RUTCOR
RESEARCH
REPORT

Sparse Signomial Classification and Regression

Kyungsik Lee[a] Norman Kim[b] Myong K. Jeong[c]

RRR 1-2010, January 13, 2010

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/~rrr

[a]Department of Industrial & Management Engineering, Hankuk University of Foreign Studies, San 89 Yongin-si, Kyunggi-do, 449-791 Korea. globaloptima@hufs.ac.kr

[b]RUTCOR Rutgers, the State University of New Jersey 640 Bartholomew Road Piscataway, NJ 08854-8003. normank@rci.rutgers.edu

[c]RUTCOR Rutgers, the State University of New Jersey 640 Bartholomew Road Piscataway, NJ 08854-8003. mjeong@rci.rutgers.edu

Rutcor Research Report
RRR 1-2010, January 13, 2010

Sparse Signomial Classification and Regression

Kyungsik Lee Norman Kim Myong K. Jeong

Abstract. In this paper, we propose the sparse signomial classification and regression (SSCR), a new supervised learning method for classification and regression. SSCR seeks a sparse signomial function by solving a linear program to minimize the weighted sum of the ℓ_1-norm of the coefficient vector of the function and the ℓ_1-norm of violation (or loss) caused by the function. SSCR gives an explicit description of the resulting function in the original input space, which can be used for prediction purposes as well as interpretation purposes. Moreover, this method can explore very high dimensional feature spaces with less sensitivity to numerical values or numeric ranges of the given data. We present a practical implementation of SSCR based on the column generation. Computational study shows that SSCR gives competitive or even better performance compared to other widely used learning methods for classification and regression.

Acknowledgements: This research was done during the first author's visit to RUTCOR in 2009 which was supported by Hankuk University of Foreign Studies Research Fund.

1 Introduction

The *supervised learning* is a machine learning technique to deduce a hidden function by investigating a number of data each of which can be represented as a pair of an input vector and the corresponding output value, which is to be used to predict the output values for new (unforeseen) input vectors. Two central topics in the development of various supervised learning methods for the past 20 years are *data classification* and *regression* problems. These two problems have been studies by diverse groups of researchers including statisticians and computer scientists, so that a number of popular supervised learning methods such as CART (classification and regression trees) for classification and regression [1, 11], MARS (multivariate adaptive regression splines) for regression [6], and SVMs (support vector machines) for classification and regression [22] have been proposed.

Among them, SVM has been received considerable attention and have shown promising empirical results in many practical applications. SVM is originally developed for binary (two-class) classification and separates points of different classes by a single hyperplane (an affine function) with a maximal margin [22]. SVMs can explore the nonlinearity of data patterns in a relatively flexible manner by means of various kernel functions, which make it possible to find a separating hyperplane in a higher dimensional space induced by a kernel function that correspond to a nonlinear classifier in the original input space. SVM for regression, which is called support vector regression (SVR), has been also devised with the introduction of an ϵ-insensitive loss function [22]. For comprehensive tutorials on SVM, we refer the readers to [2, 7, 20].

In SVMs for classification and regression, a classifier (or a regression function) is produced in the form of a kernel expansion, which is a sum of a linear combination of kernel functions and a constant term. For the purpose of improving the generalization performance and the calculation time involved in the prediction, noble and computationally efficient SVMs to get a sparser kernel expansion [10, 13, 14] and methods to get a sparse affine function in the high dimensional spaces induced by a kernel [23] have been proposed. However, it is not easy to get an explicit function description in the original input space if we use a nonlinear kernel. Thus we can use the resulting functions for prediction purposes but less easily for interpretation purposes.

In addition, in SVMs, the selection of an appropriate kernel type among many possible kernel types along with the associated parameters is a big deal. Even if a strong theoretical method for selecting a kernel is developed, unless this can be validated using independent test sets on a large number of problems, methods such as bootstrapping and cross-validation will remain the preferred method for kernel selection [7]. That is, in most cases, the only realistic approach to the decision on the kernel type and associated parameters is trial and error. However, this approach is time-consuming and does not guarantee that a kernel that was not considered in bootstrapping or cross-validation would not perform better.

The purpose of this paper is to present a new supervised learning method for classification and regression, which we call the *sparse signomial classification and regression* (SSCR), to get a sparse signomial function, which can be used for prediction purposes as well as the

interpretation purposes. The proposed method give an explicit description of the resulting function in the original input space while exploring the nonlinearity of the given data without using nonlinear kernels.

This paper is organized as follows. In section 2, we present the essential ideas of SSCR for both classification and regression along with the motivation and significance of the proposed method. A practical implementation of SSCR is given in detail in section 3. The performance of SSCR is evaluated in a computational study given in section 4. Finally, concluding remarks are given in section 5.

2 An Overview of SSCR

In this section, we present the basic ideas of SSCR first with respect to classification and then with respect to regression.

Let $x_1, \ldots, x_n$ denote n real positive variables, and $\mathbf{x} = (x_1, \ldots, x_n)$ a vector with components x_j. Let $g_{\mathbf{d}}(\mathbf{x})$ be a real valued function of $\mathbf{x}$ of the form $g_{\mathbf{d}}(\mathbf{x}) = \Pi_{j=1}^{n} x_j^{d_j}$, where $\mathbf{d} = (d_1, \ldots, d_n)$ is a real vector with components d_j. A real valued function f of $\mathbf{x}$, with the form

$$f(\mathbf{x}) = \sum_{\mathbf{d} \in D} w_{\mathbf{d}} g_{\mathbf{d}}(\mathbf{x}) + b, \tag{1}$$

where $b \in \mathbb{R}$, D is a finite subset of $\mathbb{R}^n$ with $0 \notin D$, and $w_{\mathbf{d}} \in \mathbb{R}$ for all $\mathbf{d} \in D$, is called a *signomial* function in the variables $x_1, \ldots, x_n$. If $D = \{\mathbf{d} \in \mathbb{Z}_+^n : 1 \leq \sum_{j=1}^{n} d_j \leq k\}$, $f(\mathbf{x})$ is a polynomial function of degree less than or equal to a given positive integer k. $f(\mathbf{x})$ also can be viewed as an affine function $h(\mathbf{z}) = \mathbf{w} \cdot \mathbf{z} + b$ in $\mathbb{R}^{|D|}$ space, where $\mathbf{z}$ is a vector with components $z_{\mathbf{d}} = g_{\mathbf{d}}(\mathbf{x})$ for all $\mathbf{d} \in D$.

