Predicting the Political Spectrum with Support Vector Machine

Hok San Yip
Erasmus University Rotterdam

Master Thesis
Quantitative Marketing
Econometrics and Management Science
Rotterdam School of Economics
Supervisor: Prof. dr. P.J.F. Groenen
Co-reader: Prof. dr. D. Fok

Predicting the Political Spectrum with Support Vector Machine

Hok San Yip

Abstract

Support Vector Machines (SVMs) have gained considerable popularity over the last two decades for binary classification. This paper concentrates on a recent optimization approach to SVMs, the SVM majorization approach, or SVM-Maj for short. This method is aimed at small and medium sized Support Vector Machine (SVM) problems, in which SVM-Maj performs well relative to other solvers. To obtain an SVM solution, most other solvers need to solve the dual problem. In contrast, SVM-Maj solves the primal SVM optimization iteratively thereby converging to the SVM solution. Furthermore, the simplicity of SVM-Maj makes it intuitively more accessible to the researcher than the state-of-art decomposition methods. Moreover, SVM-Maj can easily handle any well-behaved error function, while the traditional SVM solvers focus particularly on the absolute-hinge error. In this paper, the SVM-Maj approach is enhanced to include the use of different kernels, the standard way in the SVM literature for handling nonlinearities in the predictor space. In addition, the R package **SVMMaj** is introduced that implements this methodology. Amongst its features are the weighting of the error for individual objects in the training dataset, handling nonlinear prediction through monotone spline transformations and through kernels, and functions to do cross validation.

As an application of SVM, this paper also investigates the practicability of using Support Vector Machine as an automatic Machine Learning Technique on predicting the political spectrum of a person based on their choice of words in political issues. For this research, the statements of the parliamentarians in the plenary meetings of the House of Representatives have been used. Using the term frequency of the selected features in the model, it was possible to obtain a hit rate of up to nearly 70%, which was higher than the hit rate obtained by purely majority voting.

Keywords: Support Vector Machine, SVM-Maj, R, Majorization, text analysis, political spectrum, Dutch House of Representatives.

1. Introduction

For understanding what a support vector machine (SVM) is, consider the following data analysis problem: there are two groups of objects, say products of type A and type B, having some common attributes such as color, price, weight, etc. These attributes will be referred to as predictor variables in this paper. The task of separating the two types of objects from each other is formally referred to as the binary classification task. Given the values for the predictor variables of a new object, we would like to assign this new object to a given group, or class. Such a binary classification task is dealt with routinely in medical, technical, economic, humanitarian, and other fields.

Numerous learning methods have been designed to solve the binary classification task, including Linear Discriminant Analysis, Binary Logistic Regression, Neural Networks, Decision

Trees, Naive Bayes classifier, and others. In this paper, I focus on a method that has gained considerable popularity over the last two decades, called the Support Vector Machines (SVMs) classifier (Vapnik 1995). SVMs have emerged as one of the most popular and high-performing learning methods for classification. They have been successfully applied to areas ranging from bioinformatics (see, e.g., Furey *et al.* 2000; Guyon *et al.* 2002) to image recognition (see, e.g., Chapelle *et al.* 1999; Pontil and Verri 1998) and marketing (see, e.g., Cui and Curry 2005). In essence, SVMs divide two groups of objects from each other by building a hyperplane in the space of the predictor variables that separates them in an adequate way. The rapid success of SVMs can be attributed to a number of key qualities: robustness of results, avoiding overfitting, and the possibility to handle nonlinearities of the predictor variables easily through so-called kernel functions. In addition, the evaluation of an SVM model on test objects is relatively fast and simple. The SVM is formulated as a well-defined quadratic optimization problem that has a unique solution. Overfitting, that is, fitting an available training data-set too well, is avoided via a *penalty* term that suppresses too complex potential fits. Nonlinearities can be handled in two ways: (1) by a preprocessing step of the predictor variables, for example, by polynomial expansion or the use of splines (Groenen *et al.* 2007) and (2) by using the *kernel* trick that allows nonlinear solutions to be computed implicitly. Note that the use of kernels is very prominent in the SVM literature.

A variety of solvers for the SVM optimization task have been proposed in the literature. One of the initial ideas has been to apply general-purpose quadratic optimization solvers such as LOQO (Vanderbei 1994) and MINOS (Murtagh and Saunders 1998). One of the problems of such solvers is that they require the whole *kernel* matrix to be stored in memory, which is quadratic in the number of observations n. For small scale problems, this is not a problem, but for medium and large scale problems other methods are needed. One attempt to overcome the complete storage of the kernel matrix is by chunking (Boser *et al.* 1992; Osuna *et al.* 1997a). This method concentrates on a (working) subset of all training objects at a given iteration, effectively splitting the learning task into smaller subproblems that easily fit into the memory of a computer. Alternatively, direction search was proposed (Boser *et al.* 1992) that updates all unknown coefficients at each iteration in a certain feasible direction.

More recently, decomposition methods have established themselves as the mainstream technique for solving SVMs (Osuna *et al.* 1997b; Saunders *et al.* 1998; Joachims 1999). At each iteration, the decomposition method optimizes a subset of coefficients, and leaves the remaining coefficients unchanged. Solving a series of very simple optimization subproblems, this approach has proven to be one of the fastest for large scale SVM problems. The most popular decomposition method is the so-called Sequential Minimal Optimization (SMO) (Platt 1999), where only two coefficients are updated at each iteration, which can actually be done analytically. SMO is the basis of popular SVM solvers such as SVMlight (Joachims 1999) and LibSVM (Chang and Lin 2001). For the linear SVM case, alternative techniques to the decomposition methods have recently been put forward, such as the cutting plane algorithm (Joachims 2006).

This paper concentrates on a recent optimization approach to solving SVMs (Groenen *et al.* 2007, 2008), referred to as the majorization approach to SVMs, or SVM-Maj for short. This method is aimed at small and medium sized SVM problems. An overview of some popular SVM solvers and some of their properties is given in Table 1. The other solvers are LibSVM, SVMlight, SVMTorch (Collobert and Bengio 2001), mySVM (Rüping 2000), SVM-Perf (Joachims 2006), and LibLINEAR (Fan *et al.* 2008).

Properties	**SVM-Maj**	**LIBSVM**	**SVMlight**	**SVMTorch**	**mySVM**	**SVM-Perf**	**LIBLINEAR**
Nonlinear kernels	yes	yes	yes	yes	yes	no	no
Splines	yes	no	no	no	no	no	no
Suitable for large n	yes	yes	yes	yes	yes	no	yes
Suitable for large k	yes	yes	yes	no	yes	no	yes
Suitable for large n and large k	no	yes	yes	no	yes	no	
Dual approach	no	yes	yes	yes	yes	yes	no
Allows quadratic hinge	yes	yes*	no	no	yes	no	yes
Allows Huber hinge	yes	no	no	no	no	no	no
Allows k-fold cross val.	yes	yes	yes	no	yes	no	yes
Language	MatLab, R	C++,Java	C, C++	C++	C	C++	C,C++
Gui interface or syntax-like	MatLab, R	command prompt	command prompt	command prompt	command prompt	command prompt	command prompt
Availability in R	yes	yes	no	no	no	no	yes
Multi-class problems	no	yes	yes	yes	no	yes	yes

* after modification

Table 1: Comparison table of different SVM solvers available in 2010.

In this paper, the method of Groenen *et al.* (2008) is enhanced to include the use of different kernels, the standard way in the SVM literature for handling nonlinearities in the predictor space. Moreover, a new special treatment is offered in this paper for the case where the number of objects is less than the number of predictor variables. Three cases are distinguished for which SVM-Maj computes the solution efficiently: (1) the case with many more observations n than variables k, (2) the case with many variables k but medium sized n, and (3) the case that n is medium sized and the kernel space or the space of the variables is of lower rank than k or n. The case of large n is relatively more difficult for all SVM solvers, including SVM-Maj. This particularly holds when kernels are used. For this case, the alternative of introducing nonlinearity through splines in SVM-Maj can be used so that still for medium sized n, nonlinearity can be handled efficiently.

Amongst the advantages of the SVM-Maj approach are its intuitive optimization algorithm, its versatility, and the competitively fast estimation speed for medium sized problems. The majorization solver is an iterative algorithm with two easily tractable alternating steps that reveal the nature of solving the SVM optimization problem in an appealing way. The relative simplicity of SVM-Maj arguably makes it intuitively more accessible to the researcher than the state-of-art decomposition methods. Traditional SVM solvers focus particularly on the absolute-hinge error (the standard SVM error function), whereas the majorization algorithm has been designed to incorporate any well-behaved error function. This property can be viewed as a attractive feature of the majorization approach. The SVM-Maj package comes with the following in-built error functions: the classic absolute-hinge, Huber-hinge, and quadratic hinge. Furthermore, SVM-Maj solves the *primal* SVM optimization problem even in the nonlinear case, in contrast to other solvers, which solve the *dual* problem. The advantage of solving the primal is that SVM-Maj converges to the optimal solution in each iterative step. In contrast, other methods solving the dual optimization problem need the dual problem to be fully converged to attain a solution.

The main features of the package **SVMMaj** are: implementation of the SVMMaj majorization algorithm for SVMs, handling of nonlinearity through splines and kernels, the ability to handle several error functions (among other the classic hinge, quadratic hinge and Huber hinge error).

In addition, SVM-Maj is able to assign a fixed weight to each individual objects in a training dataset to receive different individual weights. In this way, the user can set importance of misclassifying the object by varying the individual weight. These weights can also be set per class. The SVM-Maj package comes with a cross validation function to asses out-of-sample performance or evaluate meta parameters that come with certain SVM models.

One of the applications of SVM is text analysis: a way to extract specific information from a text without the need of a human. Among others, (Pang *et al.* 2002) extracted the sentiment of the movie review - whether a movie was good or bad. The idea behind automated text analysis is to find a way to convert text into meaningful numbers, which is done by defining a list of *features*. A feature is word or a combination of words which serve as an indicator of the sentiment. For example, if the word ‘good’ has been mentioned frequently, a higher likelihood that the review has given a positive rating. Therefore, the frequency that the word ‘good’ appears in a text is an adequate feature to be used in the data analysis. Previous researches have shown that, by using such features, it is possible to predict its sentiment very well.

As an example of using SVM in text analysis, the political speeches during the plenary meetings of the Dutch House of Representatives (in Dutch: De Tweede Kamer) is used to analyse the difference in contents between the political spectra of the parliamentarians. In particular, the parliamentarians have been divided into two classes: the *progressive* and the *conservative* parliamentarians. If a decent classification model can be set up based on these speeches of the politicians, it can then be used to classify text in for example forums, where the sentiment of the bloggers are not known in advance. This automated classification method can be very valuable for analysis of the segmentation of the spectra betwee bloggers, or even between forum sites.

This paper is organized as follows. First, Section 6 gives a brief introduction to SVM and the majorization approach to solving SVMs, or SVM-Maj for short. Section 2 gives technical specification of SVM. In Section 3, the SVM-Maj algorithm is described and its update for each iteration is derived. Furthermore, Section 3 also discusses some computational efficiencies, and Section 4 presents the way nonlinearities are handled. Section 5 gives several detailed examples of how the **SVMMaj** package can be used. Subsequently, the research in text analysis will be discussed. In Section 7, the data which has been used for the text analysis is introduced followed by the methodology of the research, while Section 8 discusses the performance indicators of the models and Section 9 concludes. The Appendix gives technical derivations underlying the SVM-Maj algorithm.

2. Definitions

First of all, let us introduce some notations: n denotes the number of object; k denotes the number of predictor variables, $\mathbf{X}$ is an $n \times k$ matrix containing the k predictor variables, without a column of ones to specify the intercept, $\mathbf{y}$ is an $n \times 1$ vector with 1 if object i belongs to class 1 and -1 if it belongs to class -1, r denotes the rank of matrix $\mathbf{X}$, and $\lambda > 0$ is the penalty parameter for the penalty term. The purpose of SVM is to produce a linear combination of predictor variables $\mathbf{X}$ such that aW positive prediction is classified to class $+1$ and a negative prediction to class -1. Let $\boldsymbol{\beta}$ and α be parameters of the linear combination $\alpha + \mathbf{x}_i'\boldsymbol{\beta}$. Then, for a given intercept α and vector with predictor weights $\boldsymbol{\beta}$, the predicted class is given by

$$\hat{y}_i = \text{sign}(\mathbf{x}_i'\boldsymbol{\beta} + \alpha) = \text{sign}(q_i + \alpha) = \text{sign}(\tilde{q}_i), \tag{1}$$

where $q_i = \mathbf{x}_i\boldsymbol{\beta}$. Here, the term $\tilde{q}_i = q_i + \alpha$ is used to indicate the predicted score for object

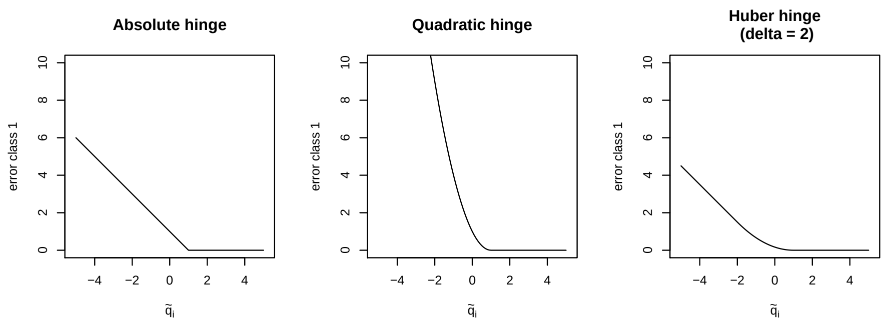

Figure 1: This figure shows the hinge error functions. Note that the error function is only nonzero if $\tilde{q} < 1$.

i, which also accounts for the intercept α.

To find the optimal of the linear combination, the SVM loss function is used as a function of α and $\boldsymbol{\beta}$ to be minimized. The SVM loss function can be represented in several ways. Here, the notation of Groenen *et al.* (2008) is used that allows for general error functions $f_1(\tilde{q}_i)$ and $f_{-1}(\tilde{q}_i)$. The goal of SVMs is to minimize the SVM loss function

$$\begin{array}{llllll} L_{\text{SVM}}(\alpha, \boldsymbol{\beta}) & & & & & \\ = & \sum_{i \in G_1} w_i f_1(\tilde{q}_i) & + & \sum_{i \in G_{-1}} w_i f_{-1}(\tilde{q}_i) & + & \lambda \boldsymbol{\beta}' \boldsymbol{\beta} \\ = & \text{Class 1 errors} & + & \text{Class -1 errors} & + & \text{Penalty for nonzero } \boldsymbol{\beta}, \end{array}$$

over α and $\boldsymbol{\beta}$, where G_1 and G_{-1} respectively denote the sets of class 1 and -1 objects, and w_i is a given nonnegative importance weight of observation i. Note that Groenen *et al.* (2008) proved that minimizing $L_{\text{SVM}}(\alpha, \boldsymbol{\beta})$ is equivalent to minimizing the SVM error function in Vapnik (1995) and Burges (1998). Figure 1 and Table 2 contain different error functions that can be used, such as the classical hinge error standard in SVMs, the quadratic hinge, and the Huber hinge. Figure 1 plots the error functions as function value of $\tilde{q}_i$. All three hinge functions have the property that their error is only larger than zero if the predicted value $\tilde{q}_i < 1$ for class $+1$ objects and $\tilde{q}_i > -1$ for class -1 objects. The classic absolute hinge error is linear in $\tilde{q}_i$ and is thus robust for outliers. However, it is also non-smooth at $\tilde{q}_i = 1$. The quadratic hinge is smooth but may be more sensitive to outliers. The Huber loss is smooth as well as robust. Those observations $\mathbf{x}_i$ yielding zero errors do not contribute to the SVM solution and could therefore be removed without changing the SVM solution. The remaining observations $\mathbf{x}_i$ that do have an error or have $|\tilde{q}_i| = 1$ determine the solution and are called support vectors, hence the name Support Vector Machine. Unfortunately, before the SVM solution is available it is unknown which of the observations are support vectors and which are not. Therefore, the SVM always needs all available data to obtain a solution.

The weight w_i of observation i can be interpreted as the relative importance of the error of the observation. One can also assign the same weights for all observations in G_1 and different weight for those in G_{-1}. Assigning weights per class is especially useful when one class is substantially larger than the other. By assigning a larger weight for the smaller subset, one can correct the dominance of the errors of the larger subset. For example, if there are n_1 objects in G_1 and n_{-1} objects in G_{-1}, then choosing

Error function	$f_1(\tilde{q}_i)$
Absolute hinge	$\max(0, 1 - \tilde{q}_i)$
Quadratic hinge	$\max(0, 1 - \tilde{q}_i)^2$
Huber hinge	$\begin{cases} \dfrac{1}{2(\delta+1)} \max(0, 1-\tilde{q}_i)^2 & \tilde{q}_i > -\delta \\ \dfrac{(\delta-1)}{2} - \tilde{q}_i & \tilde{q}_i \leq -\delta \end{cases}$

Table 2: Common error function used. The further $\tilde{q}_i$ is from wrongly predicted, the higher its error. Note that $f_{-1}(\tilde{q}_i) = f_1(-\tilde{q}_i)$, therefore, only $f_1(\tilde{q}_i)$ is described below.

$$w_i = \begin{cases} (n_1 + n_{-1})/(2n_1) & i \in G_1 \\ (n_1 + n_{-1})/(2n_{-1}) & i \in G_{-1} \end{cases}$$

is one way to obtain equal weighting of the classes.

