\kappa-1)/(p-2)$. This yields the coefficients

$$a = \tfrac{1}{4}p^2\left(\frac{p}{p-2}\left(1-\overline{x}-\frac{\kappa+1}{2}\right)\right)^{p-2}, \tag{47}$$

$$b = a\left(\frac{p\overline{x}+\kappa-1}{p-2}\right) + \tfrac{1}{2}\left(\frac{p}{p-2}\left(1-\overline{x}-\frac{\kappa+1}{2}\right)\right)^{p-1}, \tag{48}$$

$$\begin{aligned} c = {} & a\left(\frac{p\overline{x}+\kappa-1}{p-2}\right)^2 + p\left(\frac{p\overline{x}+\kappa-1}{p-2}\right)\left(\frac{p}{p-2}\left(1-\overline{x}-\frac{\kappa+1}{2}\right)\right)^{p-1} \\ & + \left(\frac{p}{p-2}\left(1-\overline{x}-\frac{\kappa+1}{2}\right)\right)^{p}. \end{aligned} \tag{49}$$

Finally, observe that some of the above coefficients are invalid if $p = 2$. However, since the upper bound on the interval $\overline{x} \in (-\kappa, 1]$ given in (42) is still valid if $p = 2$, it is possible

Table 2: Overview of quadratic majorization coefficients for different pieces of $h^p(x)$, depending on $\overline{x}$.

	a	b	c
$\overline{x} \leq \dfrac{p+\kappa-1}{p-2}$	(45)	(43)	(44)
$\overline{x} \in \left(\dfrac{p+\kappa-1}{p-2}, -\kappa\right]$	(42)	(43)	(44)
$\overline{x} \in (-\kappa, 1]$	(42)	(40)	(41)
$\overline{x} > 1,\ p \neq 2$	(47)	(48)	(49)
$\overline{x} > 1,\ p = 2$	(42)	$a\overline{x}$	$a\overline{x}^2$

to do a separate derivation with this value for a to find for $\overline{x} > 1$, $b = a\overline{x}$ and $c = a\overline{x}^2$. For the other regions the previously derived coefficients still hold. Table 2 gives an overview of the various coefficients depending on the location of $\overline{x}$.

B. Kernels in GenSVM

To include kernels in GenSVM a pre-processing step is needed on the kernel matrix, and a post-processing step is needed on the obtained parameters before doing class prediction. Let $k : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^+$ denote a positive definite kernel satisfying Mercer's theorem, and let $\mathcal{H}_k$ denote the corresponding reproducing kernel Hilbert space. Furthermore, define a feature mapping $\phi : \mathbb{R}^m \to \mathcal{H}_k$ as $\phi(\mathbf{x}) = k(\mathbf{x}, \cdot)$, such that by the reproducing property of k it holds that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}_k}$.

Using this, the kernel matrix $\mathbf{K}$ is defined as the $n \times n$ matrix with elements $k(\mathbf{x}_i, \mathbf{x}_j)$ on the i-th row and j-th column. Thus, if $\mathbf{\Phi}$ denotes the $n \times l$ matrix with rows $\phi(\mathbf{x}_i)$ for $i = 1, \ldots, n$ and $l \in [1, \infty]$, then $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}'$. Note that it depends on the chosen kernel whether $\mathbf{\Phi}$ is finite dimensional. However, the rank of $\mathbf{\Phi}$ can still be determined through $\mathbf{K}$, since $r = \operatorname{rank}(\mathbf{\Phi}) = \operatorname{rank}(\mathbf{K}) \leq \min(n, l)$.

Now, let the reduced singular value decomposition of $\mathbf{\Phi}$ be given by

$$\mathbf{\Phi} = \mathbf{P}\mathbf{\Sigma}\mathbf{Q}',$$

where $\mathbf{P}$ is $n \times r$, $\mathbf{\Sigma}$ is $r \times r$, and $\mathbf{Q}$ is $l \times r$. Note that here, $\mathbf{P}'\mathbf{P} = \mathbf{I}_r$, $\mathbf{Q}'\mathbf{Q} = \mathbf{I}_r$, and $\mathbf{\Sigma}$ is diagonal. Under the mapping $\mathbf{X} \to \mathbf{\Phi}$ it follows that the simplex space vectors become

$$\begin{aligned} \mathbf{S} &= \mathbf{\Phi}\mathbf{W} + \mathbf{1}\mathbf{t}' \\ &= \mathbf{P}\mathbf{\Sigma}\mathbf{Q}'\mathbf{W} + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\mathbf{Q}'\mathbf{W} + \mathbf{1}\mathbf{t}'. \end{aligned}$$