2.1 SSCR for Classification

We are given two sets of points in $\mathbb{R}_{++}^n$ (the set of n-dimensional positive real vectors), $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$, and wish to find a signomial function $f(\mathbf{x})$ defined as (1) such that

$$f(\mathbf{x}_i) > 0, i = 1, \ldots, N, \qquad f(\mathbf{y}_i) < 0, i = 1, \ldots, M. \tag{2}$$

Since the strict inequalities (2) are homogeneous in $\mathbf{w}$ and b, they are satisfied if and only if there exists a function f of the form (1) such that

$$f(\mathbf{x}_i) \geq 1, i = 1, \ldots, N, \qquad f(\mathbf{y}_i) \leq -1, i = 1, \ldots, M. \tag{3}$$

Our goal is to find a function f of the form (1) that satisfies (3) while keeping the number of terms (the number of nonzero $w_{\mathbf{d}}$'s) as small as possible. Specifically, our approach to this goal is to minimize $\|\mathbf{w}\|_1$, which may yield a sparse $\mathbf{w}$, where $\|\cdot\|_1$ is the ℓ_1-norm of a

vector. Geometrically, to minimize the ℓ_1-norm of the coefficient vector of $f(\mathbf{x})$ is equivalent to maximize the so called margin measured by the ℓ_∞-norm (the distance measured by the ℓ_∞-norm between two hyperplanes $\{\mathbf{z} : \mathbf{w}\cdot\mathbf{z}+b = 1\}$ and $\{\mathbf{z} : \mathbf{w}\cdot\mathbf{z}+b = -1\}$ in the $\mathbf{z}$-space), which is given by $2/\|\mathbf{w}\|_1$ [12]. If we use the ℓ_2-norm instead of the ℓ_1-norm, the meaning of the margin is the same as that used in standard support vector machine (SVM) [22].

In general, it may not be possible to find f of the form (1) that satisfies (3) because the set D is finite. Therefore, we seek a function of the form (1) that approximately separates the points, which is a function that satisfies (4) for some nonnegative variables $u_1, \ldots, u_N$ and $v_1, \ldots, v_M$.

$$f(\mathbf{x}_i) \geq 1 - u_i, i = 1, \ldots, N, \qquad f(\mathbf{y}_i) \leq -1 + v_i, i = 1, \ldots, M. \tag{4}$$

The values of u_i's and v_i's can be thought as a measure of how much the condition (4) is violated by the function, so that we want to find $f(\mathbf{x})$ that keeps $\|\mathbf{u}\|_1 + \|\mathbf{v}\|_1$ as small as possible. In SSCR for classification, we consider the trade-off between the margin defined by $2/\|\mathbf{w}\|_1$ (which we want to maximize), and the amount of violation measured by $\|\mathbf{u}\|_1 + \|\mathbf{v}\|_1$ (which we want to minimize).

The *sparse signomial classifiers* obtained by SSCR for the sets $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$ are defined as the solution of the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{w}\|_1 + C(\|\mathbf{u}\|_1 + \|\mathbf{v}\|_1) \\ \text{subject to} \quad & \sum_{\mathbf{d}\in D} g_{\mathbf{d}}(\mathbf{x}_i) w_{\mathbf{d}} + b \geq 1 - u_i, i = 1, \ldots, N \\ & \sum_{\mathbf{d}\in D} g_{\mathbf{d}}(\mathbf{y}_i) w_{\mathbf{d}} + b \leq -1 + v_i, i = 1, \ldots, M \\ & \mathbf{w} \in \mathbb{R}^{|D|}, b \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^N_+, \mathbf{v} \in \mathbb{R}^M_+ . \end{aligned} \tag{5}$$

The parameter C, which is a positive real number, gives the relative weight of the amount of violation compared to the margin. In the problem (5), the set D is prespecified by using the four parameters, d_{max}, d_{min}, T, and L, as follows:

$$D = \{\mathbf{d} \in \mathbb{R}^n : d_{min} \leq d_i \leq d_{max}, i = 1, \ldots, n, \sum_{i=1}^{n} |d_i| \leq L, T\mathbf{d} \in \mathbb{Z}^n\}, \tag{6}$$

where $T > 0$ and $L > 0$. If we set $d_{max} = k, d_{min} = 0, T = 1$, and $L = k$ for some positive integer k, then the function obtained by the solution of the problem (5) is a polynomial function of degree less than or equal to k.

The problem (5) can be readily reformulated as a linear program (LP). However, depending on the four parameters of the set D, the number of elements of D can be huge, which means there may be huge number of variables in the LP. Therefore, in our implementation of SSCR, we devise an algorithm based on the column generation technique [4] to handle this issue, which will be presented in detail in section 3.

2.2 SSCR for Regression

We are given N data points $(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^n_{++}, y_i \in \mathbb{R}$, and $i = 1, \ldots, N$, and wish to find a regression function, which is a signomial function of the form (1), that minimizes the regularized risk functional [20, 22]. As in SVMs for regression, we used the ϵ-insensitive loss function $\mathcal{L}_\epsilon(\mathbf{x}, y) = \max\{|y - f(\mathbf{x})| - \epsilon, 0\}$. Instead of using the ℓ_2-norm of the coefficient vector $\mathbf{w}$ used in standard SVM, we used the ℓ_1-norm for the regularization. The regularized risk functional is then specified as

$$\|\mathbf{w}\|_1 + C \sum_{i=1}^{n} \mathcal{L}_\epsilon(\mathbf{x}_i, y_i),$$

where C and ϵ are positive constants given as parameters. Then, the regression function generated by SSCR is given by the solution of the following optimization problem in which the set of exponent vectors D is specified in the same way as in the problem (5).

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{w}\|_1 + C(\|\mathbf{u}\|_1 + \|\mathbf{v}\|_1) \\ \text{subject to} \quad & y_i - \sum_{\mathbf{d} \in D} g_{\mathbf{d}}(\mathbf{x}_i) w_{\mathbf{d}} - b \leq \epsilon + u_i, i = 1, \ldots, N \\ & \sum_{\mathbf{d} \in D} g_{\mathbf{d}}(\mathbf{x}_i) w_{\mathbf{d}} + b - y_i \leq \epsilon + v_i, i = 1, \ldots, N \\ & \mathbf{w} \in \mathbb{R}^{|D|}, b \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^N_+, \mathbf{v} \in \mathbb{R}^N_+. \end{aligned} \tag{7}$$

The above problem (7) also can be transformed into an LP with (possibly) exponentially many variables depending on the parameters of D, so that we use the same approach to get a solution to the problem as for the problem (5).

To sum up, in SSCR, we used the same approach to classification and regression, that is, we seek a sparse signomial function of the form (1) by solving an LP to minimize the weighted sum of the ℓ_1-norm of the coefficient vector of the function and the ℓ_1-norm of violation (or loss) caused by the function.

In general, the given points $\mathbf{x}_i$'s and $\mathbf{y}_i$'s ($\mathbf{x}_i$'s in the regression case) are not in the positive orthant, $\mathbb{R}^n_{++}$. Therefore, before solving LPs, we perform a preprocessing step to make sure that they are in $\mathbb{R}^n_{++}$ by translating all the points. Note that this translation of the given data points does not cause any loss of generality.