A more compact expression of L_{SVM} is obtained by exploiting the symmetric relation $f_{-1}(\tilde{q}_i) = f_1(-\tilde{q}_i) = f_1(y_i\tilde{q}_i)$. Then, the SVM loss function can be simplified into

$$\begin{array}{lclcl} L_{\mathrm{SVM}}(\alpha, \boldsymbol{\beta}) & = & \sum_{i=1}^n w_i f_1(y_i\tilde{q}_i) & + & \lambda\boldsymbol{\beta}'\boldsymbol{\beta} \\ & = & \text{Error} & + & \text{Penalty for nonzero } \boldsymbol{\beta}. \end{array} \tag{2}$$

To minimize this function, the SVM-Maj algorithm, discussed in the next section, is used.

3. The SVM-Maj algorithm

The **SVMMaj** package minimizes the SVM loss function by using the SVM-Maj algorithm, that is based on the ideas of majorization (see, e.g., De Leeuw and Heiser 1980; De Leeuw 1994; Heiser 1995; Lange *et al.* 2000; Kiers 2002; Hunter and Lange 2004; Borg and Groenen 2005). Groenen *et al.* (2007, 2008) developed the algorithm for linear SVMs. Here, the SVM-Maj algorithm is extended to nonlinear situations that use a kernel matrix.

The SVM-Maj algorithm uses iterative majorization to minimize the loss function (2). The main point of this algorithm is to iteratively replace the original function $f(x)$ by an auxiliary function $g(x, \overline{x})$ at supporting point $\overline{x}$, for which the minimum can be easily computed. The auxiliary function, called the majorization function, has the properties that:

- the minimum x^* of $g(x, \overline{x})$ can be found easily, and
- $g(x, \overline{x})$ is always larger than or equal to the $f(x)$, and
- at the supporting point $\overline{x}$, $g(\overline{x}, \overline{x})$ is equal to $f(\overline{x})$.

Combining these properties gives the so-called sandwich inequality $f(x^*) \leq g(x^*, \overline{x}) \leq g(\overline{x}, \overline{x}) = f(\overline{x})$. That is, for each support point $\overline{x}$, we can find another point x^* of whose function value $f(x^*)$ is lower or equal to the former $f(\overline{x})$. Using this point x^* as the next iteration's support point and repeating the procedure, a next update can be obtained with a lower $f(x)$ value. This iterative process produces a series of function values that is non-increasing and generally decreasing. Note that this principle of majorization also works if the argument of f is a vector. Moreover, if f is strictly convex, as is the case with the SVM loss function, the updates converge to the minimum of the original function as the number of iterations increases.

The first step is to find the majorizing functions for the different hinge errors. The majorization function of the SVM-Maj algorithm is a quadratic function so that its minimum is obtained by setting the derivative to zero. Given the support point $\overline{x}$, a quadratic majorization function $g(x, \overline{x})$ can be written as

$$g(x, \overline{x}) = a(\overline{x})x^2 - 2b(\overline{x})x + o(\overline{x}). \tag{3}$$

As the parameter $o(\overline{x})$ is irrelevant for determining $\hat{x}$ as it is constant for a given $\overline{x}$, the notation o will be used instead of $o(\overline{x})$ to indicate all terms that are not dependent on x in the majorizing function.

Conform Groenen *et al.* (2007, 2008), $f_1(y_i\tilde{q}_i)$ in (2) can be majorized by $g(\tilde{q}_i, \overline{\tilde{q}}_i, y_i) = a_i\tilde{q}_i^2 - 2b_i\tilde{q}_i + o_i$ with the parameters a_i, b_i, and o_i given in Table 3. As $f_1(y_i\tilde{q}_i) \leq g(\tilde{q}_i, \overline{\tilde{q}}_i, y_i)$, we can obtain the majorization function for the SVM loss function (2), using $\tilde{\mathbf{q}} = \mathbf{X}\boldsymbol{\beta} + \alpha\mathbf{1} = \mathbf{q} + \alpha\mathbf{1}$ and the support point $\overline{\tilde{\mathbf{q}}} = \overline{\alpha}\mathbf{1} + \overline{\mathbf{q}}$

$$\begin{aligned}
\mathrm{L}_{\mathrm{SVM}}(\alpha, \boldsymbol{\beta}) &= \sum_i w_i f_1(y_i\tilde{q}_i) + \lambda\boldsymbol{\beta}'\boldsymbol{\beta} \\
&\leq \sum_i w_i g(\tilde{q}_i, \overline{\tilde{q}}_i, y_i) + \lambda\boldsymbol{\beta}'\boldsymbol{\beta} + o \\
&= \tilde{\mathbf{q}}'\mathbf{A}\tilde{\mathbf{q}} - 2\tilde{\mathbf{q}}'\mathbf{b} + \lambda\boldsymbol{\beta}'\boldsymbol{\beta} + o \\
&= (\alpha\mathbf{1} + \mathbf{q})'\mathbf{A}(\alpha\mathbf{1} + \mathbf{q}) - 2(\alpha\mathbf{1} + \mathbf{q})'\mathbf{b} + \lambda\boldsymbol{\beta}'\boldsymbol{\beta} + o \\
&= \mathrm{Maj}(\alpha, \boldsymbol{\beta}),
\end{aligned} \tag{4}$$

where $\mathbf{A}$ is an $n \times n$ diagonal matrix with elements $w_i a_i$ and $\mathbf{b}$ an $n \times 1$ vector with elements $w_i b_i$ (with a_i and b_i from Table 3). As $w_i > 0$ for all i, we can conclude that the diagonal matrix $\mathbf{A}$ is positive definite, which implies that the majorization function is strictly quadratically convex, thereby guaranteeing a unique minimum of the majorizing function at the right hand side of (4). Groenen *et al.* (2008) showed in their article that the minimum of (4), that is, an update for the iteration process, can be computed as followed:

$$\left(\widetilde{\mathbf{X}}'\mathbf{A}\widetilde{\mathbf{X}} + \lambda\mathbf{J}\right)\begin{bmatrix}\alpha^+ \\ \boldsymbol{\beta}^+\end{bmatrix} = \widetilde{\mathbf{X}}'\mathbf{b}, \tag{5}$$

where $\widetilde{\mathbf{X}} = \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}$ and $\mathbf{J} = \begin{bmatrix}0 & \mathbf{0}' \\ \mathbf{0} & \mathbf{I}\end{bmatrix}$. Appendix A shows the derivation of this update, while Algorithm 1 shows the steps taken by the SVM-Maj algorithm.

3.1. Computational efficiencies

Before I continue with the discussion of the extended options for the SVM-Maj algorithm, a few modificationsof the SVM-Maj algorithm will be introduced to obtain computational

Error term	Parameters
Absolute hinge	$a_i = \frac{1}{4}\max(\lvert 1 - y_i\bar{\bar{q}}_i\rvert, \epsilon)^{-1}$
	$b_i = y_i a_i(1 + \lvert y_i\bar{\bar{q}}_i - 1\rvert)$
	$o_i = a_i(1 + \lvert y_i\bar{\bar{q}}_i - 1\rvert)^2$
Quadratic hinge	$a_i = 1$
	$b_i = y_i a_i(1 + \max(y_i\bar{\bar{q}}_i - 1, 0))$
	$o_i = a_i(1 + \max(y_i\bar{\bar{q}}_i - 1, 0))^2$
Huber hinge	$a_i = \frac{1}{2}(k+1)^{-1}$
	$b_i = y_i a_i(1 + \max(y_i\bar{\bar{q}}_i - 1, 0) + \min(y_i\bar{\bar{q}}_i + k, 0))$
	$o_i = a_i(1 + \max(y_i\bar{\bar{q}}_i - 1, 0) + \min(y_i\bar{\bar{q}}_i + k, 0))^2 + \min(y_i\bar{\bar{q}}_i + k, 0)$

Table 3: Parameter values of a_i, b_i and o_i of the majorization function.

efficiencies. One efficiency can be achieved by using the QR-decomposition on matrix $\mathbf{X}$ to find a more efficient update. Another efficiency improvement can be obtained in case of using the quadratic or Huber hinge and the number of updates generally decreases when using a *relaxed update*. The algorithm is summarized in Algorithm 1 and the most important steps are discussed below and in the appendix.

Efficient updates by using QR-decomposition

Usually, the loss function will be optimized by optimizing $\boldsymbol{\beta}$. However, when $k > n$, it is more efficient to optimize $\mathbf{q}$ instead, as it has a lower dimensionality. In this situation, the dimensional space in which the optimal parameter values lies will be smaller and therefore it is more efficient and generally faster to compute an update. In case that $r = \text{rank}(\mathbf{X}) < \min(n, k)$, an even more efficient update exists. Moreover, in a higher dimensional space, one only needs a part of the space of $\boldsymbol{\beta}$ to find the optimal $\mathbf{q}$. The appendix discusses these issues more in detail and introduces efficient and consistent updates for SVM-Maj for all possible situations. An overview of all efficient updates in each occasion can be found in Table 4.

Equation 5 optimizes $\boldsymbol{\beta}$ to optimize the loss function. However, in some cases, there exist an even more efficient update. Appendix A derives an efficient update for each of these cases by making use of the singular value decomposition (SVD)

$$\underset{(n\times k)}{\mathbf{X}} = \underset{(n\times r)}{\mathbf{P}} \quad \underset{(r\times r)}{\boldsymbol{\Lambda}} \quad \underset{(r\times k)}{\mathbf{Q}'}, \tag{6}$$

where $\mathbf{P}'\mathbf{P} = \mathbf{I}$ and $\mathbf{Q}'\mathbf{Q} = \mathbf{I}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix. However, **SVMMaj** package uses the QR-decomposition to determine the rank of $\mathbf{X}$, as it is a computationally more efficient way to determine the rank of $\mathbf{X}$ than doing so through the SVD-decomposition. Let the QR-decomposition of $\mathbf{X}'$ be given by $\mathbf{X}' = \mathbf{V}\mathbf{Z}'$ with $\mathbf{V}'\mathbf{V} = \mathbf{I}$ and $\mathbf{Z}$ a lower triangular matrix. An additional QR-decomposition of $\mathbf{Z}$ gives $\mathbf{Z} = \mathbf{U}\mathbf{R}$ with $\mathbf{U}'\mathbf{U} = \mathbf{I}$ and $\mathbf{R}$ an upper

```
Algorithm: SVM-Maj
input  : $\mathbf{y} = n \times 1$ vector with class labels $+1$ and $-1$ ,
         $\mathbf{X} = n \times k$ matrix of predictor variables ,
         $\lambda > 0$ the penalty parameter,
         $Hinge$ = {absolute,quadratic,huber} the hinge error function,
         $\delta > 0$ the Huber hinge parameter,
         $relax$ determining from which step to use the relaxed update (12),
         $method$ specifies whether $\boldsymbol{\rho}$ (using matrix decomposition) will be used or $\boldsymbol{\beta}$.
output: $\alpha_t$, ($\boldsymbol{\rho}_t$ or $\boldsymbol{\beta}_t$)

$t = 0$;
Set $\epsilon$ to a small positive value;
Set ($\boldsymbol{\rho}_0$ or $\boldsymbol{\beta}_0$) and $\alpha_0$ to random initial values;
Compute $L_0 = L_{\mathrm{SVM}}(\alpha_0, \boldsymbol{\rho}_0)$ or $L_{\mathrm{SVM}}(\alpha_0, \boldsymbol{\beta}_0)$ according to (2);
if $hinge \neq absolute$ & $method = \boldsymbol{\beta}$ then
    Find $\mathbf{S}$ that solves (9);
else if $hinge \neq absolute$ & $method = \boldsymbol{\rho}$ then
    Find $\mathbf{S}$ that solves (10);
end
while $t = 0$ or $(L_{t-1} - L_t)/L_t > \epsilon$ do
    $t = t + 1$;
    Compute $a_i$ and $b_i$ by Table 3;
    Set diagonal elements of $\mathbf{A}$ to $w_i a_i$ and $\mathbf{b}$ to $w_i b_i$;
    Comment:Compute update
    if $hinge = absolute$ & $method = \boldsymbol{\rho}$ then
        Find $\alpha_t$ and $\boldsymbol{\rho}_t$ that solves (8): $\left(\widetilde{\mathbf{Z}}'\mathbf{A}\widetilde{\mathbf{Z}} + \lambda\mathbf{J}\right)\begin{bmatrix}\alpha_t \\ \boldsymbol{\rho}_t\end{bmatrix} = \widetilde{\mathbf{Z}}'\mathbf{b}$ ;
    else if $hinge = absolute$ & $method = \boldsymbol{\beta}$ then
        Find $\alpha_t$ and $\boldsymbol{\rho}_t$ that solves (5): $\left(\widetilde{\mathbf{X}}'\mathbf{A}\widetilde{\mathbf{X}} + \lambda\mathbf{J}\right)\begin{bmatrix}\alpha_t \\ \boldsymbol{\beta}_t\end{bmatrix} = \widetilde{\mathbf{X}}'\mathbf{b}$ ;
    else if $hinge \neq absolute$ & $method = \boldsymbol{\rho}$ then
        Find $\alpha_t$ and ($\boldsymbol{\rho}_t$ or $\boldsymbol{\beta}_t$) that solves: $\begin{bmatrix}\alpha^+ \\ \boldsymbol{\rho}^+\end{bmatrix} = \mathbf{Sb}$
    else if $hinge \neq absolute$ & $method = \boldsymbol{\beta}$ then
        Find $\alpha_t$ and ($\boldsymbol{\rho}_t$ or $\boldsymbol{\beta}_t$) that solves: $\begin{bmatrix}\alpha^+ \\ \boldsymbol{\beta}^+\end{bmatrix} = \mathbf{Sb}$;
    end
    if $t \geq relax$ then
        Replace $\alpha_t = 2\alpha_t - \alpha_{t-1}$;
        Replace $\boldsymbol{\rho}_t = 2\boldsymbol{\rho}_t - \boldsymbol{\rho}_{t-1}$ or $\boldsymbol{\beta}_t = 2\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}$;
    end
    Compute $L_t = L_{\mathrm{SVM}}(\alpha_t, \boldsymbol{\rho}_t)$ or $L_{\mathrm{SVM}}(\alpha_t, \boldsymbol{\beta}_t)$ according to (2);
end
```

Algorithm 1: The SVM majorization algorithm SVM-Maj

Method	Situation
$\boldsymbol{\beta}$	$n \geq k, r = k$
$\mathbf{q}$	$n \leq k, r = n$
$\boldsymbol{\theta}$	$r < \min(n, k)$

Table 4: An overview of the most efficient updates for each situation.

triangular matrix. Then we have

$$\underset{(n \times k)}{\mathbf{X}} = \underset{(n \times r)}{\mathbf{U}} \; \underset{(r \times r)}{\mathbf{R}} \; \underset{(r \times k)}{\mathbf{V}'}. \tag{7}$$

Note that matrices $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and and therefore have the same properties as $\mathbf{P}$ and $\mathbf{Q}$. Thus, we can replace matrices $\mathbf{P}$, $\boldsymbol{\Lambda}$ and $\mathbf{Q}$ by respectively $\mathbf{U}$, $\mathbf{R}$ and $\mathbf{V}$. The update (5) can then be computed through

$$(\tilde{\mathbf{Z}}'\mathbf{A}\tilde{\mathbf{Z}} + \lambda\mathbf{J}) \begin{bmatrix} \alpha^+ \\ \boldsymbol{\rho}^+ \end{bmatrix} = \tilde{\mathbf{Z}}'\mathbf{b}, \tag{8}$$

where $\tilde{\mathbf{Z}} = \begin{bmatrix} \mathbf{1} & \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{UR} \end{bmatrix}$ and where $\boldsymbol{\rho} = \mathbf{V}'\boldsymbol{\beta}$. Furthermore, $\boldsymbol{\beta}$ and $\mathbf{q}$ can be derived from $\boldsymbol{\beta} = \mathbf{V}\boldsymbol{\rho}$ and $\mathbf{q} = \mathbf{Z}\boldsymbol{\rho} = \mathbf{UR}\boldsymbol{\rho}$. Note that in most cases, the decomposition of $\mathbf{Z}$ is not necessary to perform, so that only a single QR-decomposition performance is needed. As this decomposition is already performed to determine the rank of $\mathbf{X}$, it is efficient to use update (8) in all three cases distinguished in Table 4. Nevertheless, in case n and k are both very large, it may be more efficient not to perform a matrix decomposition at all to avoid unnecessary computations. Instead, one can use update (5).