Here $\mathbf{W}$ is $l \times (K-1)$ to correspond to the dimensions of $\mathbf{\Phi}$, and the $n \times r$ matrix $\mathbf{M} = \mathbf{P}\mathbf{\Sigma}$ has been introduced. In general $\mathbf{W}$ cannot be determined, since l might be infinite. This

problem can be solved as follows. Decompose $\mathbf{W}$ in two parts, $\mathbf{W} = \mathbf{W}_1 + \mathbf{W}_2$, where $\mathbf{W}_1$ is in the linear space of $\mathbf{Q}$ and $\mathbf{W}_2$ is orthogonal to that space, thus

$$\begin{aligned}\mathbf{W}_1 &= \mathbf{Q}\mathbf{Q}'\mathbf{W}, \\ \mathbf{W}_2 &= (\mathbf{I}_l - \mathbf{Q}\mathbf{Q}')\mathbf{W}.\end{aligned}$$

Then it follows that

$$\begin{aligned}\mathbf{S} &= \mathbf{M}\mathbf{Q}'\mathbf{W} + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\mathbf{Q}'(\mathbf{W}_1 + \mathbf{W}_2) + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\mathbf{Q}'(\mathbf{W}_1 + (\mathbf{I}_l - \mathbf{Q}\mathbf{Q}')\mathbf{W}) + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\mathbf{Q}'\mathbf{W}_1 + \mathbf{M}(\mathbf{Q}' - \mathbf{Q}'\mathbf{Q}\mathbf{Q}')\mathbf{W} + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\mathbf{Q}'\mathbf{W}_1 + \mathbf{M}(\mathbf{Q}' - \mathbf{Q}')\mathbf{W} + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\mathbf{Q}'\mathbf{W}_1 + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\mathbf{Q}'\mathbf{W}_1 + \mathbf{1}\mathbf{t}',\end{aligned}$$

where it has been used that $\mathbf{Q}'\mathbf{Q} = \mathbf{I}_r$. If the penalty term of the GenSVM loss function is considered, it is found that

$$P_\lambda(\mathbf{W}) = \lambda \operatorname{tr} \mathbf{W}'\mathbf{W} = \lambda \operatorname{tr} \mathbf{W}_1'\mathbf{W}_1 + \lambda \operatorname{tr} \mathbf{W}_2'\mathbf{W}_2,$$

since

$$\begin{aligned}\mathbf{W}_1'\mathbf{W}_2 &= \mathbf{W}'\mathbf{Q}\mathbf{Q}'(\mathbf{I}_l - \mathbf{Q}\mathbf{Q}')\mathbf{W} \\ &= \mathbf{W}'\mathbf{Q}\mathbf{Q}'\mathbf{W} - \mathbf{W}'\mathbf{Q}\mathbf{Q}'\mathbf{W} \\ &= \mathbf{O}.\end{aligned}$$

Here again it has been used that $\mathbf{Q}'\mathbf{Q} = \mathbf{I}_r$, and $\mathbf{O}$ is defined as a $(K-1) \times (K-1)$ dimensional matrix of zeroes. Note that the penalty term depends on $\mathbf{W}_2$ whereas the simplex vectors $\mathbf{S}$ do not. Therefore, at the optimal solution it is required that $\mathbf{W}_2$ is zero, to minimize the loss function.

Since $\mathbf{W}_1$ is still $l \times (K-1)$ dimensional with l possibly infinite, consider the substitution $\mathbf{W}_1 = \mathbf{Q}\boldsymbol{\Omega}$, with $\boldsymbol{\Omega}$ an $r \times (K-1)$ matrix. The penalty term in terms of $\boldsymbol{\Omega}$ then becomes

$$P_\lambda(\mathbf{W}_1) = \lambda \operatorname{tr} \mathbf{W}_1'\mathbf{W}_1 = \lambda \operatorname{tr} \boldsymbol{\Omega}'\mathbf{Q}'\mathbf{Q}\boldsymbol{\Omega} = \lambda \operatorname{tr} \boldsymbol{\Omega}'\boldsymbol{\Omega} = P_\lambda(\boldsymbol{\Omega}).$$