2.3 Motivation and Significance

In SVMs for classification and regression [7, 10, 13, 20, 22, 23], classifiers (or regression functions) are produced by mapping $\mathbf{x} \in \mathbb{R}^n$ to $\Phi(\mathbf{x})$ in a (possibly) high dimensional feature space via a kernel κ and constructing affine functions in the feature space, which can be expressed as a kernel expansion $f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) + \sum_{i=1}^{M} \beta_i \kappa(\mathbf{y}_i, \mathbf{x}) + b$ ($f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) + b$ in the regression case), where α_i's and β_i's are real numbers. That is, a function $f(\mathbf{x})$ produced by SVMs is expressed as a sum of a linear combination of kernel functions and a constant

term b. As we mentioned in section 1, it is not easy to explicitly look at the resulting function in the original input space if we use nonlinear kernel functions such as polynomial kernels, RBF kernels, and so on. Thus we can use the resulting function for prediction purposes but less easily for interpretation regarding which original input variables and/or feature variables in the feature space are more important (or meaningful) than others.

The initial motivation of SSCR was to develop a practical method to get an explicit description of a sparse polynomial function (possibly of a high order) in the original input space, which can be used for prediction purposes as well as interpretation purposes. One way to do this, of course, is to first get a function in a kernel expansion form by applying SVMs with a polynomial kernel of degree p, and then to expand it to get the corresponding explicit function in the original space, which takes at least $O(n^p)$ calculations. This naive approach, though not impossible, is apparently impractical if p is a very large number.

In addition to the interpretability of the resulting classifiers or regression functions, we soon discovered that SSCR can provide additional advantages over SVMs by employing signomial functions instead of confining the function form to polynomial functions. With very high order polynomial kernels or RBF kernels, SVMs can explore very high dimensional spaces. However, depending on the magnitude and numeric range of the given data points, the resulting kernel matrices may be ill-conditioned or almost identical to the identity matrix, so that learning algorithms may be suffered from numerical instability or the resulting functions may show poor prediction performance. This is especially true when some variables of the given data points are in greater numeric ranges or have much larger (or smaller) absolute values than other variables. Data scaling (normalization) generally helps to handle this kind of issues but may not be always effective.

Suppose that we specify the set D defined as (6) with parameters $d_{min} = 0, d_{max} = 1, T = 100$, and $L = 1$. Then, a signomial function obtained by SSCR can be thought as a polynomial function of degree less than or equal to 100, which can be obtained after transforming each data points $\mathbf{x} \in \mathbb{R}^n_{++}$ to $\bar{\mathbf{x}} \in \mathbb{R}^n_{++}$ such that $\bar{x}_i = \sqrt[100]{x_i}, i = 1, \ldots, n$. This observation show that SSCR has an inherent nonlinear data scaling capability which implicitly transform each given data point $\mathbf{x} \in \mathbb{R}^n_{++}$ to $\bar{\mathbf{x}} \in \mathbb{R}^n_{++}$ such that $\bar{x}_i = \sqrt[T]{x_i}, i = 1, \ldots, n$, which transform each data point into a point around the n-dimensional vector of the 1's as T becomes larger. Therefore, by allowing exponents of variables in each term of the function to be fractional, SSCR can explore very high dimensional feature spaces with less sensitivity to numeric ranges and numerical values of the given data points, which make it possible to approximate (to describe approximately) any continuous differentiable functions in $\mathbb{R}^n_{++}$. Further, if we allow those exponents to be negative, more accurate function description would be possible.

In SVMs, the selection of an appropriate kernel type and values of the parameters of the selected kernel according to the characteristics of the given data is crucial, by which the performance of the resulting functions is heavily influenced. In our SSCR, the parameters of the set D also need to be appropriately specified. The selected values of those four parameters, of course, may have great impacts on the performance of the resulting functions. However, our computational study given in section 4 shows that classifiers and regression functions

obtained by SSCR, with $d_{min} = -1, d_{max} = 1, T = 100$, and $L = 1$, show competitive training and prediction (test) accuracy for all tested real world data sets. This shows that the parameter selection decision for SSCR can be less complex than those decisions for SVMs.

Along with those good characteristics of SSCR, there is a difficulty in implementing this method. As mentioned earlier, problem (5) and (7) are linear programs (possibly) with a huge number of variables, which are proved to be generally $\mathcal{NP}$-hard unfortunately. Hence, in our implementation given in the next section, we devise heuristic algorithms to get approximate solutions to these problems. This difficulty in implementation could be one of demerits of SSCR. However, even though it is difficult to get optimal solutions to these LPs, classifiers and regression functions obtained by the current implementation of SSCR in our computational study show competitive or even better performance with respect to stability and prediction accuracy compared to the other widely used classification and regression methods.

3 An Implementation of SSCR

In this section, we will show how the problems (5) and (7) can be approached using the (*delayed*) *column generation* technique [4], along with detailed description of algorithmic components of SSCR. The column generation technique to solve linear programs (especially with a huge number of variables) is a classical technique in mathematical programming in that it has been theoretically well-studied and widely applied to many applications. For more theoretical treatments, we refer the readers to [4].

3.1 An Implementation of SSCR for Classification

As mentioned in the previous section, classifiers are obtained by solving the problem (5) in SSCR, which can be reformulated into an LP as the following MPC by rewriting $w_{\mathbf{d}} = w^+_{\mathbf{d}} - w^-_{\mathbf{d}}$ where $w^+_{\mathbf{d}}, w^-_{\mathbf{d}} \geq 0$.

$$
\begin{aligned}
\text{(MPC) minimize} \quad & \sum_{\mathbf{d}\in D}(w^+_{\mathbf{d}} + w^-_{\mathbf{d}}) + C(\sum_{i=1}^{N} u_i + \sum_{i=1}^{M} v_i) \\
\text{subject to} \quad & \sum_{\mathbf{d}\in D} g_{\mathbf{d}}(\mathbf{x}_i)(w^+_{\mathbf{d}} - w^-_{\mathbf{d}}) + b \geq 1 - u_i, i = 1, \ldots, N \qquad (8) \\
& \sum_{\mathbf{d}\in D} g_{\mathbf{d}}(\mathbf{y}_i)(w^+_{\mathbf{d}} - w^-_{\mathbf{d}}) + b \leq -1 + v_i, i = 1, \ldots, M \qquad (9) \\
& \mathbf{w}^+, \mathbf{w}^- \in \mathbb{R}^{|D|}_+, b \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^N_+, \mathbf{v} \in \mathbb{R}^M_+ .
\end{aligned}
$$

We refer the problem MPC as the *master problem* for classification. The solution has either $w^+_{\mathbf{d}}$ or $w^-_{\mathbf{d}}$ equal to zero, so $|w_{\mathbf{d}}| = w^+_{\mathbf{d}} + w^-_{\mathbf{d}}$.