Quadratic and Huber hinge

For the quadratic and Huber hinge, the parameters a_i does not depend on the support point $\overline{\mathbf{q}}$, which means that the matrix $\mathbf{A}$ remain fixed during the algorithm steps. In fact, the parameters a_i are the same for all i, that is, $a_i = a$ for all i. Therefore, extra computational efficiency can be obtained by solving the linear system

$$(a\widetilde{\mathbf{X}}'\widetilde{\mathbf{X}} + \lambda\mathbf{J})\mathbf{S} = \widetilde{\mathbf{X}}', \tag{9}$$

or, when using QR-decomposition,

$$(a\widetilde{\mathbf{Z}}'\widetilde{\mathbf{Z}} + \lambda\mathbf{J})\mathbf{S} = \widetilde{\mathbf{Z}}', \tag{10}$$

so that the original update (8) can be simplified into

$$\begin{bmatrix} \alpha^+ \\ \boldsymbol{\rho}^+ \end{bmatrix} = \mathbf{Sb}, \tag{11}$$

so that in each iteration the update can be obtained by a single matrix multiplication instead of solving a linear system.

Computational efficiency by the relaxed updates

The parameter updates obtained by (8) find the minimum of the majorization function in (4). This guarantees that the next update will always be better than the previous point. However, it turns out that often the updates converge faster when making the update twice as long by using a relaxed update (De Leeuw and Heiser 1980):

$$\boldsymbol{\theta}^*_{t+1} = \boldsymbol{\theta}_{t+1} + (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) = 2\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t. \tag{12}$$

This relaxation is most effective when SVM-Maj has already performed several iterations. Preliminary experimentation revealed that this is the case when at least 20 iterations without relaxed updates have been performed. Therefore, the relaxation (12) is implemented after 20 iterations to increase model estimation efficiency.

4. Nonlinearities

In the previous sections, we have only discussed the SVM-Maj algorithm for the linear case. However, often a better prediction can be obtained by allowing for nonlinearity of the predictor variables. Therefore, one might consider to use nonlinearity in the model estimation. In the **SVMMaj** package, two different implementation of nonlinearity can be used: I-splines and kernels. One can choose one of these implementation, or both of them.

Splines are piecewise polynomial functions of a specified variable. The **SVMMaj** package can transform each predictor variable into I-splines (Ramsay 1988). This transformation will split the original predictor variable $\mathbf{x}_j$ into a number of spline bases vectors gathered in the matrix $\mathbf{B}_j$. After specifying the interior knots k_s that define the boundaries of the pieces and the degree d_s of the polynomials, one can transform the variable $\mathbf{x}_j$ into the basis $\mathbf{B}_j$ of a size of $n \times (d_s + k_s)$. This spline basis $\mathbf{B}_j$ will then be used as a set of $(d_s + k_s)$ variables to find a linear separating plane of a higher dimension. The piecewise polynomial transformation of variable $\mathbf{x}_j$ can then be obtained through computing the linear combination of the spline bases $\mathbf{B}_j\boldsymbol{\gamma}_j$ with the initially unknown weights $\boldsymbol{\gamma}_j$. Then, all spline bases $\mathbf{B}_j$ and weights $\boldsymbol{\gamma}_j$ are gathered in $\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_k]$ and $\boldsymbol{\beta}' = [\boldsymbol{\gamma}_1', \boldsymbol{\gamma}_2', \ldots, \boldsymbol{\gamma}_k']$. Note that here the I-splines are used as they have the property that if multiplied by positive weights, there is a guaranteed monotone relation with the original variable $\mathbf{x}_j$. This property can aid the interpretation of the weights as $\boldsymbol{\beta}$ can be split into a vector of positive and one of negative weights.

It is also possible to map the matrix $\mathbf{X}$ differently into a higher dimensional space through so-called kernels. Let us map the predictor variables of observation i, that is, $\mathbf{x}_i$ into $\phi(\mathbf{x}_i)$ with mapping function $\phi : \Re^k \rightarrow \Re^m$. Furthermore, let us denote $k_{ij} = \boldsymbol{\phi}(\mathbf{x}_i)'\boldsymbol{\phi}(\mathbf{x}_j)$ as the inner product of the transformed vectors of $\mathbf{x}_i$ and $\mathbf{x}_j$. Then, the *kernel matrix* $\mathbf{K}$ is denoted as the inner product matrix with value k_{ij} in row i and column j. Note that the kernel matrix always is of size $n \times n$ and $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}'$ with row i equal to $\phi(\mathbf{x}_i)'$. Using this property, we can summarize the mapping of $\mathbf{X}$ even when $m \rightarrow \infty$ into a matrix of finite size, even if $m \rightarrow \infty$. This method is also known as the 'kernel trick'.

We will now show that this kernel matrix $\mathbf{K}$ can be used to find an efficient majorization update. Then the matrix $\boldsymbol{\Phi}$ is used instead of $\mathbf{X}$ to derive the majorization update. Let us

Type of Kernel	Kernel Function $\phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$
linear	$\mathbf{x}_i'\mathbf{x}_j$
homogeneous polynomial	$(\text{scale}\,\mathbf{x}_i'\mathbf{x}_j)^{\text{degree}}$
nonhomogeneous polynomial	$(\text{scale}\,\mathbf{x}_i'\mathbf{x}_j + 1)^{\text{degree}}$
radial basis function	$\exp(-\text{sigma}\vert\mathbf{x}_i - \mathbf{x}_j\vert^2)$
Laplace	$\exp(-\text{sigma}\vert\mathbf{x}_i - \mathbf{x}_j\vert)$

Table 5: Table of some examples of kernel functions, which can be used in the **SVMMaj** package.

perform the QR-decomposition on $\mathbf{\Phi}$ analogous to (7), that is

$$\underset{(n\times m)}{\mathbf{\Phi}} = \underset{(n\times r)}{\mathbf{U}}\ \underset{(r\times r)}{\mathbf{R}}\ \underset{(r\times m)}{\mathbf{V}'} = \underset{(n\times r)}{\mathbf{Z}}\ \underset{(r\times m)}{\mathbf{V}'}, \tag{13}$$

where r denotes the rank of $\mathbf{\Phi}$ satisfying $r \leq \min(n, m) = n \ll \infty$. Note that the update (8) does not require $\mathbf{V}$. Moreover, the relation between $\mathbf{\Phi}$ and $\mathbf{K}$ can be given by $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}'$. Using decomposition (13) yields

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}' = (\mathbf{Z}\mathbf{V}')(\mathbf{V}\mathbf{Z}') = \mathbf{Z}\mathbf{Z}'. \tag{14}$$

As $\mathbf{Z}$ is a lower triangular matrix, it can be obtained by performing a Cholesky decomposition on $\mathbf{K}$. Therefore, without actually computing the mapped space $\mathbf{\Phi}$, it is still possible to perform SVM-Maj by using the kernel matrix $\mathbf{K}$.

There is a wide variety of available kernels to obtain nonlinearity of the predictors. Table 5 shows some important examples of often used kernel functions.

As $\mathbf{V}$ is usually unknown, $\boldsymbol{\beta}$ cannot be calculated. Nevertheless, it is still possible to predict the class labels of an unseen sample $\mathbf{X}_2$. Using $\mathbf{q} = \mathbf{\Phi}\boldsymbol{\beta}$ and (13), we can derive

$$\begin{aligned}\boldsymbol{\beta} &= \mathbf{V}\boldsymbol{\rho}\\ &= \mathbf{V}(\mathbf{R}'\mathbf{U}'\mathbf{U}(\mathbf{R}')^{-1})\boldsymbol{\rho}\\ &= (\mathbf{V}\mathbf{R}\mathbf{U}')\mathbf{U}(\mathbf{R}')^{-1}\boldsymbol{\rho}\\ &= \mathbf{\Phi}'\mathbf{U}(\mathbf{R}')^{-1}\boldsymbol{\rho}.\end{aligned}$$

The predicted values of an unseen test sample $\mathbf{X}_2$ are

$$\mathbf{q}_2 = \mathbf{\Phi}_2\boldsymbol{\beta} = \mathbf{\Phi}_2\mathbf{\Phi}'\mathbf{U}(\mathbf{R}')^{-1}\boldsymbol{\rho} = \mathbf{K}_2\mathbf{U}(\mathbf{R}')^{-1}\boldsymbol{\rho}, \tag{15}$$

where $\mathbf{\Phi}_2$ and $\mathbf{\Phi}$ are denoted as the transformed matrix of respectively $\mathbf{X}_2$ and $\mathbf{X}$ into the high dimensional space and $\mathbf{K}$ denotes the kernel matrix $\mathbf{\Phi}_2\mathbf{\Phi}'$.

5. The SVMMaj package in R

The SVM-Maj algorithm for the Support Vector Machine (SVM) is implemented in the **SVM-Maj** package in R. Its main functions are `svmmaj`, which estimates the SVM, and `svmmajcrossval`, which performs a grid search of k-fold cross validations using SVM-Maj to find the combination of input values, (such as λ and degree in the case of a polynomial kernel) giving the best prediction performance.

The `svmmaj` function requires the $n \times k$ predictor matrix `X` and the $n \times 1$ vector `y` with class labels. Apart from the data objects, other parameter input values can be given as input to tune the model: `lambda`, `hinge`, `weights.obs`, `scale` and parameters for nonlinearities and settings of the algorithm itself. Table 6 shows the arguments of function `svmmaj` and its default values. For example,

```
svmmaj(X, y, lambda = 2, hinge = "quadratic", scale = "interval")
```

runs the SVM model with $\lambda = 2$, using a *quadratic* hinge and for each attribute, the values are scaled to the interval [0,1].

The function `svmmajcrossval` uses the same parameter input values and additionally the parameters to be used as grid points of the k-fold cross validation. These parameters should be given in the list object `search.grid`, e.g.,

```
svmmajcrossval(X, y, search.grid = list(lambda = c(1, 2, 4)))
```

performs a cross validation of `X` and `y` with as grid points $\lambda = 1, 2, 4$.

As an example, the `AusCredit` data set of the `libsvm` data sets (Chang and Lin 2001) is used, which is included in the **SVMMaj** package. This data set consists of in total 690 credit requests, 307 of which are classified as positive and 383 as negative. These classifications are stored in `AusCredit$y` with class label `Rejected` to represent the negative responses, and label `Accepted` for the positive responses. In total, 14 predictor variables of each applicant has been stored as predictor variables in `AusCredit$X`. Due to confidentiality, the labels of all predictor variables are not available. Moreover, the observations in the data set `AusCredit` is split into `AusCredit.tr`, consisting the first 400 observations, and `AusCredit.te`, with the remaining 290 observations. `AusCredit.tr` will be used to estimate the model, while the `AusCredit.te` is used to analyze the prediction performances.

Example 1 *Train the SVM-model using the data set of Australian credit requests to classify the creditability of the applicant.*

The command

```
R> library(SVMMaj)
```

loads the **SVMMaj** package into R. In this example, I will use the components `X` and `y` in the training set `AusCredit.tr`, which consists of predictor variables respectively class labels of 400 credit requests, to train the model. Afterwards, the characteristics of the remaining 290 requests, which has been stored into component `X` of test set `AusCredit.te`, are used to predict the classification of the applicant. Using the actual class labels, `AusCredit.te$y`, the out of sample prediction performance is analyzed by comparing the ones with the predicted using the SVM model as estimated with `AusCredit.tr`. Running the SVM on the training data is done as follows.

Parameter	Values	Description
`lambda`	scalar	Nonnegative penalty parameter of the penalty term. Default value is 1.
`hinge`		Specifies the hinge error term:
	`'absolute'`*	= Use the absolute hinge error.
	`'quadratic'`	= Use the quadratic hinge error.
	`'huber'`	= Use the Huber hinge error.
`weights.obs`	vector	Vector with the nonnegative weight for the residual of each object or class, which can either be a vector of length 2 to specify the weights for each class, or a vector of length `length(y)` giving the weights for each object i. Default value is `c(1,1)`
`scale`	`'zscore'`	=Rescales all variables in $\mathbf{X}$ to have zero mean and standard deviation equal to 1.
	`'interval'`	=Rescales all variables in $\mathbf{X}$ to be in the range $[0,1]$.
	`'none'`*	=No scaling is done.
`decomposition`	logical	In case of a linear kernel, `decomposition=FALSE` will not perform a decomposition at all and uses update 5 instead.
`spline.knots`	integer	Number of interior knots of the spline which is equal to one less than the number of adjacent intervals. Default value is 0, which corresponds with a single interval.
`spline.degree`	integer	Degree of the spline basis. Default value is 1, representing the linear case.
`kernel`		The kernel function of package `kernlab`, which specifies which kernel type to be used. Possible kernels and its parameter values are amongst others:
	`vanilladot`	= The linear kernel: `k(x,x') = <x,x'>`.
	`polydot`	= The polynomial kernel: `k(x,x') = (scale <x,x'> + offset) ^ degree`.
	`rbfdot`	= The Gaussian Radial Basis Function (RBF) kernel: `k(x,x') = exp(-sigma |x - x'|) ^ 2`.
	`laplacedot`	= The Laplacian kernel: `k(x,x') = exp(-sigma |x - x'|)` .
`kernel.sigma`	1	Kernel parameter `sigma` used in the RBF and Laplacian kernel.
`kernel.degree`	1	Kernel parameter `degree` used in the polynomial kernel.
`kernel.scale`	1	Kernel parameter `scale` used in the polynomial kernel.
`kernel.offset`	0	Kernel parameter `offset` used in the polynomial kernel.
`convergence`	double	Specifies the convergence criterion of the algorithm. Default value is `3e-7`
`print.step`	logical	`print.step=TRUE` shows the progress of the iteration. Default value is `FALSE`.
`initial.point`	vector	Initial values $[\alpha_0\ \boldsymbol{\rho}_0]$ (see Algorithm 1) to be used in the SVM-Maj algorithm, the length of the vector equals the rank of predictor matrix `X` plus one.
`increase.step`	integer	The iteration level from where the relaxed update (12) will be used. Default value is 20.
`na.action`	`na.action`	Specifies which `na.action` object will be used to handle missing values. Default is omit observations with missing values.

Table 6: Parameter input values of `svmmaj`. The default settings are marked with *

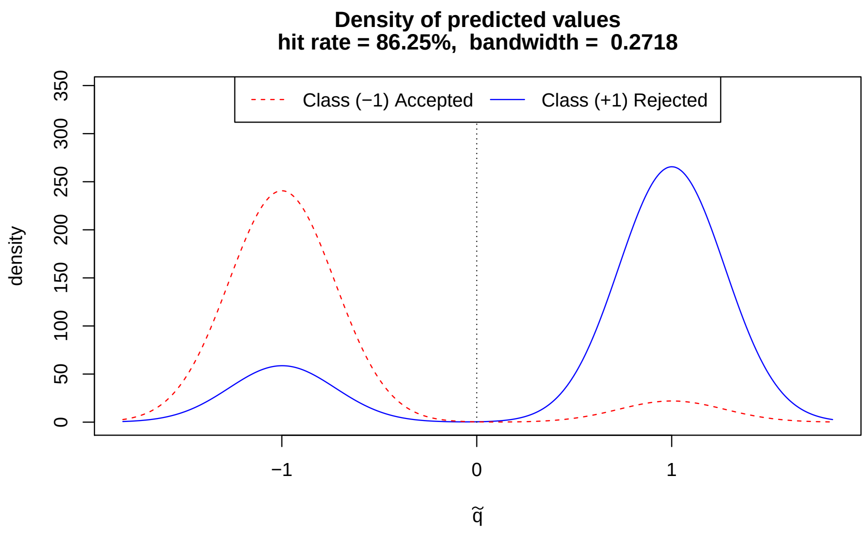

Figure 2: This figure shows the distribution of predicted values $\tilde{\bar{q}}$ of the two classes of Example 1, which can be obtained through the `plot()` method. These densities are created by the `density` function. This function specifies beforehand the bandwidth value to plot the density, which is shown on top of the graph.

```
R> model <- svmmaj(AusCredit.tr$X, AusCredit.tr$y)
R> model

Model:
    update method                      QR
    number of iterations               312
    loss value                         114.0001
    number of support vectors          121
```

As a result, the trained model will be returned as an `svmmaj`-object. The `print` method of this object shows which update method is used, the number of iterations before convergence, the found minimum loss value and the number of support vectors. In case no kernel is used, the matrix **Z** from update (8) is obtained through the QR-decomposition shown in (7) by default. In case a nonlinear kernel is used, **Z** is being calculated through the Cholesky decomposition (14). One can choose not to perform a decomposition when using a nonlinear kernel by specifying `decomposition = FALSE`. Then the original update (5) will be used. A graph of the distribution of the predicted values $\tilde{\mathbf{q}}$ for each class can be plotted via the `plot()` method using the `density` function of R, see Figure 2. The distribution shows that the majority of the `Accepted` class (-1) respondents received predicted $\tilde{q}_i$ close to -1 and only a few close to $+1$. The same holds for the `Rejected` class $(+1)$ respondents showing that the majority of respondents are correctly classified. A more detailed description of the model can be requested by using the `summary()` method.

```
R> summary(model)
```

```
Call:
   svmmaj.default(X = AusCredit.tr$X, y = AusCredit.tr$y)

Settings:
    lambda                            1
    hinge error                       absolute
    spline basis                      no
    type of kernel                    linear

Data:
    class labels                      Accepted Rejected
    rank of X                         14
    number of predictor variables     14
    number of objects                 400
    omitted objects                   0

Model:
    method                            QR
    number of iterations              312
    loss value                        114.0001
    number of support vectors         121

Confusion matrix:
            Predicted(yhat)
Observed (y) Accepted Rejected Total
    Accepted      164       15   179
    Rejected       40      181   221
    Total         204      196   400

Classification Measures:

   hit rate                          0.863
   weighted hit rate                 0.863
   misclassification rate            0.137
   weighted missclassification rate  0.137

                   TP        FP Precision
    Accepted     0.916    0.0838     0.804
    Rejected     0.819    0.1810     0.923
```

The Settings segment describes the parameter settings used to estimate the model. In this example, the scales of the predictor variables have not been changed and a linear model is specified because no spline or nonlinear kernel is used. Furthermore, the penalty term of the loss function consists of an absolute hinge, with a penalty parameter λ of 1. In the

`Data` segment, the properties of the input data are shown: the labels of each class, the rank of the predictor variable matrix **X** (in case of using I-splines, this will be the rank of the resulting spline bases) and the size of the data (the number of objects and number of predictor variables). **SVMMaj** has the possibility to handle missing values through a specified `na.action`-object. In case observations with missing values are omitted, the number of omitted observations will also be printed in this segment. The `Model` segment summarizes the trained model as a result of the SVM-Maj algorithm: it specifies which update has been used, the number of iterations needed to obtain this model, the optimal loss value and the number of support vectors. The classification performance of the model on the data used to estimate can be found in the last segment, `Classification table`, where the confusion matrix is followed by measured *true positive (TP)*, *false positive (FP)* and *precision* below. True positive of a class denotes the proportion of objects of that class that are being predicted correctly, whereas false positive denotes proportion of the incorrectly predicted objects of a class. The precision of a class is the proportion of correctly predicted objects of the class among all objects predicted to be classified as that class.