Note also that

$$\begin{aligned}\mathbf{S} &= \mathbf{M}\mathbf{Q}'\mathbf{W}_1 + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\mathbf{Q}'\mathbf{Q}\boldsymbol{\Omega} + \mathbf{1}\mathbf{t}' \\ &= \mathbf{M}\boldsymbol{\Omega} + \mathbf{1}\mathbf{t}'.\end{aligned}$$

The question remains on how to determine the matrices $\mathbf{P}$ and $\boldsymbol{\Sigma}$, given that the matrix $\boldsymbol{\Phi}$ cannot be determined explicitly. These matrices can be determined by the eigendecomposition of $\mathbf{K}$, where $\mathbf{K} = \mathbf{P}\boldsymbol{\Sigma}^2\mathbf{P}'$. In the case where $r < n$, $\boldsymbol{\Sigma}^2$ contains only the first r

eigenvalues of $\mathbf{K}$, and $\mathbf{P}$ the corresponding r columns. Hence, if $\mathbf{K}$ is not of full rank, a dimensionality reduction is achieved in $\boldsymbol{\Omega}$. The complexity of finding the eigendecomposition of the kernel matrix is $O(n^3)$.

Since the distances $q_i^{(kj)}$ in the GenSVM loss function can be written as $q_i^{(kj)} = \mathbf{s}_i' \boldsymbol{\delta}_{kj}$ it follows that the errors can again be calculated in this formulation. Finally, to predict the simplex space vectors of a test set $\mathbf{X}_2$ the following is used. Let $\boldsymbol{\Phi}_2$ denote the feature space mapping of $\mathbf{X}_2$, then

$$\begin{aligned}
\mathbf{S}_2 &= \boldsymbol{\Phi}_2 \mathbf{W}_1 + \mathbf{1}\mathbf{t}' \\
&= \boldsymbol{\Phi}_2 \mathbf{Q} \boldsymbol{\Omega} + \mathbf{1}\mathbf{t}' \\
&= \boldsymbol{\Phi}_2 \mathbf{Q} \boldsymbol{\Sigma} \mathbf{P}' \mathbf{P} \boldsymbol{\Sigma}^{-1} \boldsymbol{\Omega} + \mathbf{1}\mathbf{t}' \\
&= \boldsymbol{\Phi}_2 \boldsymbol{\Phi}' \mathbf{P} \boldsymbol{\Sigma}^{-1} \boldsymbol{\Omega} + \mathbf{1}\mathbf{t}' \\
&= \mathbf{K}_2 \mathbf{P} \boldsymbol{\Sigma}^{-1} \boldsymbol{\Omega} + \mathbf{1}\mathbf{t}' \\
&= \mathbf{K}_2 \mathbf{M} \boldsymbol{\Sigma}^{-2} \boldsymbol{\Omega} + \mathbf{1}\mathbf{t}',
\end{aligned}$$

where $\mathbf{K}_2 = \boldsymbol{\Phi}_2 \boldsymbol{\Phi}'$ is the kernel matrix between the test set and the training set, and it was used that $\boldsymbol{\Sigma} \mathbf{P}' \mathbf{P} \boldsymbol{\Sigma}^{-1} = \mathbf{I}_r$, and $\boldsymbol{\Phi}' = \mathbf{Q} \boldsymbol{\Sigma} \mathbf{P}'$ by definition.

With the above expressions for $\mathbf{S}$ and $P_\lambda(\boldsymbol{\Omega})$, it is possible to derive the majorization function of the loss function for the nonlinear case. The first order conditions can then again be determined, which yields the following system

$$\left(\begin{bmatrix} \mathbf{1}' \\ \mathbf{M}' \end{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{1} & \mathbf{M} \end{bmatrix} + \lambda \begin{bmatrix} 0 & \mathbf{0}' \\ \mathbf{0} & \mathbf{I}_r \end{bmatrix} \right) \begin{bmatrix} \mathbf{t}' \\ \boldsymbol{\Omega} \end{bmatrix} = \begin{bmatrix} \mathbf{1}' \\ \mathbf{M}' \end{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{1} & \mathbf{M} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{t}}' \\ \overline{\boldsymbol{\Omega}} \end{bmatrix} + \begin{bmatrix} \mathbf{1}' \\ \mathbf{M}' \end{bmatrix} \mathbf{B}. \tag{50}$$

This system is analogous to the system solved in linear GenSVM. In fact, it can be shown that by writing $\mathbf{Z} = [\mathbf{1}\ \mathbf{M}]$ and $\mathbf{V} = [\mathbf{t}'\ \boldsymbol{\Omega}]'$, this system is equivalent to (31). This property is very useful for the implementation of GenSVM, since nonlinearity can be included by simply adding a pre- and post-processing step to the existing GenSVM algorithm.