Depending on the four parameters of D defined as (6), MPC may have a huge number of variables. Rather than the brute force approach of enumerating all the elements of D, we

will generate them "as we go along". To this end, we consider a subset of D, $I \subset D$. We refer the associated restriction of MPC to I, which is an LP obtained by replacing D with I in the formulation of MPC, as the *restricted master problem* for classification (RMPC), whose optimal solution gives a suboptimal solution to MPC together with an upper bound to the optimal value of MPC. The key idea of the column generation technique is to successively improve the upper bound by adding a new exponent vector $\mathbf{d} \in D \setminus I$ to RMPC. Adding a new exponent vector makes I a better approximation of D and eventually allows the optimal solution to RMPC to be optimal to MPC in finite iterations.

Given an initial subset I of D, we solve RMPC using the simplex algorithm [4], a well-known method for solving linear programs, which yields an optimal solution to RMPC together with an optimal solution to the dual of RMPC. Then, using the optimal dual solution, we search for a *profitable* exponent vector $\mathbf{d} \in D \setminus I$ whose addition to RMPC may result in the decrease of the optimal objective value of RMPC. If there is no such $\mathbf{d} \in D \setminus I$, the optimal solution to RMPC at hand is also an optimal solution to MPC. Otherwise, we update I and then repeat the above process.

The key step of the above process is to check whether there exists a profitable $\mathbf{d} \in D \setminus I$ or not. We can formalize this issue as another optimization problems, the so called *column generation problems*. Let α_i for all $i = 1, \ldots, N$ and β_i for all $i = 1, \ldots, M$ be the nonnegative dual variables associated with constraints (8) and (9), respectively. Then, the constraints in the dual of MPC corresponding to the variables $w^+_{\mathbf{d}}$'s and $w^-_{\mathbf{d}}$'s are

$$\sum_{i=1}^{N} \alpha_i g_{\mathbf{d}}(\mathbf{x}_i) - \sum_{i=1}^{M} \beta_i g_{\mathbf{d}}(\mathbf{y}_i) \leq 1, \mathbf{d} \in D,$$
$$\sum_{i=1}^{N} \alpha_i g_{\mathbf{d}}(\mathbf{x}_i) - \sum_{i=1}^{M} \beta_i g_{\mathbf{d}}(\mathbf{y}_i) \geq -1, \mathbf{d} \in D.$$

Let $(\hat{\alpha}, \hat{\beta})$ be an optimal solution to the dual of RMPC. Then, it is optimal to the dual of MPC if and only if $(\hat{\alpha}, \hat{\beta})$ satisfies the above constraints. Therefore, we may write the optimality condition for MPC as follows.

$$\max\{z(\mathbf{d}) : \mathbf{d} \in D\} \leq 1, \quad \min\{z(\mathbf{d}) : \mathbf{d} \in D\} \geq -1, \tag{10}$$

where $z(\mathbf{d}) = \sum_{i=1}^{N} \hat{\alpha}_i g_{\mathbf{d}}(\mathbf{x}_i) - \sum_{i=1}^{M} \hat{\beta}_i g_{\mathbf{d}}(\mathbf{y}_i)$.

From the condition (10), we can check if the current optimal solution to RMPC is optimal to MPC by solving two optimization problems, which we call SPC1 and SPC2, respectively. SPC1 is the problem of maximizing $z(\mathbf{d})$ over all $\mathbf{d} \in D$ and SPC2 is the exactly the same problem as SPC1 except that the objective function is to be minimized. Let $\hat{z}^U$ and $\hat{z}^L$ be the optimal objective value of SPC1 and SPC2, respectively. Then, the optimality condition (10) is satisfied if and only if $\hat{z}^U \leq 1$ and $\hat{z}^L \geq -1$.

Now, we present our implementation of the algorithm, SPARSESIGNOMIALCLASSIFIER, to solve the problem (5) in Algorithm 1. The column generation algorithm, GENERATE,

which is used to solve SPC1 and SPC2 as a subroutine in SPARSESIGNOMIALCLASSIFIER is given in the next subsection.

Algorithm 1 Algorithm for classification based on the column generation

Given: $S_1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, S_2 = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}, d_{min}, d_{max}, L, T$, and C.
procedure SPARSESIGNOMIALCLASSIFIER
  $I \leftarrow \emptyset, J \leftarrow \emptyset$;
  repeat
    $r \leftarrow 0$;
    Formulate RMPC with I and solve it using the simplex algorithm;
    $(\hat{\mathbf{w}}^+, \hat{\mathbf{w}}^-, \hat{b}, \hat{\mathbf{u}}, \hat{\mathbf{v}}) \leftarrow$ the optimal solution to RMPC;
    $(\hat{\alpha}, \hat{\beta}) \leftarrow$ the optimal solution to the dual of RMPC;
    Solve SPC1 using GENERATE$(S_1, \hat{\alpha}, S_2, \hat{\beta}, d_{min}, d_{max}, L, T)$;
    $\hat{\mathbf{d}}^U \leftarrow$ the obtained solution to SPC1;
    $\hat{z}^U \leftarrow$ the corresponding objective value to $\hat{\mathbf{d}}^U$;
    if $\hat{z}^U \leq 1$ **then**
      Solve SPC2 using GENERATE$(S_2, \hat{\beta}, S_1, \hat{\alpha}, d_{min}, d_{max}, L, T)$;
      $\hat{\mathbf{d}}^L \leftarrow$ the obtained solution to SPC2;
      $\hat{z}^L \leftarrow$ the corresponding objective value to $\hat{\mathbf{d}}^L$;
      if $\hat{z}^L < -1$ **then**
        $I \leftarrow I \cup \{\hat{\mathbf{d}}^L\}$;
        $r \leftarrow 1$;
      end if
    else
      $I \leftarrow I \cup \{\hat{\mathbf{d}}^U\}$;
      $r \leftarrow 1$;
    end if
  until $r = 0$
  $\hat{\mathbf{w}} \leftarrow (\hat{\mathbf{w}}^+ - \hat{\mathbf{w}}^-)$;
  $J \leftarrow \{\mathbf{d} \in I | \hat{w}_{\mathbf{d}} \neq 0\}$;
  Return the classifier $f(\mathbf{x}) = \sum_{\mathbf{d} \in J} \hat{w}_{\mathbf{d}} g_{\mathbf{d}}(\mathbf{x}) + \hat{b}$;
end procedure

3.2 Column Generation Algorithm

Before presenting the column generation algorithm, we first formally define the column generation problem. Then, we analyze the computational complexity of the problem. After that we present the details of the algorithm which is based on the Frank-Wolfe algorithm [5].