Next, we would like to test how well the estimated SVM model predicts an unseen sample: the 290 objects in `AusCredit.te$X` is used as hold-out sample. This is done through the `predict()` method.

```
R> predict(model, AusCredit.te$X)

Prediction frequencies:
          Accepted Rejected
frequency      157      133
```

If the actual class labels are known, one can include this object in the method to show the prediction performance:

```
R> predict(model, AusCredit.te$X, AusCredit.te$y)

Prediction frequencies:
          Accepted Rejected
frequency      157      133

Confusion matrix:
            Predicted(yhat)
Observed (y) Accepted Rejected Total
    Accepted      120        8   128
    Rejected       37      125   162
    Total         157      133   290

Classification Measures:

   hit rate                          0.845
   misclassification rate            0.155
```

```
                TP       FP Precision
Accepted     0.938   0.0625     0.764
Rejected     0.772   0.2284     0.940
```

The classification measures of this unseen sample prediction are similar to these of the in-sample predictions, which mean that this model has no problem of overfitting. Moreover, average hit rate of 85% indicates that this model predicts the objects quite well. Note the difference in true positive value between the classes; it appears that this model predicts classes in class `Accepted` slightly better than the other class.

To show a the distribution of $\hat{\mathbf{q}}$ for all applicants or, if the actual class labels are given, for each class, the argument `show.plot=TRUE` can be included, which result into a figure as in Figure 3.

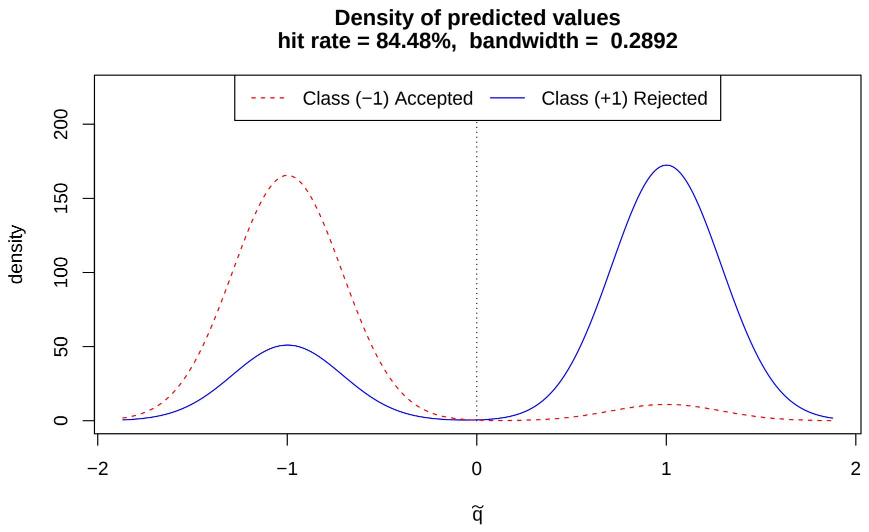

Figure 3: Densities of the predictions in $\tilde{\mathbf{q}}$ split by class.

As this model does not contain any nonlinearity and $\mathbf{X}$ is nonstandardized, $\tilde{q}_i$ for the holdout sample can also be directly computed using the weights `model$beta` and constant `model$theta[1]` found in the model and multiply this with the predictor variables of the unseen sample `AusCredit.te$X`, after being coerced into a `matrix` object.

```
R> alpha <- model$theta[1]
R> beta  <- model$beta
R> qu   <-  drop(alpha + data.matrix(AusCredit.te$X) %*% beta)
```

The predicted classes are then easily obtained by computing.

```
R> y <- factor(qu < 0, levels = c(TRUE, FALSE), labels = model$classes)
```

5.1. Cross validation

Until now, the default settings have been used. However, it is recommended to experiment with different parameters to obtain an optimal prediction model, in particular by varying the penalty parameter λ. To determine the optimal parameter values to be used in further analysis, one can use the `svmmajcrossval` function to perform cross validation with different parameter values. To show an example of using cross validation to determine λ, consider the `voting` data set the `libsvm` data sets (Chang and Lin 2001). This data set corresponds with 434 members of the U.S. House of Representatives Congressmen consisting of 167 republicans and 267 democrats. As the predictor variables, for each of the 16 different political propositions, the votes of the politicians are registered. Using the SVM, we are trying to predict the political wing of the last 134 members using their 16 votes on the propositions as predictor variables. The first 300 members are used as a training sample. A five-fold cross validation is performed on these 300 members in the training sample, with a fine grid of lambda values $\lambda = 10^{-6}, 10^{-5.5}, \ldots, 10^{5}, 10^{5.5}, 10^{6}$. Amongst these lambda values, the optimal λ value is the one which results in the lowest misclassification rate.

Example 2 *Performing cross validation using the voting data sets to find an optimal value for* `lambda`*.*

```
R> library(SVMMaj)
```

In this example, we will use the data in `voting.tr` to perform fivefold-cross validation to determine the optimal `lambda`. Then, this `lambda` is used in an SVM analysis on the entire data set `voting.tr`. Subsequently, this model is used to predict the classification of the unseen sample `voting.te`. Then, `voting.te$X` can be used to compare the prediction with the actual classification `voting.te$y`. This procedure can be executed in R using the following commands:

```
R> results.absolute <- svmmajcrossval(voting.tr$X, voting.tr$y,
+    search.grid = list(lambda = 10^seq(6, -6, length.out = 25)),
+    hinge = "absolute", convergence = 1e-4)
R> model <- svmmaj(voting.tr$X, voting.tr$y, hinge = "absolute",
+    lambda = results.absolute$param.opt$lambda)
R> q.absolute <- predict(model, voting.te$X, voting.te$y)
```

5.2. Hinge functions

SVMMaj allows using the quadratic hinge or the Huber hinge instead of the standard absolute hinge. An important advantage of the quadratic hinge is that it has the potential to be computationally much more efficient than the absolute hinge. Let us perform a similar cross validation on the `voting` data set using the quadratic hinge by

```
R> results.quadratic <- svmmajcrossval(voting.tr$X, voting.tr$y,
+    search.grid = list(lambda = 10^seq(6, -6, length.out = 25)),
+     hinge = "quadratic", convergence = 1e-4)
R> model <- svmmaj(voting.tr$X, voting.tr$y, hinge = "quadratic",
+    lambda = results.quadratic$param.opt$lambda)
R> q.quadratic <- predict(model, voting.te$X, voting.te$y)
```

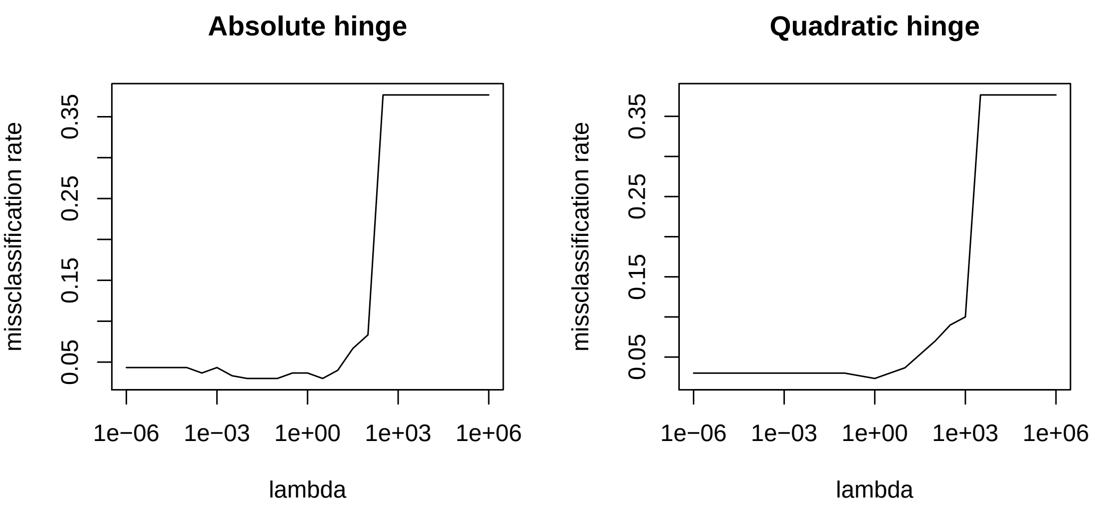

Figure 4: The function `svmmajcrossval` performs a fivefold cross validation of different `lambda` values of both absolute hinge (left panel) and the quadratic hinge (right panel). This figure shows the misclassification rate of different `lambda` values.

A summary of the results of these examples can be found in Table 7. Figure 4 shows the hit rate of the cross validation using different lambda values. It can be seen that both hinges have similar out-of-sample predictive power. It is also clear that per iteration, the quadratic hinge is much faster than the absolute hinge. The effect of the increased computational efficiency of the quadratic hinge is in this example canceled by a large increase of the number of iterations as it requires more iterations to converges.

5.3. Nonlinearities

The package **SVMMaj** can also implement nonlinearity in the model. One can choose to use I-splines, kernel matrices, or both methods to specify the nonlinearity. In the latter case, **SVMMaj** will first convert the explanatory matrix `X` into spline-basis and subsequently generate the kernel matrix of the spline-basis. An advantage of using I-splines over kernels is that on can easily plot the effect of one variable on the predicted value $\hat{q}_i$. In the following example, we will use the `diabetes` dataset of `libsvm` data sets (Chang and Lin 2001).

Example 3 *Train a model with nonlinearities on a data set of 786 females of Pima Indian heritage, predicting the presence of diabetes using several demographic and medial variables.*

In this example, we will use the nonlinear options of `svmmaj` to train a model consisting of 768 female at least 21 years old of Pima Indian heritage, of which 268 are being tested positive for diabetes. In this model, we use 8 variables of state-of-health measures to classify these patients as positive for diabetes or negative. As the number of persons with a positive test result is smaller than the ones with negative results, the loss term of the patients belonging to the former group will be weighted twice as heavy by the extra argument `weights.obs=list(positive=2,negative=1)` to indicate the double weight on the second

	Absolute	Quadratic
CPU-time (sec)	11.73	12.89
Mean no. of iter	94.96	1146.20
CPU-time (per iter)	3.08	0.28
Optimal p	-0.50	0.50
Hit rate(average TP)	0.95	0.94

Table 7: Results of the fivefold cross validation estimation using the `svmcrossval` function. CPU-time is the time in CPU seconds needed to perform fivefold cross validation and Optimal p the value of which 10^p returns the best hit rate in the cross validation. Mean no. of iterations denotes the average value of the sum of the number of iterations per `lambda`-value. CPU-time per iter is the mean computation time needed to perform one iteration. Hit rate is the average TP value by predicting the class labels of the holdout sample of 134 congress men using the first 300 congress men as sample of estimation.

group. As the rank of the explanatory matrix $\mathbf{X}$ is expected to be large, we will use quadratic hinge to make use of the computational efficiency discussed before.

I-Spline

One way of applying nonlinearities in the model is using splines. In this example, we will transform each variable into spline basis of 5 knots and a degree 2, yielding a rank of $8 \times (5 + 2) = 56$, that is, $k = 8$ times the number of columns per spline basis (5 interior knots plus the degree of 2). Five-fold cross validation is used with a grid of $\lambda = 10^{-6}, 10^{-5}, ..., 10^{4}, 10^{5}, 10^{6}$ to determine the optimal λ value.

```
R> results.spline <- svmmajcrossval(diabetes.tr$X, diabetes.tr$y,
+   scale = "interval", search.grid = list(lambda = 10^seq(6, -6)),
+   hinge = "quadratic", spline.knots = 5, spline.degree = 2,
+   weights.obs = list(positive = 2, negative = 1))
R> model.spline <- svmmaj(diabetes.tr$X, diabetes.tr$y,
+  scale = "interval", lambda = results.spline$param.opt$lambda,
+  spline.knots = 5, spline.degree = 2, hinge = "quadratic",
+  weights.obs = list(positive = 2, negative = 1))
R> predict(model.spline, diabetes.te$X, diabetes.te$y,
+  weights = list(positive = 2, negative = 1))

Prediction frequencies:
          negative positive
frequency       92       76

Confusion matrix:
            Predicted(yhat)
Observed (y) negative positive Total
    negative       81       27   108
    positive       11       49    60
    Total          92       76   168
```

```
Classification Measures:

   hit rate                          0.774
   weighted hit rate                 0.785
   misclassification rate            0.226
   weighted missclassification rate  0.215

                    TP        FP Precision
    negative     0.750     0.250     0.880
    positive     0.817     0.183     0.645
```

The optimal `lambda` for the model using I-splines can be found in `results.spline$param.opt`, which is 10^1. This `lambda` value equals the penalty value which will give the lowest misclassification rate in the cross-validation. One of the advantages of using splines to handle nonlinear prediction is the possibility to show the effect of a variable by plotting its estimated transformation. Figure 5 shows these plots of the splines per variable, which can be used for interpretation of the effects of each individual variable. In this figure, one can clearly see nonlinear effects in most variables. For example, the *diabetes pedigree* (`x7`) shows a positive relation with respect to female patients with `positive` results, but with a diminishing returns to scale. On the other hand, *Age* (`x8`) has a reverse *v*-shape: respondents who are around 40 are more likely to have diabetes. Overall, *glucose concentration* (`x2`) and *BMI* (`x6`) have the largest effect on the class prediction when its values are large.