References

A. Aizerman, E.M. Braverman, and L.I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25: 821–837, 1964.

E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141, 2001.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

C.C.J.H. Bijleveld and J. De Leeuw. Fitting longitudinal reduced-rank regression models by alternating least squares. *Psychometrika*, 56(3):433–447, 1991.

E.J. Bredensteiner and K.P. Bennett. Multicategory classification by support vector machines. *Computational Optimization and Applications*, 12(1):53–79, 1999.

C.C. Chang and C.J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5): 1155–1178, 2007.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002a.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2-3):201–233, 2002b.

J. De Leeuw. Applications of convex analysis to multidimensional scaling. In *Recent Developments in Statistics*, pages 133–146. North Holland Publishing Company, Amsterdam, 1977.

J. De Leeuw. Convergence of the majorization method for multidimensional scaling. *Journal of Classification*, 5(2):163–180, 1988.

J. De Leeuw. Fitting distances by least squares. Technical Report 130, Los Angeles: Interdivisional Program in Statistics, UCLA, 1993.

J. De Leeuw. Block-relaxation algorithms in statistics. In *Information systems and data analysis*, pages 308–324. Springer, 1994.

J. De Leeuw and W.J. Heiser. Multidimensional scaling with restrictions on the configuration. *Multivariate Analysis*, 5:501–522, 1980.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

U. Dogan, T. Glasmachers, and C. Igel. Fast training of multi-class support vector machines. Technical Report 03/2011, University of Copenhagen, Faculty of Science, 2011.

E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

P.J.F. Groenen and W.J. Heiser. The tunneling method for global optimization in multidimensional scaling. *Psychometrika*, 61(3):529–550, 1996.

P.J.F. Groenen, G. Nalbantov, and J.C. Bioch. Nonlinear support vector machines through iterative majorization and I-splines. In *Advances in Data Analysis*, pages 149–161. Springer Berlin Heidelberg, 2007.

P.J.F. Groenen, G. Nalbantov, and J.C. Bioch. SVM-Maj: A majorization approach to linear support vector machines with different hinge errors. *Advances in Data Analysis and Classification*, 2(1):17–43, 2008.

Y. Guermeur and E. Monfrini. A quadratic loss multi-class SVM for which a radius–margin bound applies. *Informatica*, 22(1):73–96, 2011.

G.H. Hardy, J.E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1934.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2nd edition, 2009.

T. F. Havel. An evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance. *Progress in biophysics and molecular biology*, 56(1):43–78, 1991.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

C.W. Hsu and C.J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

P.J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

D.R. Hunter and K. Lange. A tutorial on MM algorithms. *The American Statistician*, 58 (1):30–37, 2004.

R.L. Iman and J.M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, 1980.

U.H.G. Kreßel. Pairwise classification and support vector machines. In *Advances in Kernel Methods*, pages 255–268. MIT Press, 1999.

F. Lauer and Y. Guermeur. MSVMpack: A multi-class support vector machine package. *Journal of Machine Learning Research*, 12:2269–2272, 2011.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

J.M. Ortega and W.C. Rheinboldt. *Iterative Solutions of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.

J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, pages 185–208. MIT press, 1999.

J.C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, 12(3):547–553, 2000.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1997.

S. Rosset and J. Zhu. Piecewise linear regularized solution paths. *The Annals of Statistics*, 35(3):1012–1030, 2007.

J.M. Santos and M. Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, pages 175–184. Springer-Verlag, 2009.

H. Tsutsu and Y. Morikawa. An l_p norm minimization using auxiliary function for compressed sensing. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2012.

V. Vapnik. *Statistical learning theory*. Wiley, New York, 1998.

H. Voss and U. Eckhardt. Linear convergence of generalized Weiszfeld's method. *Computing*, 25(3):243–251, 1980.

E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal*, 43:355–386, 1937.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, University of London, Royal Holloway, Department of Computer Science, 1998.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, pages 682–688, 2001.