For given finite subsets of $\mathbb{R}^n_{++}$, S_1 and S_2, and nonnegative real numbers $a_i(\mathbf{x})$ for all $\mathbf{x} \in S_i$, $i = 1, 2$, together with the four parameters of D given as (6), consider the following

optimization problem which we call CGP.

$$\begin{aligned} \text{(CGP) maximize} \quad & \sum_{\mathbf{x}\in S_1} a_1(\mathbf{x}) \prod_{j=1}^{n} x_j^{d_j} - \sum_{\mathbf{x}\in S_2} a_2(\mathbf{x}) \prod_{j=1}^{n} x_j^{d_j} \\ \text{subject to} \quad & d_{min} \le d_j \le d_{max}, j = 1, \ldots, n \\ & \sum_{j=1}^{n} |d_j| \le L, \\ & Td_j \in \mathbb{Z}, j = 1, \ldots, n. \end{aligned}$$

If we set $S_1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $S_2 = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$, $a_1(\mathbf{x}_i) = \hat{\alpha}_i, i = 1, \ldots, N$, and $a_2(\mathbf{y}_i) = \hat{\beta}_i, i = 1, \ldots, M$, then CGP is the same problem as SPC1. Also, if we switch those two sets S_1 and S_2, and set $a_1(\mathbf{y}_i) = \hat{\beta}_i, i = 1, \ldots, M$, and $a_2(\mathbf{x}_i) = \hat{\alpha}_i, i = 1, \ldots, N$, then CGP is the same problem as SPC2. Therefore, if we can solve CGP, then we can optimize SPC1 and SPC2.

Theoretically, MPC can be solved in polynomial time if and only if CGP is polynomially solvable according to *the polynomial equivalence between optimization and separation for rational polyhedra* [19]. Unfortunately, it turns out that CGP is generally $\mathcal{NP}$-hard, which implies MPC, the problem (5) equivalently, is $\mathcal{NP}$-hard.

Theorem 1 *CGP is $\mathcal{NP}$-hard.*

Proof The result can be established by showing that the well-known *maximum independent set* problem, which is known to be $\mathcal{NP}$-hard [19], is a special case of CGP. For a given undirected simple graph $G = (V, E)$, an *independent set* of G is a subset U of V such that $(i, j) \notin E$ for every pair of nodes $i, j \in U$. The problem is to find a maximum cardinality independent set of G.

For each node $v \in V$, we define a n-dimensional positive vector $\mathbf{x}_v$ with components x_{vk} for all $k \in V$ such that $x_{vk} = 1$ for $k \neq v$ and $x_{vk} = 1/n^3$, where $n = |V|$. We also define, for each edge $e \in E$, a n-dimensional vector $\mathbf{y}_e$ with components y_{ek} for all $k \in V$ such that $y_{ek} = 1/n^6$ if $k = i$ or $k = j$ and $y_{ek} = 1$ otherwise, where $e = (i, j)$. Let $S_1 = \{\mathbf{x}_v | v \in V\}$ and let $S_2 = \{\mathbf{y}_e | e \in E\}$. We specify the four parameters of D as $d_{min} = 0, d_{max} = 1, T = 1$, and $L = n$. Then, the set D is the set of all n-dimensional binary vectors. Now, we set $a_1(\mathbf{x}_v)$ for each $v \in V$ to be equal to 1, and set each element of $a_2(\mathbf{y}_e)$ for each $e \in E$ to be n. Now, we define a special case of CGP with S_1 and S_2 along with $a_1(\mathbf{x}_v)$ for each $v \in V$, $a_2(\mathbf{y}_e)$ for each $e \in E$, and the four parameters of D. This special case of CGP is the problem of maximizing the function $z(\mathbf{d}) = \sum_{i\in V}(1/n^3)^{d_i} - n(\sum_{(i,j)\in E}(1/n^6)^{d_i}(1/n^6)^{d_j})$ over all $\mathbf{d} \in \mathbb{B}^n$.

Now, we show that a maximum independent set U such that $|U| = K$ exists for some positive integer $K < n$ if and only if $K \le \hat{z} < K + 1$ where $\hat{z} = \max\{z(\mathbf{d}) | \mathbf{d} \in \mathbb{B}^n\}$. Suppose that we have a maximum independent set U of G such that $|U| = K$. Then, for the vector $\mathbf{d}$ such that $d_i = 0$ if $i \in U$ and $d_j = 1$ otherwise, $z(\mathbf{d}) \ge K + 1/n^3(n - K) - n|E|/n^6 \ge K$

and $z(\mathbf{d}) \leq K + 1/n^3(n-K) < K+1$. Now, suppose that we have a binary vector $\mathbf{d}$ such that $K \leq \hat{z} < K+1$ for some positive integer such that $K < n$. This means that at least one of d_i and d_j should be equal to 1 if and only if $(i,j) \in E$, otherwise, $\hat{z} \leq 0$. Moreover, the number of zero elements of $\mathbf{d}$ is equal to K. If we define $U = \{i \in V | d_i = 0\}$, then U should be an independent set of cardinality K. Therefore, the result follows. QED.

Even though CGP is $\mathcal{NP}$-hard, we can find an optimal solution to CGP by using an enumerative algorithm such as a branch-and-bound algorithm. However, it may be too computationally demanding so that Algorithm 1, in which CGP has to be solved many times, may be impractical. Hence, we devise a heuristic algorithm to get an approximate solution to CGP, which implies Algorithm 1 does not guarantee optimal solutions to MPC.

CGP can be reformulated as the following nonlinear integer program by rewriting $d_j = p_j/T$ where p_j is an integer variable.

$$\begin{aligned} \text{maximize} \quad & \sum_{\mathbf{x} \in S_1} a_1(\mathbf{x}) \prod_{j=1}^{n} x_j^{p_j/T} - \sum_{\mathbf{x} \in S_2} a_2(\mathbf{x}) \prod_{j=1}^{n} x_j^{p_j/T} \\ \text{subject to} \quad & Td_{min} \leq p_j \leq Td_{max}, j = 1, \ldots, n \\ & \sum_{j=1}^{n} |p_j| \leq LT, \\ & \mathbf{p} \in \mathbb{Z}^n. \end{aligned} \tag{11}$$

To get an approximate solution to CGP, we first solve the continuous relaxation of the problem (11), the problem obtained by dropping the integrality restrictions imposed on the variables, then, recover a feasible integer solution to CGP by rounding.

The objective function of the problem (11), which we denote $z(\mathbf{p})$, is not convex nor concave. So, it may not be practically possible to get an optimal solution to the continuous relaxation of it. Therefore, we again devise a heuristic algorithm to solve the continuous relaxation approximately based on the Frank-Wolfe algorithm [5]. Let Ω be the set of feasible solutions to the continuous relaxation, that is,

$$\Omega = \{\mathbf{p} \in \mathbb{R}^n \,|\, \sum_{j=1}^{n} |p_j| \leq LT, Td_{min} \leq p_j \leq Td_{max}, j = 1, \ldots, n\}.$$

The basic idea is to generate a sequence of points in Ω starting from the initial point $\mathbf{p}^0$ until the improvement in the objective value is less than or equal to a given positive real number ε. For a given $\mathbf{p}^{k-1}$, the new point generated at kth iteration is given as $\mathbf{p}^k = \lambda \mathbf{p}^{k-1} + (1-\lambda)\hat{\mathbf{p}}$, where $\lambda \in [0,1]$ and $\hat{\mathbf{p}}$ is an optimal solution to an optimization problem which is formed by replacing the objective function of the continuous relaxation of the problem (11) with an affine function obtained by the first order Taylor approximation of $z(\mathbf{p})$ at $\mathbf{p}^{k-1}$, which we call CP_k. We choose the step length, λ, among the values $i/s, i = 0, \ldots, s$, so that $z(\mathbf{p}^k)$ is minimized over those possible choices. In our implementation, we set $\varepsilon = 10^{-4}$ and $s = 10$.