Kernel

Another way of implementing nonlinearity in the model by using a kernel. In this example, we will use the Radial Basis Function in our model training. To find the optimal λ and σ values we performed a five fold cross validation with the grids: $\lambda = 10^{-6}, 10^{-5}...10^{4}, 10^{5}, 10^{6}$ and $\sigma = 2^{-5}, 2^{-4}...2^{4}, 2^{5}$ by the following commands.

```
R> results.kernel <- svmmajcrossval(diabetes.tr$X, diabetes.tr$y,
+  scale = "interval", search.grid = list(kernel.sigma = 2^seq(-5, 5),
+  lambda = 10^seq(6, -6)), hinge = "quadratic", kernel = rbfdot,
+  weights.obs = list(positive = 2, negative = 1))
R> model.kernel <- svmmaj(diabetes.tr$X, diabetes.tr$y,
+  scale = "interval", lambda = results$param.opt$lambda,
+  kernel.sigma = crossval2$param.opt$kernel.sigma,
+  kernel = rbfdot, hinge = "quadratic",
+  weights.obs = list(positive = 2, negative = 1))
R> predict(model.kernel, diabetes.te$X, diabetes.te$y,
+  weights = list(positive = 2, negative = 1))

Prediction frequencies:
          negative positive
frequency       93       75
```

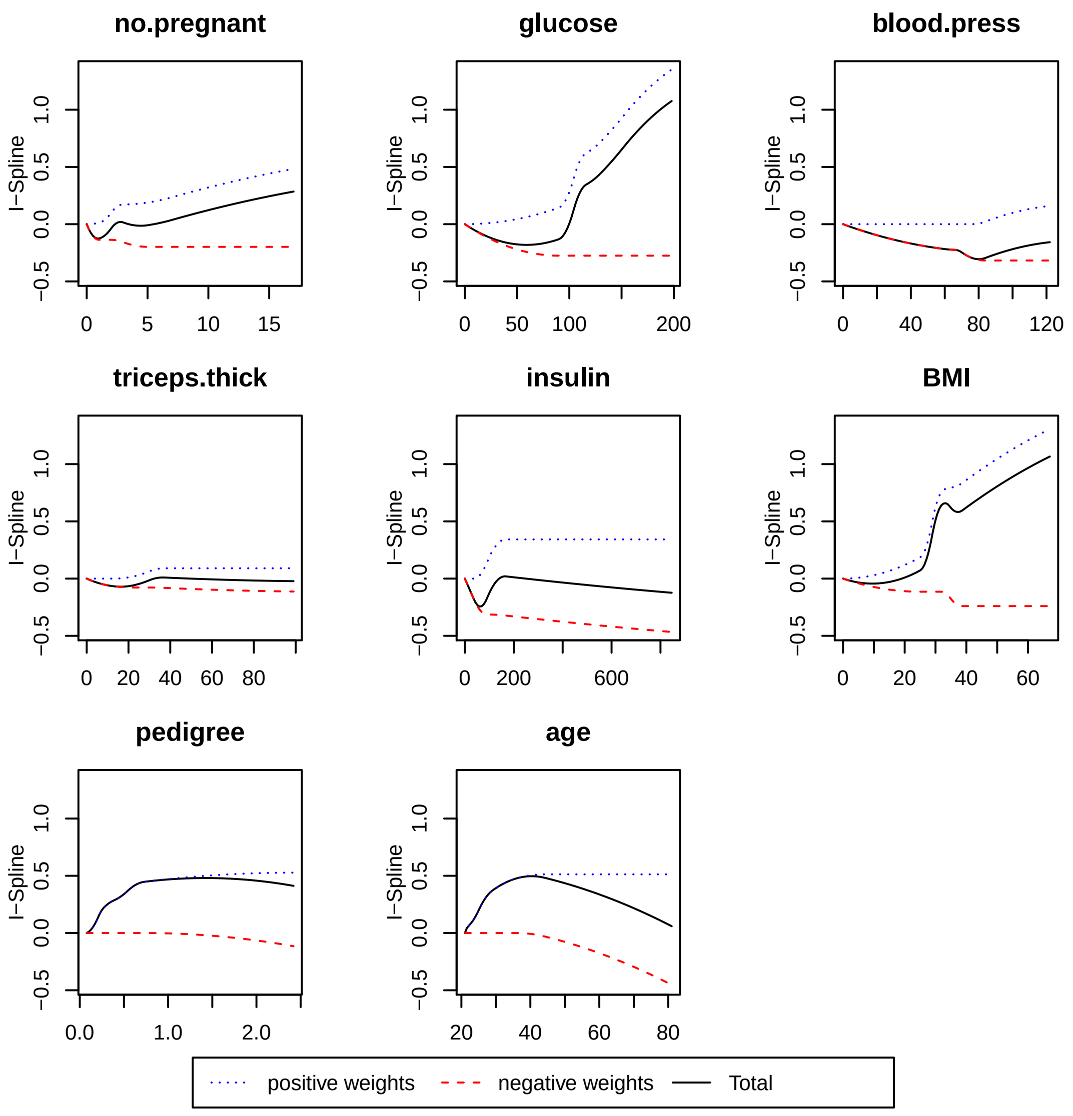

Figure 5: The spline plots of each of 8 variables used to predict the result of a test for diabetes among females of Pima Indian heritages. This model has been performed with `lambda = 10` and the spline basis with 5 inner knots and a degree of 2. Each graph denotes the loss term of the corresponding variable with different values. The higher the predicted value $\tilde{q}$, the higher the probability of positive test.

```
Confusion matrix:
            Predicted(yhat)
Observed (y) negative positive Total
    negative       83       25   108
    positive       10       50    60
    Total          93       75   168

Classification Measures:
```

```
hit rate                            0.792
weighted hit rate                   0.726
misclassification rate              0.208
weighted missclassification rate    0.274


                TP        FP Precision
 negative    0.769     0.231     0.892
 positive    0.833     0.167     0.667
```

The optimal parameter values for the kernel model are: `lambda=`10^0 and `sigma=`2^0.

Observing the prediction results, we can see that the model using I-splines has a higher TP-value of female persons having `positive` result in diabetes as well as the hit rate (average TP-value) suggesting that for these data the I-spline transformation is better able to pick up the nonlinearities in the predictor variables than the radial basis kernel.

6. Text Analysis

In the previous sections, the R-package **SVMMaj** has been introduced. In the following sections, this package is used on the speeches of the Dutch House of Representatives during the plenary meetings. As mentioned in Section 1, this application analyses the practicability of using SVM on predicting the political spectrum based on the contents of their speeches.

One major challenge is to find a proper feature selection, as many classification techniques suffer from overfitting when using too many features. Recently, another classification technique has emerged whose results are very promising: Support Vector Machine (SVM) (Vapnik 1995). Various researches have shown that SVM performs on average better than other techniques (Drucker *et al.* 1999; Joachims 1998; Pang *et al.* 2002). One of the advantages of SVM is that it can handle overfitting very well (Joachims 1998). Because of this robustness of SVM, feature selection may even be redundant. In this paper, the **SVMMaj** package in R (R Development Core Team 2011) will be introduced, which uses a majorization approach to solving SVMs, or SVM-Maj for short (Groenen *et al.* 2007, 2008).

Subsequently, the political speeches during the plenary meetings of the Dutch House of Representatives (in Dutch: De Tweede Kamer) is used to analyse the difference in contents between the political spectra of the parliamentarians. In particular, the parliamentarians have been divided into two classes: the *progressive* and the *conservative* parliamentarians. If a decent classification model can be set up based on these speeches of the politicians, it can then be used to classify text in for example forums, where the sentiment of the bloggers are not known in advance. This automated classification method can be very valuable for analysis of the segmentation of the spectra between bloggers, or even between forum sites.

With respect to text analysis, SVM has proven to be very powerful in not only text classifications, but also sentiment analysis. The idea of this implementation is to extract features from text which serve as predictor variables in the SVM model (Sebastiani 2005). Various types of features have been used in previous literature in order to improve the accuracy of the model. The simplest method is to define a single word as a feature, also known as a *unigram*. However, Polanyi and Zaenen (2005) argued that the presence of the so called *valence shifters* could change the semantic meaning of a sentence. Therefore, they introduce the so-called n-gram: a combination of n subsequent words as one feature. In particular, the use of *bigrams* and *trigrams*, a combination of respectively two and three subsequent words, are the most common used types of n-grams. On the other hand, machine learning started to gain more popularity due to its competitive prediction performance without the need of much data processing (Sebastiani 2005). Furthermore, several researches (Pang *et al.* 2002; Bekkerman and Allan 2003) has shown that the use of unigrams in combination with Machine Learning, in particular SVM, gives comparable, or even superior, prediction accuracy than using n-grams.

After selecting the right features, a measurement needs to be chosen for these features. The most straightforward one is to use the frequency of the feature in the text. Nevertheless, several other measurements have been used, in other researches. Wu *et al.* (2008), for example, introduced the term frequency – inverse document frequency (*tf-idf*), which equals the frequency divided by the number of words in the document. On the other hand, the use of a presence indicator instead has also been investigated before, which has not leaded to a unanimous conclusion: Pang *et al.* (2002) and Joachims (1998) showed that using the presence indicator leads to a better prediction, whereas McCallum and Nigam (1998) give opposing results.

Subsequently, these features can be used as predictor variables in order to predict the sentiment label – the dependent variable. One of the popular techniques is the Naïve Bayes classifier, which has been increasingly used for among others spam filtering (Sahami *et al.* 1998).

One of the problems that may arise in text analysis is the number of features to be used in the classification procedure. The more features, on the one hand, the more accurate the model can be. However, one major downside of using many features is the possibility of over fitting: the model is too specifically specified for the supervised data and predicts poor on new observations. In order to reduce this over fitting problem, feature selections are frequently used to reduce the number of features (Guyon and Elisseeff 2003). Nevertheless, SVM was proven to be robust to over fitting. (Joachims 1998)

As the relation between predictor variables and the dependent variable may not be linear, and as there exist interactions between predictor variables, neglecting these nonlinear dependencies can lead to biased results (Seber and Wild 1989). Therefore, another advantage of SVM is its possibility to include such non-linearity in its model. Many models of SVM use non-linear kernel to include non-linearity (Boser *et al.* 1992), of which the Radial Basis Function (RBF) is the most commonly used. These kernels remap the predictor matrix into an infinitely high-dimensional space, so that nonlinear relations as well as interactions are allowed. However, one disadvantage of using kernels is that the resulting model can hardly be used for interpretation. Therefore, Groenen *et al.* (2007) have introduced I-splines as an alternative of kernels to implement non-linearity. These splines make interpretation of the effects of input variables possible, while allowing non-linearity. However, these splines do no incorporate interaction.
With respect to the implementation of sentiment analysis in politics, it is still in development: a limited number of previous papers can currently be found. For example, Strapparava *et al.* (2010) have used Support Vector Machine (SVM) to show its viability to predict the

Party	Full name	Members
CDA	Christen-Democratisch Appèl	21
CU	ChristenUnie	5
D66	Politieke Partij Democraten 66	10
GL	GroenLinks	10
PvdA	Partij van de Arbeid	30
PvdD	Partij voor de Dieren	2
PVV	Partij Voor de Vrijheid	24
SGP	Staatkundig Gereformeerde Partij	2
SP	Socialistische Partij	15
VVD	Volkspartij voor Vrijheid en Democratie	31
Total		150

Table 8: Overview of the members of the House of the Representatives by political party

persuasiveness of political discourses. On the other hand, Thomas *et al.* (2006) analysed the automatic prediction of whether a political speech supports or opposes the political issue.

7. Dutch House of Representatives

In the Netherlands, the parliament – the States-General of the Netherlands – consists of two chambers: the House of the Representatives (in Dutch: de Tweede Kamer) and the Senate (in Dutch: de Eerste Kamer). As the Netherlands has a multi-party system, these chambers are being seated by members of various political parties. Of these two chambers, the House of the Representatives is the lower house of the parliament, in which also forms the executive government of the Netherlands – the Cabinet. This chamber consists of 150 members – parliamentarians, among which a Cabinet is formed every four years. Table 8 shows the mapping of the current members of the House of the Representatives by its political party.

This Cabinet consists of ministers and state secretaries, who are responsible for governing the country, while the parliamentarians are scrutinizing their work on behalf of the Dutch people and making laws in cooperation with the Cabinet. Therefore, the House regularly hold meetings to discuss the political issues (*topics*), which are being brought forward by fellow parliamentarians. Among these meetings, the Plenary Meetings is the assembly of all 150 parliamentarians. The former incumbent Cabinet, Rutte Cabinet, which was officially formed on 14 October 2010, has officially resigned on the 23 April 2012. Therefore, as of today, the country is being governed by the caretaker government, the former Rutte Cabinet. Nevertheless, this paper focus on the period before their resignation.

7.1. Plenary Meetings of the Dutch House of Representatives

Each Plenary Meeting of the Dutch House of Representatives starts with the agenda and other schemes of the meeting. Subsequently, several topics will be put forward for debate or discussion. Although each topic will be discussed subsequently, it is common that the discussion of a topic will be continued to the next meeting several times. Moreover, during the meetings, each parliamentarian has the right to plead the Cabinet for taking specific

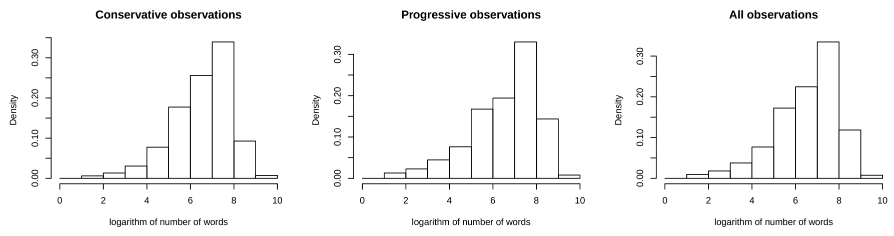

Figure 6: This figures shows the distribution of words in each observation.

matters into consideration. This is usually done by proposing a motion (in Dutch: motie). After the proposal of the motion, the chairman will arrange time for the members to cast their voting, which can be held at the same day, or postpone it to another meeting. Before the actual voting takes place, the motion can be withdrawn, suspended, changed or decayed.

For each plenary meeting, minutes are being made, which has been separated in topics of discussion. Moreover, in these minutes all statements of each politician have been noted in a chronological order. These minutes have been stored and made available for reference (Tweede Kamer der Staten-Generaal 2012) in html-format. From these minutes, each statement of a politician within the meeting has been stored systematically within an html-tag, while each proposal of a motion, as well as the change of status of a motion – that is, whether the motion has been voted, withdrawn, suspended, changed or decayed – are stored in a different tag.

For this study, a collection of these minutes of the past plenary meetings of the Dutch House of Representatives, between 6 September 2011 and 6 March 2012 has been used. These meetings do not only include the speeches of the parliamentarians, but also of the chairman of the meeting, the members of the Cabinet – that is, the ministers and state secretaries – as well as other guests. These speeches will not be taken into account in this research, as the political spectrum of these speakers are usually unknown. In total, the data set consist of 67 meetings, covering 250 different topics. After collection of these minutes, the content has been divided in observations. One possibility is to define one observation as one turn of speaking time a parliamentarian has used. In this situation, however, there is a high possibility that many observations only consist of short questions or short answers of the parliamentarian, which will be hard to predict its sentiment. To overcome this problem, all turns of a specific parliamentarian within the same session will be concatenated into one observation, that is, all turns of the same parliamentarian, in the same meeting and the same topic being discussed. In total, there are 1990 observations, of which the distribution among the parties is shown in Table 9. Figure 6 shows the distribution of number of words in each observation.

As the purpose of this research is to analyse the practicability of using text analysis to predict the sentiment of an observation, only the texts are included as explantory variables in this model. Including the party of which the parliamentarian belongs to as explanatory variable, for example, would cause the model to specify a too heavy weight on these variables instead of the contents of the text.

7.2. Class Labels

Various classifications of political spectrum are possible. Due to simplicity, only the classi-

Spectrum	Party	Total Members		Number of Observations	
Conservative	CDA	21	14.0%	251	12.6%
	CU	5	3.3%	137	6.9%
	VVD	31	20.7%	257	12.9%
	PVV	24	16.0%	236	11.9%
	SGP	2	1.3%	100	5.0%
	Total	83	55.3%	981	49.3%
Progressive	D66	10	6.7%	219	11.0%
	GL	10	6.7%	189	9.5%
	PvdA	30	20.0%	281	14.1%
	PvdD	2	1.3%	63	3.2%
	SP	15	10.0%	257	12.9%
	Total	67	44.7%	1,109	50.7%
Total		150	100.0%	1,990	100.0%

Table 9: Distribution of the number of members and the number of turns by political party. All ten political parties has been classified as conservative or progressive party based on the classification of Kieskompas (Kieskompas 2010).

fication between progressivism and conservatism has been used in this research. During the General Elections in 2010, Kieskompas (Kieskompas 2010) has classified the political spectrum of all participating parties based on the 30 political issues. This classification is based on progressivism versus conservatism, as well as left-wing versus right-wing. See Figure 7 for the graphical interpretation of their spectrums. Here we can see a clear separation between de conservative parties and the progressive parties. In order to investigate more deeply into the classification, the votings of the *motions* during the meetings has been used for analysis.

Table 10 shows an overview of the statuses of all proposed motions between 6 September 2011 and 6 March 2012. Here we can see that for most of the motions, a voting was being held (77.1%). For each of these 1250 votings, two of them are voted per head of parliamentarian, while the rest of the votings are voted per party. For these votes, the voting per party has been used to analyze the correlation of the votings between the parties. For this analysis, a multidimensional scaling (Borg and Groenen 2005) with two dimensions has been performed. First of all, the correlation of the votings between political parties are calculated, which is shown in Table 11. Subsequently, these correlations are transformed into dissimilarity measures by taking the root of 1 minus the correlation, that is $s_{ij} = \sqrt{1 - \rho_{ij}}$ where ρ_{ij} and s_{ij} respectively denotes the correlation and its derived dissimilarity measure between party i and j. The resulting dissimilarity matrix can then be used to find the multidimensional scaling by using the **SMACOF** package de Leeuw and Mair (2009), which is shown in Figure 8. Here we see that the similarity plot is quite similar to the spectrum introduced by Kieskompas (2010). Nevertheless, in both graphs, the ChristenUnie and D66 are rather close to the center, which implies their spectrum to be rather centered.