The problem, CP_k, that we solve at kth iteration can be formulated as follows.

$$\begin{aligned} (CP_k) \text{ maximize} \quad & c_0 + \sum_{j=1}^{n} c_j p_j \\ \text{subject to} \quad & (p_1, \ldots, p_n) \in \Omega, \end{aligned}$$

where the constant $c_0 = z(\mathbf{p}^{k-1}) - \nabla z(\mathbf{p}^{k-1}) \cdot \mathbf{p}^{k-1}$ and $\nabla z(\mathbf{p}^{k-1}) = (c_1, \ldots, c_n)$. It might not be quite obvious at first glance but CP_k is a simple knapsack problem [19] which can be solved in polynomial time. This can be shown by noting that an optimal solution to CP_k can be obtained by sorting c_j's in the decreasing order of absolute values and then assigning the value of p_j's in a greedy manner. So, we have the following result.

Proposition 1 *CP_k can be solved in $O(n \log n)$ time.*

Now, we present our algorithm for CGP in Algorithm 2. Note that the objective function of the problem (11) is bounded over Ω. Moreover, the objective function value of the point generated at each iteration of Algorithm 2 is strictly decreasing and ε is a positive number. Therefore, Algorithm 2 terminates in finite iterations.

3.3 An Implementation of SSCR for Regression

Regression functions are obtained by solving the problem (7) which can be reformulated into the following LP (MPR) as in the case of MPC.

$$\begin{aligned} \text{minimize} \quad & \sum_{\mathbf{d} \in D} (w_{\mathbf{d}}^{+} + w_{\mathbf{d}}^{-}) + C(\sum_{i=1}^{N} u_i + \sum_{i=1}^{M} v_i) \\ \text{subject to} \quad & y_i - \sum_{\mathbf{d} \in D} g_{\mathbf{d}}(\mathbf{x}_i)(w_{\mathbf{d}}^{+} - w_{\mathbf{d}}^{-}) - b \leq \epsilon + u_i, i = 1, \ldots, N \qquad (12) \\ & \sum_{\mathbf{d} \in D} g_{\mathbf{d}}(\mathbf{x}_i)(w_{\mathbf{d}}^{+} - w_{\mathbf{d}}^{-}) + b + y_i \leq \epsilon + v_i, i = 1, \ldots, N \qquad (13) \\ & \mathbf{w}^{+}, \mathbf{w}^{-} \in \mathbb{R}_{+}^{|D|}, b \in \mathbb{R}, \mathbf{u} \in \mathbb{R}_{+}^{N}, \mathbf{v} \in \mathbb{R}_{+}^{N}. \end{aligned}$$

As in the case of MPC, we devise an algorithm for MPR based on the column generation technique. Consider a subset of D, $I \subset D$, and the associated restriction of MPR to I, which we call the *restricted master problem* for regression (RMPR).

Let α_i for all $i = 1, \ldots, N$ and β_i for all $i = 1, \ldots, N$ be the nonnegative dual variables associated with constraints (12) and (13), respectively. For a given optimal solution, $(\hat{\alpha}, \hat{\beta})$, to the dual of RMPR, we may write the optimality condition for MPR as follows.

$$\max\{z(\mathbf{d}) : \mathbf{d} \in D\} \leq 1, \quad \min\{z(\mathbf{d}) : \mathbf{d} \in D\} \geq -1, \qquad (14)$$

where $z(\mathbf{d}) = \sum_{i=1}^{N} \hat{\alpha}_i g_{\mathbf{d}}(\mathbf{x}_i) - \sum_{i=1}^{N} \hat{\beta}_i g_{\mathbf{d}}(\mathbf{x}_i)$.

Algorithm 2 Algorithm for the column generation problem (CGP)

Given: S_1 and $a_1(\mathbf{x}), \mathbf{x} \in S_1$, S_2 and $a_2(\mathbf{x}), \mathbf{x} \in S_2$, d_{min}, d_{max}, L, T.
procedure GENERATE($S_1, \mathbf{a}_1, S_2, \mathbf{a}_2, d_{min}, d_{max}, L, T$)
  $\mathbf{p}^0 \leftarrow \mathbf{0}$, $z_0 \leftarrow z(\mathbf{p}^0)$, $k \leftarrow 1$;
  repeat
    $r \leftarrow 0$;
    Formulate CP_k with $\mathbf{p}^{k-1}$ and solve it;
    $\hat{\mathbf{p}} \leftarrow$ the optimal solution to CP_k;
    $\hat{z} \leftarrow z_{k-1}, \hat{\lambda} \leftarrow 0$;
    for $0 \leq i \leq s$ **do**
      $\lambda \leftarrow i/s$, $\hat{\mathbf{q}} \leftarrow \lambda \mathbf{p}^{k-1} + (1 - \lambda)\hat{\mathbf{p}}$;
      if $z(\hat{\mathbf{q}}) > \hat{z}$ **then**
        $\hat{z} \leftarrow z(\hat{\mathbf{q}})$, $\hat{\lambda} \leftarrow \lambda$;
      end if
    end for
    $\mathbf{p}^k \leftarrow \hat{\lambda}\mathbf{p}^{k-1} + (1 - \hat{\lambda})\hat{\mathbf{p}}$, $z_k \leftarrow z(\mathbf{p}^k)$;
    if $z_k - z_{k-1} > \varepsilon$ **then**
      $r \leftarrow 1, k \leftarrow k + 1$;
    end if
  until $r = 0$
  $\hat{p}_j \leftarrow \lfloor p_j^k \rfloor, j = 1, \ldots, n$;
  $\hat{z} \leftarrow z(\hat{\mathbf{p}})$, $\hat{\mathbf{d}} \leftarrow \hat{\mathbf{p}}/T$;
  Return $\hat{\mathbf{d}}, \hat{z}$;
end procedure

From the condition (14), we can check if the current optimal solution to RMPR is optimal to MPR by solving two optimization problems, which we call SPR1 and SPR2, respectively. SPR1 is the problem of maximizing $z(\mathbf{d})$ over all $\mathbf{d} \in D$ and SPR2 is the exactly the same problem as SPR1 except that the objective function is to be minimized. Let $\hat{z}^U$ and $\hat{z}^L$ be the optimal objective value of SPR1 and SPR2, respectively. Then, the optimality condition (14) is satisfied if and only if $\hat{z}^U \leq 1$ and $\hat{z}^L \geq -1$.