In this research, the classification of the conservative versus progressive parties is used as class labels y by classifying all parliamentarians of the same party with the same class label, independent of the topic. This results in a total data set of 1990 observations, of which 981 are classified as conservative, and 1009 as progressive. Table 12 contains the descriptive statistics

Status	Number of motions	Percentage
Accepted	498	30.7%
Rejected	752	46.4%
Suspended	133	8.2%
Changed	9	0.6%
Withdrawn	50	3.1%
Decayed	40	2.5%
Unknown	140	8.6%
Total	1,622	100.0%

Table 10: Overview of the status of the motions, which are were proposed between 6 September 2011 and 6 March 2012.

	CDA	CU	D66	GL	PvdA	PvdD	PVV	SGP	SP
CDA									
CU	0.399								
D66	0.261	0.333							
GL	0.048	0.279	0.538						
PvdA	0.180	0.332	0.441	0.546					
PvdD	-0.049	0.184	0.238	0.477	0.452				
PVV	0.409	0.027	-0.088	-0.291	-0.181	-0.197			
SGP	0.699	0.484	0.209	0.025	0.125	-0.071	0.371		
SP	0.024	0.182	0.198	0.401	0.402	0.674	-0.119	-0.014	
VVD	0.675	0.231	0.212	-0.025	0.087	-0.105	0.613	0.541	-0.021

Table 11: Correlation matrix between parties of the votings of motions, which were proposed and voted between 6 September 2001 and 6 March 2012.

of the number of words of the collected observations, while Figure 6 shows the histogram of the number of words of these observations. Here we can see no significant difference in the number of words each parliamentarian uses during each session.

7.3. Predicting observations from a new meeting

The selection of the test set can be based on different division methods. Usually, these observations have been randomly selected, in order to avoid biased selection. However, it is also interesting to investigate the prediction performances of a new meeting based on its previous meetings. Therefore, the total set of observations has been divided into three subsets based on the date of the meeting. Using this division, the ability of SVM to predict from an unseen meeting is being examined. Throughout the study, the observations has been divided into 3 subsets. Subset 1 is primary being used for feature selection, that is, to define the variables to be used in the model, while Subset 2 is used for the estimation of the model. By dividing these first two subsets, I try to minimize the problem that temporarily used words will be obtain a higher weight in the model. Subsequently, Subset 3 is used as test set in order to analyse the prediction performances. Table 13 shows the descriptives of the subsets.

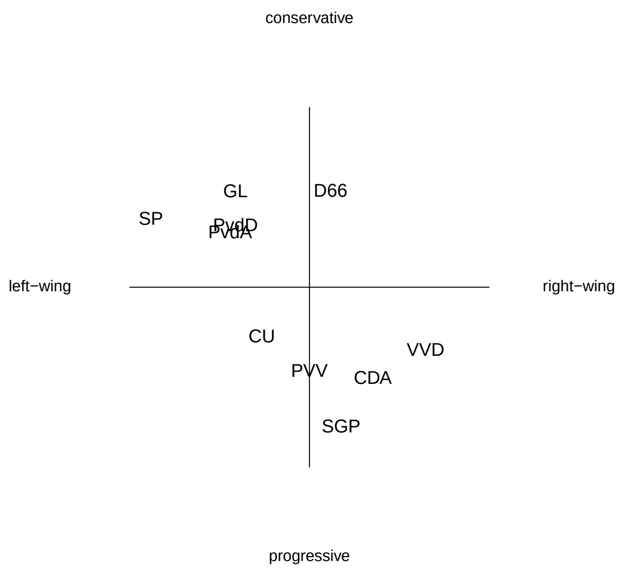

Figure 7: Political landscape of the Dutch political parties according to Kieskompas (Kieskompas 2010).

7.4. Text pre-processing

To be able to extract useable information out of the spoken sentences, a few text pre-processing measures are needed. First of all, we would like to remove the conjugation of words. Whether a person has walked, has been walking, or currently walks on the street, all these conjugations refers to the same verb: to walk. Therefore, a word stemming is performed, where all morphological changes of a word has been transformed to its stem. This is done with an algorithm, available in the Snowball package [Porter 2001]. A common application in text categorization is the use of ontology, where words with similar meaning are being standardized into one feature. However, as it is assumed that the word choice may inherent the sentiment of the speaker, this has not been performed in this research. After these corrections, there are in total 20,364 features. Figure 9 shows the distribution of the frequencies of the resulting features of the total dataset, as well as the distribution per sentiment class, while Table 14 shows the mean, median and the standard deviation of the frequencies. In the latter figure, one can see slightly higher frequencies of the conservative observations of the relative frequently used features, which can be explained by the relatively larger number of words per observation of the conservative parliamentarians. Furthermore, there are 5,254 features which only appear in conservative observations, while 6,392 features did only appear in the progressive observations. The frequency distributions are shown in Figure 15.

Although SVM does not require limiting the number of features in order to perform well, implementing too many features may increase the computational time of the model. Also,

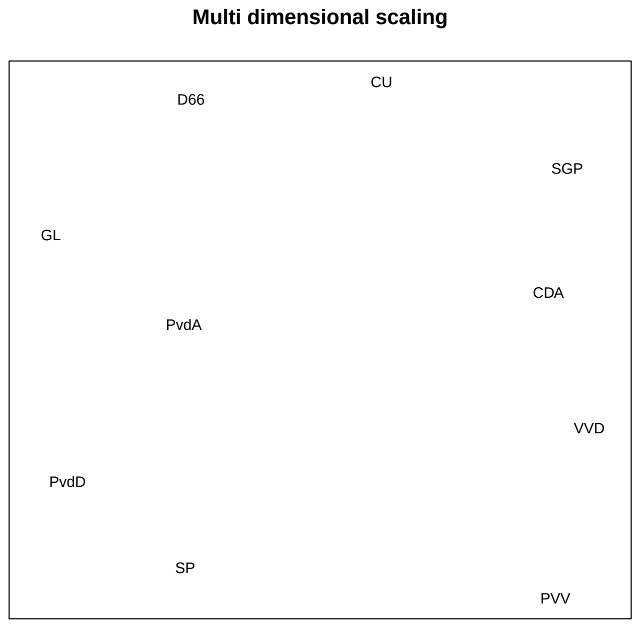

Figure 8: The multidimensional scaling of the votings of the motions of the House of Representatives.

there are many features which have only appeared once in the total dataset. Therefore, it is recommended to reduce the number of features by omitting features with a total frequency below a certain threshold. In this research, the probability per word that the feature appears is assumed to be binary distributed with probability $p = \overline{tf}_{j,cons}$ for conservative observations and $p = \overline{tf}_{j,prog}$ for progressive observations, where $\overline{tf}_{j,class}$ denotes the average term frequency of word j occurring a observation of class *class*. Using this assumption, the features will be selected using

$$\overline{tf}_{j,cons} - \overline{tf}_{j,prog} \sim N(0, \frac{var(tf_{i,j,cons})}{n_{cons}} + \frac{var(tf_{i,j,prog})}{n_{prog}}), \tag{16}$$

where $tf_{j,cons}$ and $tf_{j,prog}$ denotes the term frequency of word j occurring in the conservative and respectively the progressive observations. Subsequently, the p-value is denoted as the probability that the probability that the difference is larger than $\overline{tf}_{j,cons} - \overline{tf}_{j,prog}$. In this study, the rejection criteria of $\alpha = 1.0\%$, $\alpha = 2.5\%$ and $\alpha = 5.0\%$ are used.

Figure 10 shows the distribution of all frequencies in of the observations in Subset 2, while Figure 11 shows the distribution of the selected features. Here we see that not only the more frequently used features, but also the rather occassionally used features are being selected from the feature selection.

Of those selected, several measurements have been used in order to compare the accuracy between them.

		Total	Mean	St.dev.	Observations
Conservative	Absolute	1,317,024.	1,342.53	1,475.	981
	Logarithm	6,427.37	6.55	1.344	
Progressive	Absolute	1,506,946.	1,493.50	1,614.	1,009
	Logarithm	658.58	6.52	1.552	

Table 12: Statistics of the number of words in the observations in absolute value, as well as in the logarithmic scale.

	Subset		
	1	2	3
First meeting	2011-09-06	2011-11-02	2011-11-22
Last meeting	2011-11-01	2011-12-21	2012-03-08
Number of meetings	22	22	22
Conservative observations	346	357	278
Progressive observations	338	358	313
Total observations	684	715	591

Table 13: This table shows some information about the divided subsets: date of the first and last meeting which are included into the subset, the total number of meetings and number of observations within the subset.

- *The term frequency* denotes the frequency that the feature has appeared in the observation.
- *The presence indicator* to indicate whether the feature appears in the observation. As (Pang *et al.* 2002) has shown in his paper that the use of these binary variables instead are already sufficient for text categorization, it is interesting whether this is also the case for text sentimentation.

7.5. Cross Validation

As mentioned before, Subset 2 is used to estimate the model. As it is not known on beforehand what the optimal model is, I introduce several models whose results be compared with each other.

First of all, two implementations of non-linearities are used: I-splines and the RBF-kernel. Although the RBF-kernel is more likely to result in higher forecast performance, the advantage of using splines over RBF-kernel is its possibility to analyse the actual effects of the features. On the other hand, the use of term frequency will be compared against the use of the presence indicator. Thus, there are in total 4 different models to be compared, which is shown in Table 16. After selecting the optimal model, this model will be performed as well by using a lower rejection criterium ($\alpha = 10\%$) in the feature selection, in order to analyse the effect of the feature selection.

From this *training set*, a 5-fold cross validation has been performed, with a fine grid of $\lambda = \{2^{-10}; 2^{-9}; 2^{-8}; \ldots; 2^{8}; 2^{9}; 2^{10}\}$. Amongst these values, the optimal λ-value is the

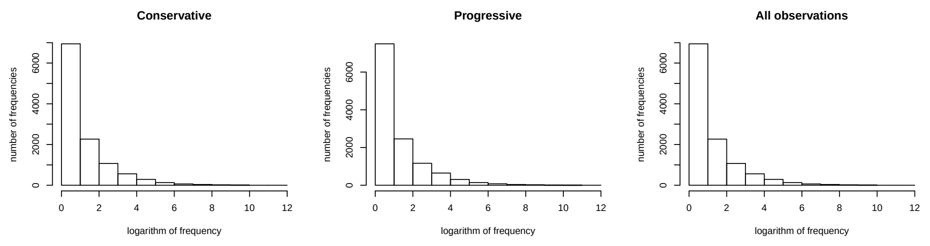

Figure 9: Distribution of the feature frequencies, categorised by political spectrum.

	Sentiment class		
Statistics	Conservative	Progressive	Total
minimum	0.00	0.00	1.00
median	1.00	1.00	2.00
mean	23.62	28.19	51.81
maximum	19,672.00	23,720.00	43,392.00
st. Dev.	346.68	414.93	761.45

Table 14: Statistics of the feature frequencies, categorised by political spectrum.

one which results in the lowest misclassification rate. In case of an RBF-kernel, the optimal regularization parameter λ as well as the optimal kernel parameter σ has been found by performing a cross validation with a grid of $\lambda = \{2^0; 2^1; 2^2; \ldots; 2^8; 2^9; 2^{10}\}$ and $\sigma = \{2^{-5}; 2^{-4}; 2^{-3}; \ldots; 2^{-13}; 2^{-14}; 2^{-15}\}$.

With these optimal values, a new model is being performed. As a control set, the remaining 591 observations in Subset 3 will be used to analyse the prediction performances of a set of unseen observations by comparing the hit rate – percentage of correctly predicted observations.

8. Results

The prediction results of the cross-validated models are shown in Table 17. For each combination of nonlinearity and feature measurement, the percentage of correctly predicted observations (hit rate) and its corresponding area-under the receiver operating characteristic-curve (auc) are shown. The receiver operating characteristic curve (ROC-curve) (Bradley 1997) illustrates the performance in binary classification of the model, where it graphically shows the false positive rate at the horizontal axis against the true positive rate at the vertical axes using different threshold value for $\widetilde{q}_i$ in (1). From this curve, the total area below the curve is denoted as the area-under the curve (auc) value. Note that this value usually range from 0.5 and 1, where 1 denotes that $\widetilde{q}_i$ between both classes can be perfectly separated, while an auc-value of 0.5 shows no differentiation between both classes at all. In both the training set (Subset 2) and test set (Subset 3), the number of conservative observations are about equal to the number of progressive observations. Therefore, when using majority voting to predict the class, we would obtain a hit rate of 50%, which will be used as benchmark performance.

Table 17 shows the prediction results of the SVM-model with different nonlinearity implemen-

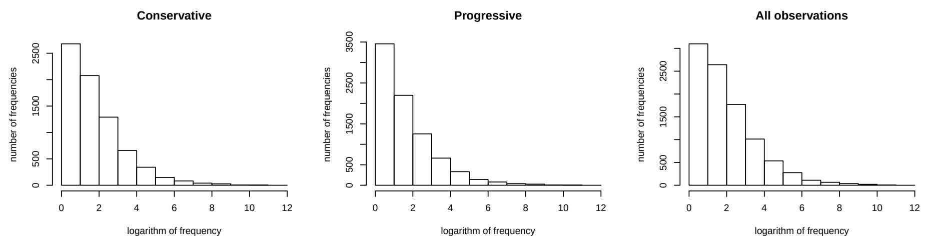

Figure 10: Distribution of the feature frequencies, in Subset 2, before applying the feature selection.

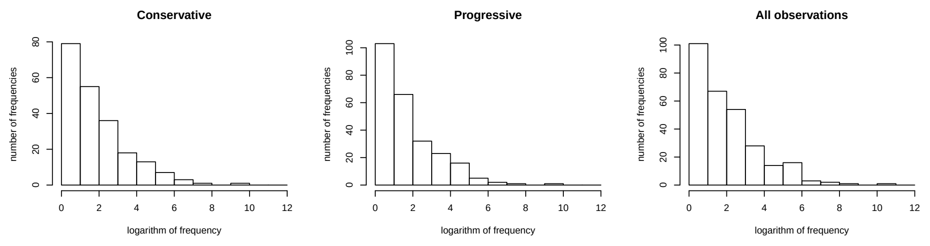

Figure 11: Distribution of the feature frequencies, in Subset 2, after applying the feature selection.

tation and different feature measurement. First of all, all models are being able to predict better than the benchmark with hit rates of larger than 60%. Furthermore, we see that the use of presence indicator instead of the term frequency will have a negative effect on the prediction outcome. In both nonlinear models, the use of term frequency yields about 7% higher hit rate in case of RBF-kernel, and about 1% in case of splines. Therefore, this data set suggest that not only the use of the word, but also the frequency is an important measure to take into account. Furthermore, in case of using the term frequency, the model with RBF-kernel results in the highest hit rate of nearly 70%. From these 4 models, Model (1) and (3) results in the highest hit rates, therefore, both models is used with varying α-valudes to analyse the effect of feature selection on the prediction. These results are shown in Table 17b. Remarkinly, the prediction of the spline model (1) increases when the selection criterium either increase or decrease. In case of $\alpha = 1.0\%$, the hit rate of the spline-model can be increased to 65.5%. On the other hand, the formerly used criterium $\alpha = 2.5\%$ results in the best prediction among all. A stricter selection criterium result into a slightly lower hit rate, while a higher α value will lower the hit rate by nearly 2%. Overall, it seems that the chosen feature selection does not have a clear effect on the prediction outcome, which can be explained by the overall robustness to over fitting.

9. Conclusion

This paper introduces the R-package **SVMMaj**, this package implements the SVM-Maj algorithm of Groenen *et al.* (2007, 2008) with the addition of nonlinear models with kernels.

	Before	After
minimum	0.000	0.000
maximum	10.861	10.301
median	1.609	1.609
mean	1.911	1.42
standard deviation	1.667	1.794
kurtosis	1.986	1.643
skewness	1.211	1.177

Table 15: Statistics of the feature frequencies of all observations, before and after applying the feature selection.

Model	Non-linearity	Feature Measurement
1	Splines	Presence Indicator
2	Kernel	Presence Indicator
3	Splines	Term Frequency
4	Kernel	Term Frequency

Table 16: Four different models which has been used to compare the results. Among these models, the one with the highest hit rate is used to perform a second model with a less strict feature selection criterium in order to analyse its effect.

.

One of the advantages of the SVM-Maj approach is the competitively fast training speed for medium sized problems. Furthermore, it allows individual objects in a training dataset to receive different individual weight.