Observe that SPR1 and SPR2 are special cases of CGP in which $S_1 = S_2$. This implies that we can solve these problems by using Algorithm 2. One may expect that these problems are easier to solve than CGP because these are more specialized problems than CGP. However, the following theorem shows that CGP is $\mathcal{NP}$-hard even if $S_1 = S_2$, which implies SPR1 and SPR2 are $\mathcal{NP}$-hard.

Theorem 2 *CGP is $\mathcal{NP}$-hard even if $S_1 = S_2$.*

Proof The result can also be established by showing that the *maximum independent set* problem is a special case of CGP with $S_1 = S_2$. For any given instance of the maximum independent set problem, we define the vectors $\mathbf{x}_v, v \in V$ and $\mathbf{y}_e, e \in E$ as in the proof of theorem (1). Let $S_1 = S_2 = \{\mathbf{x}_v, v \in V\} \cup \{\mathbf{y}_e, e \in E\}$. In addition to $a_1(\mathbf{x}_v)$ for each $v \in V$ and $a_2(\mathbf{y}_e)$ for each $e \in E$ defined in the proof of theorem (1), we define $a_1(\mathbf{y}_e) = 0$ for each $e \in E$ and $a_2(\mathbf{x}_v) = 0$ for each $v \in V$. Then, we can define a special case of CGP with $S_1, S_2, \mathbf{a}_1, \mathbf{a}_2$ along with the four parameters of D specified as those in the proof of theorem (1). Observe that this special case of CGP with $S_1 = S_2$ is the same problem as that we discussed in the proof of theorem (1). Therfore, the result follows. QED.

Now, we present our implementation of the algorithm, SPARSESIGNOMIALREGRESSION, to solve the problem (7) in Algorithm 3, in which the column generation algorithm for CGP (GENERATE) given in the previous subsection is used to solve SPR1 and SPR2 as a subroutine.

Algorithm 3 Algorithm for regression based on the column generation

Given: $\{(\mathbf{x}_i, y_i), i = 1, \ldots, N\}, d_{min}, d_{max}, L, T, C$, and ϵ.
procedure SPARSESIGNOMIALREGRESSION
  $I \leftarrow \emptyset, J \leftarrow \emptyset, S \leftarrow \{\mathbf{x}_i, i = 1, \ldots, N\}$;
  repeat
    $r \leftarrow 0$;
    Formulate RMPR with I and solve it using the simplex algorithm;
    $(\hat{\mathbf{w}}^+, \hat{\mathbf{w}}^-, \hat{b}, \hat{\mathbf{u}}, \hat{\mathbf{v}}) \leftarrow$ the optimal solution to RMPR;
    $(\hat{\alpha}, \hat{\beta}) \leftarrow$ the optimal solution to the dual of RMPR;
    Solve SPR1 using GENERATE$(S, \hat{\alpha}, S, \hat{\beta}, d_{min}, d_{max}, L, T)$;
    $\hat{\mathbf{d}}^U \leftarrow$ the obtained solution to SPR1;
    $\hat{z}^U \leftarrow$ the corresponding objective value to $\hat{\mathbf{d}}^U$;
    if $\hat{z}^U \leq 1$ **then**
      Solve SPR2 using GENERATE$(S, \hat{\beta}, S, \hat{\alpha}, d_{min}, d_{max}, L, T)$;
      $\hat{\mathbf{d}}^L \leftarrow$ the obtained solution to SPR2;
      $\hat{z}^L \leftarrow$ the corresponding objective value to $\hat{\mathbf{d}}^L$;
      if $\hat{z}^L < -1$ **then**
        $I \leftarrow I \cup \{\hat{\mathbf{d}}^L\}$;
        $r \leftarrow 1$;
      end if
    else
      $I \leftarrow I \cup \{\hat{\mathbf{d}}^U\}$;
      $r \leftarrow 1$;
    end if
  until $r = 0$
  $\hat{\mathbf{w}} \leftarrow (\hat{\mathbf{w}}^+ - \hat{\mathbf{w}}^-)$;
  $J \leftarrow \{\mathbf{d} \in I | \hat{w}_{\mathbf{d}} \neq 0\}$;
  Return the regression function $f(\mathbf{x}) = \sum_{\mathbf{d} \in J} \hat{w}_{\mathbf{d}} g_{\mathbf{d}}(\mathbf{x}) + \hat{b}$;
end procedure

4 Performance of SSCR

In this section, we report on the performance of SSCR on several widely circulated real data sets from the UCI Machine Learning Repository [18]. Table 1 shows the characteristics of those data sets including the name of each data set, the number of observations and variables of each data set, and the associated task with each data set.

Table 1: The characteristics of data sets

Data sets	#Observations	#Variables	Associated task
Breast Cancer	683	9	Classification
Bupa	345	6	Classification
Heart Disease	297	13	Classification
Housing	506	13	Classification/Regression
Abalone	4177	7	Regression
Machine	209	6	Regression

The `Heart Disease` data set originally consists of 303 observations but 6 of them have some missing values. We discarded those observations and used the remaining 297 observations. The `Housing` data set was used for both classification and regression. For the classification, the data set was split into two classes depending whether the housing value is above or below the median.

In section 4.1, we present the performance of SSCR on classification data sets and compare it to the performances of some of current popular methods including SVM [22], logistic regression [9], and CART [1, 11]. Section 4.2 presents the performances of SSCR for regression and compared it to those of linear least-squares regression(LSR) [17], MARS [6] and SVR [20].

SSCR was implemented by using Xpress Mosel language and associated optimization library functions [24]. SVM and SVR were tested by using the LIBSVM package [3], and the R software package [16] was used for testing MARS. We used the MATLAB toolbox [15] for testing CART, logistic regression, and LSR. The experiments were executed on a Windows desktop (2.3GHz CPU, 3GB RAM).

In our preliminary computational experiments, we observed that it takes much computation time (it took several hours for polynomial kernels in many cases) for SVM and SVR to run on the original data. So, we used normalized data for SVM and SVR. For each original observation $\mathbf{x}_i = (x_{i1}, \ldots, x_{in})$, each component $\bar{x}_{ij}$ of the corresponding normalized vector $\bar{\mathbf{x}}_i$ was obtained as follows.

$$\bar{x}_{ij} = \frac{x_{ij} - \min_{i \in O} x_{ij}}{\max_{i \in O} x_{ij} - \min_{i \in O} x_{ij}},$$

where O is the set of observations of a data set. For SSCR, we translated the given data by adding 1 to each component x_{ij} of each observation to make sure that all the data are in the positive orthant.