Another advantage of SVM-Maj is the possibility to use different loss functions, besides the commonly used absolute hinge. In this package, the absolute hinge, quadratic hinge and Huber hinge has been implemented. Nevertheless, this can be expanded to any other error function $f(q)$ that satisfies the following condition: the second derivative of its function has a bounded maximum, so that a quadratic majorization function can be found. If in addition $f(q)$ is convex, then the overall loss function (4) is strictly convex, so that the the SVM-Maj algorithm is guaranteed to stop at the global minimum.

Furthermore, this paper also investigates the practicability of using Support Vector Machine as an automatic Machine Learning Technique on predicting the political spectrum of a person based on their choice of words in political issues. For this research, the statements of the parliamentarians in the plenary meetings of the House of Representatives have been used.

Overall, the SVM model can obtain a prediction hit rate of 68.7%, which is much higher than the majority voting of 60%, which serves as benchmark performance. The results has shown a slightly lower hit rate when using the presence indicator in combination with a kernel to allow for non-linearity. On the other hand, the use of splines instead of the kernel yield a much worse performance: it results in a slightly higher hit rate than majority voting. This may indicate the importance of the interaction effects of the *presence* of the features in combination with its frequency within an observation. Nevertheless, it is therefore hard to obtain a clear interpretation of these effects.

Model	α	Non-linearity	Measurement	Hit Rate	AUC
1	2.50%	Splines	Presence Indicator	61.8%	0.679
2	2.50%	Kernel	Presence Indicator	61.8%	0.669
3	1.00%	Splines	Term Frequency	65.5%	0.701
3	2.50%	Splines	Term Frequency	63.3%	0.697
3	5.00%	Splines	Term Frequency	64.8%	0.709
4	1.00%	Kernel	Term Frequency	68.2%	0.737
4	2.50%	Kernel	Term Frequency	68.7%	0.746
4	5.00%	Kernel	Term Frequency	66.8%	0.746

Table 17: Results of the text sentimentation by using Support Vector Machine. This table hows the prediction hit rate as well as the area under the ROC-curve of the 4 models, including the prediction performances of using a different feature selection criterium.

Apart from this, these results has indicated several other issues. First of all, the political spectrum which has been used as class label in SVM has been strictly based on the political spectrum of the party where the parliamentarian belongs to. This is a very rough way of labelling, as there may be differences in the sentiment between parliamentarians, or a party may even be on a different spectrum for different topics. Nevertheless, when neglecting the differences in the meeting, SVM can still predict decently despite this issue. Furthermore, the voting behaviour of the parties shows that on overall, the chosen clustering can still be observed.

Moreover, as the topics of the plenary meetings of the House of the Representatives ranges very widely, the diversity of terminologies and jargons which has been used may also heavily vary. Therefore, it may be preferable to divide the topics in categories on beforehand, in order to limit the variety of topics.

A. Efficient updates for SVM-Maj

Recall that the relationship between $\mathbf{q}$ and $\boldsymbol{\beta}$ can be written as

$$\mathbf{q} = \mathbf{X}\boldsymbol{\beta}. \tag{17}$$

In some situations, different values of $\boldsymbol{\beta}$ may lead to the same $\mathbf{q}$, when deriving $\mathbf{q}$ from $\boldsymbol{\beta}$. In other situations, the opposite could happen, that is, several values of $\mathbf{q}$ may result into the same $\boldsymbol{\beta}$ value. Thus, there is not always a one-to-one mapping of $\mathbf{q}$ to $\boldsymbol{\beta}$ and the reverse. Therefore, one should take these possible situations into account when performing iterative majorization. To illustrate this, we will use the singular value decomposition (SVD) of $\mathbf{X}$:

$$\underset{(n\times k)}{\mathbf{X}} = \underset{(n\times r)}{\mathbf{P}} \; \underset{(r\times r)}{\boldsymbol{\Lambda}} \; \underset{(r\times k)}{\mathbf{Q}'}, \tag{18}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are orthonomal matrices which satisfy $\mathbf{P}'\mathbf{P} = \mathbf{I}$ and $\mathbf{Q}'\mathbf{Q} = \mathbf{I}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix. The relationship between $\mathbf{q}$ and $\boldsymbol{\beta}$ can then be written as:

$$\mathbf{q} = \mathbf{X}\boldsymbol{\beta} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{Q}'\boldsymbol{\beta}, \tag{19}$$

$$\underset{(r\times 1)}{\mathbf{P}'\mathbf{q}} = \underset{(r\times r)}{\boldsymbol{\Lambda}} \quad \underset{(r\times 1)}{\mathbf{Q}'\boldsymbol{\beta}.} \tag{20}$$

Let us first examine the left part of the equation. $\mathbf{q}$ can be written as a combination of two orthogonal vectors, a projection of $\mathbf{q}$ on $\mathbf{P}$ and a projection on its complement $(\mathbf{I}-\mathbf{PP}')$, that is,

$$\mathbf{q} = \mathbf{PP}'\mathbf{q} + (\mathbf{I}-\mathbf{PP}')\mathbf{q} = \mathbf{q}_B + \mathbf{q}_N. \tag{21}$$

Multiplying both sides with $\mathbf{P}'$ gives

$$\begin{aligned}\mathbf{P}'\mathbf{q} &= \mathbf{P}'(\mathbf{q}_B + \mathbf{q}_N)\\ &= \mathbf{P}'\mathbf{q}_B + \mathbf{P}'(\mathbf{I}-\mathbf{PP}')\mathbf{q}\\ &= \mathbf{P}'\mathbf{q}_B + (\mathbf{P}'-\mathbf{P}')\mathbf{q}\\ &= \mathbf{P}'\mathbf{q}_B + \mathbf{0}\\ &= \mathbf{P}'\mathbf{q}_B.\end{aligned}$$

In other words, the left part of (20) is only dependent of $\mathbf{q}_B$. When $r = n$, $\mathbf{PP}'$ equals $\mathbf{I}$ and thus, $\mathbf{q} = \mathbf{q}_B + \mathbf{q}_N = \mathbf{q}_B + (\mathbf{I}-\mathbf{I})\mathbf{q} = \mathbf{q}_B$. In this case there is always an unique solution of

$$\mathbf{P}'\mathbf{q} = \boldsymbol{\Lambda}\boldsymbol{\theta}, \text{for any } \boldsymbol{\theta} \in \Re^r. \tag{22}$$

However, when $r < n$, $\mathbf{PP}' = \mathbf{I}$ does not hold and there are infinitely many solutions to (22) so that an one-to-one relationship of $\mathbf{q}$ and $\boldsymbol{\beta}$ in (17) is lost indeed.

Similarly, $\boldsymbol{\beta}$ can be written as $\boldsymbol{\beta} = \mathbf{QQ}'\boldsymbol{\beta} + (\mathbf{I}-\mathbf{QQ}')\boldsymbol{\beta} = \boldsymbol{\beta}_B + \boldsymbol{\beta}_N$, and $\mathbf{Q}'\boldsymbol{\beta} = \mathbf{Q}'(\boldsymbol{\beta}_B + \boldsymbol{\beta}_N) = \mathbf{Q}'\boldsymbol{\beta}_B$. If $r < k$ then there is no unique solution to $\mathbf{Q}'\boldsymbol{\beta} = \boldsymbol{\theta}$ with $\boldsymbol{\theta} \in \Re^r$. Note that $\boldsymbol{\beta}_B$ and $\boldsymbol{\beta}_N$ are independent to each other and that

$$\boldsymbol{\beta}_N'\boldsymbol{\beta}_B = \boldsymbol{\beta}_B'\boldsymbol{\beta}_N = \boldsymbol{\beta}'(\mathbf{QQ}')(\mathbf{I}-\mathbf{QQ}')\boldsymbol{\beta} = 0, \tag{23}$$

as $\mathbf{QQ}'(\mathbf{I}-\mathbf{QQ}') = \mathbf{0}$.

As a result, we have to take care that a proper relation between $\boldsymbol{\beta}$ and $\mathbf{q}$ is retained, when more efficient updates are derived. In the next section, we will examine three different situations and introduce the optimization method by optimizing the parameters which has the lowest dimension, that is, $\min(r, n, k)$. In this way, it can be assured that $\mathbf{q}_N$ and $\boldsymbol{\beta}_N$ are both zero vectors. Subsequently, we will discuss the use of each optimization method in each situation.

A.1. $\boldsymbol{\beta}$-method: Full rank and more objects than variables ($n > k$ and $r = k$)

When the number of variables is smaller than the number of objects, and if $\mathbf{X}$ is of full rank, then $\mathbf{QQ}' = \mathbf{I}$ and each $\boldsymbol{\beta}$ will give an unique $\mathbf{q}$. As $\dim(\boldsymbol{\beta}) < \dim(\mathbf{q})$, it is most efficient to optimize the loss function through $\boldsymbol{\beta}$. The majorization function can as follows be written as a function of $\boldsymbol{\beta}$.

$$\text{Maj}(\boldsymbol{\beta}, \alpha) = (\alpha\mathbf{1} + \mathbf{X}\boldsymbol{\beta})'\mathbf{A}(\alpha\mathbf{1} + \mathbf{X}\boldsymbol{\beta}) - 2(\alpha\mathbf{1} + \mathbf{X}\boldsymbol{\beta})'\mathbf{b} + \lambda\boldsymbol{\beta}'\boldsymbol{\beta} + o. \tag{24}$$

To derive an update, we set the first derivatives of (24) with respect to $\boldsymbol{\beta}$ and α to zero, which yields

$$\begin{aligned}\mathbf{1}'\mathbf{A}(\mathbf{1}\alpha + \mathbf{X}\boldsymbol{\beta}) &= \mathbf{1}'\mathbf{b} \\ \mathbf{X}'\mathbf{A}(\mathbf{1}\alpha + \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta} &= \mathbf{X}'\mathbf{b},\end{aligned}$$

or in matrix form,

$$\begin{bmatrix} \mathbf{1}'\mathbf{A}\mathbf{1} & \mathbf{1}'\mathbf{A}\mathbf{X} \\ \mathbf{X}'\mathbf{A}\mathbf{1} & \mathbf{X}'\mathbf{A}\mathbf{X} + \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \alpha \\ \boldsymbol{\beta} \end{bmatrix} = \begin{bmatrix} \mathbf{1}'\mathbf{b} \\ \mathbf{X}'\mathbf{b} \end{bmatrix}.$$

Using the fact that $\alpha_0\mathbf{1} + \mathbf{X}\boldsymbol{\beta} = \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}\begin{bmatrix}\alpha \\ \boldsymbol{\beta}\end{bmatrix}$, an update of both $\boldsymbol{\beta}$ and α_0 can be derived by solving the linear system

$$\left(\begin{bmatrix}\mathbf{1}' \\ \mathbf{X}'\end{bmatrix} \mathbf{A} \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix} + \lambda \begin{bmatrix} 0 & \mathbf{0}' \\ \mathbf{0} & \mathbf{I}\end{bmatrix}\right) \begin{bmatrix}\alpha^+ \\ \boldsymbol{\beta}^+\end{bmatrix} = \begin{bmatrix}\mathbf{1}'\mathbf{b} \\ \mathbf{X}'\mathbf{b}\end{bmatrix},$$

or, in compact form

$$\left(\widetilde{\mathbf{X}}'\mathbf{A}\widetilde{\mathbf{X}} + \lambda\mathbf{J}\right)\begin{bmatrix}\alpha^+ \\ \boldsymbol{\beta}^+\end{bmatrix} = \tilde{\mathbf{X}}'\mathbf{b}, \tag{25}$$

where $\widetilde{\mathbf{X}} = \begin{bmatrix}\mathbf{1} & \mathbf{X}\end{bmatrix}$ and $\mathbf{J} = \begin{bmatrix} 0 & \mathbf{0}' \\ \mathbf{0} & \mathbf{I}\end{bmatrix}$.

To prove that the the linear system (25) has a unique solution under the assumption that $\mathbf{X} \neq \mathbf{0}$, we will show that the matrix $\widetilde{\mathbf{X}}'\mathbf{A}\widetilde{\mathbf{X}} + \lambda\mathbf{J}$ is positive definite. Let us examine the following equation

$$\begin{aligned}\begin{bmatrix}\alpha & \boldsymbol{\beta}'\end{bmatrix}\left(\widetilde{\mathbf{X}}'\mathbf{A}\widetilde{\mathbf{X}} + \lambda\mathbf{J}\right)\begin{bmatrix}\alpha \\ \boldsymbol{\beta}'\end{bmatrix} &= \begin{bmatrix}\alpha & \boldsymbol{\beta}\end{bmatrix}\widetilde{\mathbf{X}}'\mathbf{A}\widetilde{\mathbf{X}}\begin{bmatrix}\alpha \\ \boldsymbol{\beta}\end{bmatrix} + \lambda\begin{bmatrix}\alpha & \boldsymbol{\beta}'\end{bmatrix}\mathbf{J}\begin{bmatrix}\alpha \\ \boldsymbol{\beta}\end{bmatrix} \\ &= (\alpha\mathbf{1} + \mathbf{X}\boldsymbol{\beta})'\,\mathbf{A}\,(\alpha\mathbf{1} + \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}'\boldsymbol{\beta}.\end{aligned} \tag{26}$$

From this equation, we can see that (26) equals the sum of two positive values. Furthermore, the right part $\lambda\boldsymbol{\beta}'\boldsymbol{\beta}$ of the equation equals zero only if $\boldsymbol{\beta} = \mathbf{0}$, whereas the left part will be zero only when $(\alpha\mathbf{1} + \mathbf{X}\boldsymbol{\beta}) = \mathbf{0}$. As both equations only hold when $\alpha = 0$ and $\boldsymbol{\beta} = \mathbf{0}$, we know that the matrix $\widetilde{\mathbf{X}}'\mathbf{A}\widetilde{\mathbf{X}} + \lambda\mathbf{J}$ is positive definite.

An update of $\mathbf{q}$ can be calculated by $\mathbf{q}^+ = \mathbf{X}\boldsymbol{\beta}^+$. Note that $\mathbf{q}_N$ necessarily equals zero, as

$$\mathbf{q}_N = (\mathbf{I} - \mathbf{P}\mathbf{P}')\mathbf{q} = (\mathbf{I} - \mathbf{P}\mathbf{P}')\mathbf{P}\boldsymbol{\Lambda}\mathbf{Q}'\boldsymbol{\beta} = (\mathbf{P} - \mathbf{P}\mathbf{P}'\mathbf{P})\boldsymbol{\Lambda}\mathbf{Q}'\boldsymbol{\beta} = \mathbf{0}.$$

A.2. q-method: Full rank and less objects than variables ($n < k$ and $r = n$)

When $k > n$, or more general $k > r$, we know that there are no unique solutions to $\mathbf{q} = \mathbf{X}\boldsymbol{\beta}$ and that $\mathbf{X}\boldsymbol{\beta} = \mathbf{X}(\boldsymbol{\beta}_B + \boldsymbol{\beta}_N) = \mathbf{X}\boldsymbol{\beta}_B$, that is, $\mathbf{q}$ is not dependent of $\boldsymbol{\beta}_N$. Consider the penalty term $\lambda\boldsymbol{\beta}'\boldsymbol{\beta}$. As $\mathbf{Q}\mathbf{Q}'(\mathbf{I} - \mathbf{Q}\mathbf{Q}') = \mathbf{0}$, the penalty term can be simplified as

$$\lambda\boldsymbol{\beta}'\boldsymbol{\beta} = \lambda(\boldsymbol{\beta}_B + \boldsymbol{\beta}_N)'(\boldsymbol{\beta}_B + \boldsymbol{\beta}_N) = \lambda\boldsymbol{\beta}_B'\boldsymbol{\beta}_B + \lambda\boldsymbol{\beta}_N'\boldsymbol{\beta}_N$$

Moreover, as $\lambda\boldsymbol{\beta}_N'\boldsymbol{\beta}_N \geq 0$ and $\mathbf{q}$ does not depend on $\boldsymbol{\beta}_N$, $\boldsymbol{\beta}_N$ can be set to zero, with the result that $\boldsymbol{\beta} = \boldsymbol{\beta}_B$. Nevertheless, when the number of variables k is larger than the number of objects n, and when $\mathbf{X}$ is of full rank, that is $r = k$, then $\mathbf{PP}' = \mathbf{I}$ and each $\mathbf{q}$ will give an unique $\boldsymbol{\beta}$. As $\dim(\mathbf{q}) < \dim(\boldsymbol{\beta})$, it is most efficient to optimize the loss function through $\mathbf{q}$. $\boldsymbol{\beta}$ can then be derived using (20), that is,

$$\boldsymbol{\beta} = \mathbf{Q}\boldsymbol{\Lambda}^{-1}\mathbf{P}'\mathbf{q} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{P}'\mathbf{P}\boldsymbol{\Lambda}^{-2}\mathbf{P}'\mathbf{q} = \mathbf{X}'(\mathbf{XX}')^{-1}\mathbf{q},$$

using the fact that $(\mathbf{XX}')(\mathbf{P}\boldsymbol{\Lambda}^{-2}\mathbf{P}')(\mathbf{XX}') = (\mathbf{P}\boldsymbol{\Lambda}^{2}\mathbf{P})(\mathbf{P}\boldsymbol{\Lambda}^{-2}\mathbf{P}')(\mathbf{P}\boldsymbol{\Lambda}^{2}\mathbf{P}) = (\mathbf{P}\boldsymbol{\Lambda}^{2}\mathbf{P})$. Note that $\boldsymbol{\beta}$ does not depend on $\mathbf{q}_N$, as $\boldsymbol{\beta} = \mathbf{Q}\boldsymbol{\Lambda}^{-1}\mathbf{P}'\mathbf{q} = \mathbf{Q}\boldsymbol{\Lambda}^{-1}\mathbf{P}'\mathbf{q}_B$. The penalty term $\lambda\boldsymbol{\beta}'\boldsymbol{\beta}$ can then be written as

$$\begin{aligned}\lambda\boldsymbol{\beta}'\boldsymbol{\beta} &= \lambda(\mathbf{X}'(\mathbf{XX}')^{-1}\mathbf{q})'(\mathbf{X}'(\mathbf{XX}')^{-1}\mathbf{q}) \\ &= \lambda\mathbf{q}'(\mathbf{XX}')^{-1}\mathbf{XX}'(\mathbf{XX}')^{-1}\mathbf{q} = \lambda\mathbf{q}'(\mathbf{XX}')^{-1}\mathbf{q} = \lambda\mathbf{q}'\mathbf{K}^{-1}\mathbf{q},\end{aligned}$$

where $\mathbf{K} = \mathbf{XX}' = \mathbf{P}\boldsymbol{\Lambda}^2\mathbf{P}'$.

Therefore, the majorization function can as follows be written as a function of $\mathbf{q}$.

$$\text{Maj}(\boldsymbol{\beta}, \alpha) = (\alpha\mathbf{1} + \mathbf{q})'\mathbf{A}(\alpha\mathbf{1} + \mathbf{q}) - 2(\alpha\mathbf{1} + \mathbf{q})'\mathbf{b} + \lambda\mathbf{q}'\mathbf{K}^{-1}\mathbf{q}. \tag{27}$$

The first-order conditions of (27) are

$$\begin{aligned}\mathbf{1}'\mathbf{A}(\mathbf{1}\alpha + \mathbf{q}) &= \mathbf{1}'\mathbf{b}, \\ \mathbf{A}(\mathbf{1}\alpha + \mathbf{q}) + \lambda\mathbf{K}^{-1}\mathbf{q} &= \mathbf{b}.\end{aligned}$$

Using $\begin{bmatrix}\mathbf{1} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\alpha & \mathbf{q}\end{bmatrix} = \mathbf{1}\alpha + \mathbf{q} = \tilde{\mathbf{q}}$, the parameters $\mathbf{q}$ and α can be updated by deriving

$$\left(\begin{bmatrix}\mathbf{1}' \\ \mathbf{I}\end{bmatrix}\mathbf{A}\begin{bmatrix}\mathbf{1} & \mathbf{I}\end{bmatrix} + \lambda\begin{bmatrix}0 & \mathbf{0}' \\ \mathbf{0} & \mathbf{K}^{-1}\end{bmatrix}\right)\begin{bmatrix}\alpha^+ \\ \mathbf{q}^+\end{bmatrix} = \begin{bmatrix}\mathbf{1}'\mathbf{b} \\ \mathbf{b}\end{bmatrix},$$

or, in compact form

$$(\tilde{\mathbf{I}}'\mathbf{A}\tilde{\mathbf{I}} + \lambda\mathbf{L})\begin{bmatrix}\alpha^+ \\ \mathbf{q}^+\end{bmatrix} = \tilde{\mathbf{I}}'\mathbf{b}, \tag{28}$$

where $\tilde{\mathbf{I}} = \begin{bmatrix}\mathbf{1} & \mathbf{I}\end{bmatrix}$ and $\mathbf{L} = \begin{bmatrix}0 & \mathbf{0}' \\ \mathbf{0} & \mathbf{K}^{-1}\end{bmatrix}$.

Similar to (25), we can show that $(\tilde{\mathbf{I}}'\mathbf{A}\tilde{\mathbf{I}} + \lambda\mathbf{L})$ is positive definite and thus (28) can always be solved.

Also, the corresponding update for $\boldsymbol{\beta}$, which is

$$\boldsymbol{\beta}^+ = \mathbf{Q}\boldsymbol{\Lambda}^{-1}\mathbf{P}'\mathbf{q} = \mathbf{X}(\mathbf{XX}')^{-1}\mathbf{q}, \tag{29}$$

equals $\boldsymbol{\beta}_B$, as

$$\boldsymbol{\beta}_N = (\mathbf{I} - \mathbf{QQ}')\boldsymbol{\beta} = (\mathbf{I} - \mathbf{QQ}')\mathbf{Q}\boldsymbol{\Lambda}^{-1}\mathbf{P}'\mathbf{q} = (\mathbf{Q} - \mathbf{QQ}'\mathbf{Q})\boldsymbol{\Lambda}^{-1}\mathbf{P}'\mathbf{q} = \mathbf{0}.$$

A.3. $\boldsymbol{\theta}$-method: Rank is smaller than either n or k ($r <$min(n, k))

When $r < \min(n, k)$, the interdependence of $\mathbf{q}$ and $\mathbf{X}$ can be summarized in an $r \times 1$ vector $\boldsymbol{\theta} = \mathbf{Q}'\boldsymbol{\beta} = \boldsymbol{\Lambda}^{-1}\mathbf{P}'\mathbf{q}$ from (20). As $\mathbf{q}_N$ and $\boldsymbol{\beta}_N$ are not of interest, it is efficient to optimize the loss function by $\boldsymbol{\theta}$. $\boldsymbol{\beta}$ will then be calculated through $\boldsymbol{\beta} = \mathbf{Q}\boldsymbol{\theta}$, which will assure that $\boldsymbol{\beta} = \mathbf{Q}\boldsymbol{\theta} = \mathbf{Q}\mathbf{Q}'\boldsymbol{\beta} = \boldsymbol{\beta}_B$, in a similar way, it can be shown that $\mathbf{q} = \mathbf{q}_B$. Using the fact that $\mathbf{q} = \mathbf{X}\boldsymbol{\beta} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{Q}'\boldsymbol{\beta} = \mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\theta}$ and $\boldsymbol{\theta}'\boldsymbol{\theta} = \boldsymbol{\beta}'\mathbf{Q}\mathbf{Q}'\boldsymbol{\beta} = \boldsymbol{\beta}_B'\boldsymbol{\beta}_B = \boldsymbol{\beta}'\boldsymbol{\beta}$, the majorization function can be written as

$$\mathrm{Maj}(\boldsymbol{\beta}, \alpha) = (\alpha\mathbf{1} + \mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\theta})'\mathbf{A}(\alpha\mathbf{1} + \mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\theta}) - 2(\alpha\mathbf{1} + \mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\theta})'\mathbf{b} + \lambda\boldsymbol{\theta}'\boldsymbol{\theta}, \tag{30}$$

with the first-order condition

$$\begin{gathered}\mathbf{1}'\mathbf{A}(\mathbf{1}\alpha + \mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\theta}) = \mathbf{1}'\mathbf{b}\\ \boldsymbol{\Lambda}\mathbf{P}'\mathbf{A}(\mathbf{1}\alpha + \mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\theta}) + \lambda\mathbf{I}\boldsymbol{\theta} = \boldsymbol{\Lambda}\mathbf{P}'\mathbf{b}.\end{gathered}$$

Using $\begin{bmatrix}\mathbf{1} & \mathbf{P}\boldsymbol{\Lambda}\end{bmatrix}\begin{bmatrix}\alpha & \boldsymbol{\theta}\end{bmatrix} = \mathbf{1}\alpha_0 + \mathbf{P}\boldsymbol{\Lambda}\boldsymbol{\theta} = \tilde{\mathbf{q}}$, the parameters $\boldsymbol{\theta}$ and α_0 can be updated by deriving

$$\left(\begin{bmatrix}\mathbf{1}'\\ \boldsymbol{\Lambda}\mathbf{P}'\end{bmatrix}\mathbf{A}\begin{bmatrix}\mathbf{1} & \mathbf{P}\boldsymbol{\Lambda}\end{bmatrix} + \lambda\begin{bmatrix}0 & \mathbf{0}'\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\right)\begin{bmatrix}\alpha^+\\ \boldsymbol{\theta}^+\end{bmatrix} = \begin{bmatrix}\mathbf{1}'\mathbf{b}\\ \boldsymbol{\Lambda}\mathbf{P}'\mathbf{b}\end{bmatrix},$$

or, in compact form

$$(\widetilde{\mathbf{P}}'\mathbf{A}\widetilde{\mathbf{P}} + \lambda\mathbf{J})\begin{bmatrix}\alpha^+\\ \boldsymbol{\theta}^+\end{bmatrix} = \widetilde{\mathbf{P}}'\mathbf{b}, \tag{31}$$

where $\widetilde{\mathbf{P}} = \begin{bmatrix}\mathbf{1} & \mathbf{P}\boldsymbol{\Lambda}\end{bmatrix}$.

The advantage of this method is that $r \leq \min(n, k)$, which means that it restricts to the space of which the relationship between $\mathbf{q}$ and $\boldsymbol{\beta}$ is described. Moreover, it is assured that the dimension of $\boldsymbol{\theta}$ is the lowest of three (that is, $r \leq \min(n, k)$), and thus it is most efficient and consistent algorithm. However, this method requires the SVD or the QR decomposition to be computed, which may need much computational time in case n and k are large. Therefore, one should consider the alternatives when the matrix $\mathbf{X}$ is of full rank, that is when $r = \min(n, k)$.

References

Bekkerman R, Allan J (2003). “Using Bigrams in Text Categorization.”

Borg I, Groenen P (2005). *Modern multidimensional scaling*. Springer-Verlag New York, Inc.

Boser BE, Guyon IM, Vapnik VN (1992). “A training algorithm for optimal margin classifiers.” In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT ’92, pp. 144–152. ACM, New York, NY, USA.

Bradley AP (1997). “The use of the area under the ROC curve in the evaluation of machine learning algorithms.” *Pattern Recogn.*, **30**(7), 1145–1159.

Burges CJC (1998). “A Tutorial on Support Vector Machines for Pattern Recognition.” *Data Min. Knowl. Discov.*, **2**, 121–167.

Chang CC, Lin CJ (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chapelle O, Haffner P, Vapnik V (1999). “Support vector machines for histogram-based image classification.” *IEEE Transactions on Neural Networks*, **10**(5), 1055–1064.

Collobert R, Bengio S (2001). "SVMTorch: support vector machines for large-scale regression problems." *Journal of Machine Learning Research*, **1**, 143–160.

Cui D, Curry D (2005). "Prediction in Marketing Using the Support Vector Machine." *Marketing Science*, **24**(4), 595–615.

De Leeuw J (1994). "Block relaxation algorithms in statistics." In H-H Bock, W Lenski, MM Richter (eds.), *Information systems and data analysis*, pp. 308–324. Springer-Verlag, Berlin.

De Leeuw J, Heiser W (1980). *Multidimensional scaling with restrictions on the configuration*, volume 5, pp. 285–317. Amsterdam, North-Holland.

de Leeuw J, Mair P (2009). "Multidimensional Scaling Using Majorization: **SMACOF** in R." *Journal of Statistical Software*, **31**(3), 1–30.

Drucker H, Wu D, Vapnik VN (1999). "Support vector machines for spam categorization." *IEEE TRANSACTIONS ON NEURAL NETWORKS*, **10**(5), 1048–1054.

Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008). "LIBLINEAR: A Library for Large Linear Classification." *Journal of Machine Learning Research*, **9**, 1871–1874.

Furey TS, Cristianini N, Duffy N, Bednarski DW, Schummer M, Haussler D (2000). "Support vector machine classification and validation of cancer tissue samples using microarray expression data." *Bioinformatics*, **16**(10), 906–914.

Groenen PJF, Nalbantov G, Bioch JC (2007). "Nonlinear Support Vector Machines Through Iterative Majorization and I-Splines." In *Advances in Data Analysis*, Studies in Classification, Data Analysis, and Knowledge Organization, pp. 149–161. Springer-Verlag Berlin Heidelberg.

Groenen PJF, Nalbantov GI, Bioch JC (2008). "SVM-Maj: a majorization approach to linear support vector machines with different hinge errors." *Advances in Data Analysis and Classification*, **2**(1), 17–43.

Guyon I, Elisseeff A (2003). "An introduction to variable and feature selection." *J. Mach. Learn. Res.*, **3**, 1157–1182.

Guyon I, Weston J, Barnhill S, Vapnik V (2002). "Gene Selection for Cancer Classification using Support Vector Machines." *Machine Learning*, **46**, 389–422.

Heiser WJ (1995). *Convergent computation by iterative majorization: Theory and applications in multidimensional data analysis*, pp. 157–189. Oxford University Press, Oxford.

Hunter DR, Lange K (2004). "A tutorial on MM algorithms." *The American Statistician*, **39**, 30–37.

Joachims T (1998). "Text Categorization with Suport Vector Machines: Learning with Many Relevant Features." In *Proceedings of the 10th European Conference on Machine Learning*, pp. 137–142. Springer-Verlag, London, UK. ISBN 3-540-64417-2.

Joachims T (1999). *Making large-scale support vector machine learning practical*, pp. 169–184. MIT Press, Cambridge, MA, USA.

Joachims T (2006). "Training linear SVMs in linear time." In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pp. 217–226. ACM, New York, NY, USA.

Kiers HAL (2002). "Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems." *Computational Statistics and Data Analysis*, **41**, 157–170.

Kieskompas (2010). "Tweede Kamerverkiezingen 2010." URL http://www.tweedekamer2010.kieskompas.nl/.

Lange K, Hunter DR, Yang I (2000). "Optimization transfer using surrogate objective functions." *Journal of Computational and Graphical Statistics*, **9**, 1–20.

McCallum A, Nigam K (1998). "A comparison of event models for Naive Bayes text classification." In *Proceedings of the ninth international conference on Information and knowledge management*, AAAI/ICML-98 Workshop on Learning for Text Categorization, pp. 41–48.

Murtagh B, Saunders M (1998). "**MINOS** 5.5: User's Guide." *Technical Report SOL 83-20R*, Dept. of Operations Research, Stanford University.

Osuna E, Freund R, Girosi F (1997a). "An Improved Training Algorithm for Support Vector Machines." IEEE Workshop on Neural Networks and Signal Processing. IEEE Press.

Osuna E, Freund R, Girosi F (1997b). "Support vector machines: Training and applications." *Technical Report AIM-1602*, MIT Artificial Intelligence Laboratory.

Pang B, Lee L, Vaithyanathan S (2002). "Thumbs up?: sentiment classification using machine learning techniques." In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - VOLUME 10*, EMNLP '02, pp. 79–86. Association for Computational Linguistics, Stroudsburg, PA, USA.

Platt J (1999). *Fast training of support vector machines using sequential minimal optimization*, chapter 12, pp. 185–208. MIT Press, Cambridge, MA, USA.

Polanyi L, Zaenen A (2005). "Contextual valence shifters." In *Computing Attitude and Affect in Text*. Springer.

Pontil M, Verri A (1998). "Support Vector Machines for 3D Object Recognition." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, pp. 637–646. IEEE Computer Society, Los Alamitos, CA, USA.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Ramsay J (1988). "Monotone regression splines in action. Statistical Science." **3**(4), 425–461.

Rüping S (2000). *mySVM-Manual*. Software available at http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.

Sahami M, Dumais S, Heckerman D, Horvitz E (1998). "A Bayesian Approach to Filtering Junk E-Mail."

Saunders C, Stitson M, Weston J, Léon B, Schölkopf B, Smola A (1998). "Support vector machine reference manual." *Technical Report CSD-TR-98-03*, Department of Computer Science, Royal Holloway, University of London.

Sebastiani F (2005). "Text categorization." In *Text Mining and its Applications to Intelligence, CRM and Knowledge Management*, pp. 109–129. WIT Press.

Seber G, Wild C (1989). *Nonlinear Regression.* John Wiley and Sons, New York.

Strapparava C, Guerini M, Stock O (2010). "Predicting Persuasiveness in Political Discourses." In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10).* European Language Resources Association (ELRA), Valletta, Malta.

Thomas M, Pang B, Lee L (2006). "Get out the vote: Determining support or opposition from Congressional floor-debate transcripts." In *Proceedings of EMNLP*, pp. 327–335.

Tweede Kamer der Staten-Generaal (2012). URL http://www.tweedekamer.nl.

Vanderbei RJ (1994). "Interior-point methods: Algorithms and formulations." *Informs Journal on Computing*, **6**(1), 32–34.

Vapnik VN (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc. 2nd edition, 2000.

Wu HC, Luk RWP, Wong KF, Kwok KL (2008). "Interpreting TF-IDF term weights as making relevance decisions." *ACM Trans. Inf. Syst.*, **26**(3), 13:1–13:37.