4.1 Classification Results

We tested SSCR for classification on four data sets, `Breast Cancer`, `Bupa`, `Heart Disease`, and `Housing`, and we compared its performance in terms of classification accuracy measured by the portion of correctly classified observations to those of SVM, logistic regression, and CART.

For SSCR, we specified the set D defined as (6) with its parameters $d_{min} = -1, d_{max} = 1, T = 100$ and $L = 1$ for all the data sets. The model parameter C of (5) was varied such that $C = 10^{-3}$, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}. For each value of C, we performed ten independent runs of twofold cross validations. We summarize the average training and test accuracies in table 2 for the best value of C with respect to the average test accuracy and the corresponding average computation time in seconds in table 3.

For SVM, linear, polynomial(with degree 2 and 3) and RBF kernels were all tested with varying model parameter P (similar parameter to C in SSCR) such that $P = 10^{-3}$, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}. For RBF kernels, the kernel parameter σ was varied such that $\sigma =$ 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}. For each combination of a kernel, P, and σ among 77 combinations in all, we also performed ten independent runs of twofold cross validations. Then, the average training and test accuracies for the best combination in terms of the average test accuracy and the corresponding average computation time are summarized in table 2 and table 3, respectively.

For CART, trees are pruned based on an optimal pruning scheme that first prunes branches giving less improvement in error cost. By varying the pruning level, we performed ten independent runs of twofold cross validations and the average training and test accuracies for the best pruning level in terms of the average test accuracy and the corresponding average computation time are summarized in table 2 and table 3, respectively.

From table 2, we can see that SSCR and SVM performed consistently better than CART and logistic regression. For all data sets except the `Breast Cancer` data set, SSCR showed consistently better results than others with respect to the average test accuracies. The results show that the proposed SSCR is a comparable classification method to existing popular methods.

Table 3 summarizes the average CPU seconds taken by each method for the best parameter combination for each classification data set. The results shows that SSCR requires more computation time than the other three methods. However, we expect that more sophisticated implementation of SSCR with further enhancements would make it possible to speed up SSCR, which remains to be studied further. The values of each cell of table 3 is for one specific parameter combination of a specific method. The overall average computation time taken by SVM and SSCR for all parameter combinations for each data set is given in table 4.

Table 2: Classification results - average training and test accuracies for the best parameter combination for each data set (standard deviations in parentheses)

Data sets	CART		Logistic regression		SVM		SSCR	
	Train	Test	Train	Test	Train	Test	Train	Test
Breast Cancer	0.975 (0.005)	0.946 (0.007)	0.974 (0.004)	0.963 (0.005)	0.976 (0.002)	0.971 (0.002)	0.974 (0.001)	0.967 (0.004)
Bupa	0.835 (0.014)	0.635 (0.025)	0.709 (0.010)	0.677 (0.023)	0.788 (0.009)	0.720 (0.016)	0.803 (0.008)	0.723 (0.016)
Heart Disease	0.876 (0.016)	0.771 (0.025)	0.871 (0.008)	0.817 (0.016)	0.849 (0.009)	0.821 (0.007)	0.875 (0.009)	0.822 (0.014)
Housing	0.907 (0.016)	0.834 (0.017)	0.882 (0.005)	0.860 (0.008)	0.920 (0.004)	0.867 (0.020)	0.970 (0.005)	0.869 (0.011)

Table 3: Computation time in seconds for classification - average CPU seconds taken by each method for the best parameter combination for each data set (standard deviations in parentheses)

Data sets	CART	Logistic regression	SVM	SSCR
Breast Cancer	0.167 (0.029)	0.107 (0.039)	0.004 (0.006)	0.871 (0.239)
Bupa	0.166 (0.028)	0.034 (0.026)	0.020 (0.005)	1.910 (0.438)
Heart Disease	0.049 (0.058)	0.039 (0.023)	0.018 (0.007)	2.314 (0.333)
Housing	0.185 (0.027)	0.070 (0.030)	0.008 (0.005)	8.322 (1.516)

Table 4: Computation time in seconds for classification - average CPU seconds taken by SVM and SSCR for all parameter combinations for each data set

Data sets	SVM	SSCR
Breast Cancer	0.064	5.961
Bupa	0.024	3.825
Heart Disease	0.091	7.062
Housing	0.029	8.468

Table 5 show how the number of generated columns (`#GenTerms`), that of selected columns (the number of terms in the resulting signomial function) (`#SelTerms`), and training and test accuracies change with respect to the value of model parameter C in SSCR. The values of each cell of the table show the corresponding average and standard deviation of ten runs of

twofold cross validations. As the value of C increases, SSCR generates more columns (terms) and produces classifiers with more terms in order to minimize the amount of violation caused by the resulting function, so that the average training accuracies increase. More importantly, we can see that SSCR generated very small number of terms compared to the number of possible terms and produced very sparse classifiers with high test accuracies. Thus those obtained functions would be useful for interpretation regarding which original input variables and/or nonlinear terms are more important than others in classification.

Table 5: The behavior of SSCR for classification with respect to the value of C (standard deviations in parentheses)

C	Breast Cancer				Bupa			
	#SelTerms	#GenTerms	Train	Test	#SelTerms	#GenTerms	Train	Test
0.001	0.0 (0.0)	18.0 (0.0)	0.650 (0.000)	0.650 (0.000)	1.6 (0.5)	14.2 (0.7)	0.580 (0.000)	0.580 (0.000)
0.01	6.4 (0.2)	18.8 (0.8)	0.962 (0.003)	0.959 (0.003)	5.4 (0.5)	23.3 (2.5)	0.721 (0.017)	0.688 (0.025)
0.1	7.9 (0.8)	22.2 (1.1)	0.974 (0.001)	0.967 (0.004)	9.6 (0.9)	23.4 (2.9)	0.760 (0.011)	0.722 (0.014)
1	14.1 (1.4)	36.4 (4.1)	0.982 (0.003)	0.962 (0.004)	19.1 (2.0)	40.7 (6.8)	0.803 (0.008)	0.723 (0.016)
10	25.6 (1.6)	79.5 (13.2)	1.000 (0.000)	0.940 (0.008)	36.5 (2.5)	71.3 (9.4)	0.861 (0.007)	0.681 (0.029)
100	25.5 (1.3)	75.7 (8.8)	1.000 (0.000)	0.943 (0.007)	56.1 (3.8)	107.0 (16.2)	0.929 (0.012)	0.635 (0.016)
1000	25.4 (1.3)	75.1 (4.6)	1.000 (0.000)	0.945 (0.006)	68.5 (3.9)	133.7 (7.2)	0.995 (0.004)	0.602 (0.022)
C	Heart Disease				Housing			
	#SelTerms	#GenTerms	Train	Test	#SelTerms	#GenTerms	Train	Test
0.001	1.0 (0.4)	26.5 (0.4)	0.545 (0.021)	0.543 (0.014)	4.1 (0.4)	34.7 (1.6)	0.775 (0.008)	0.769 (0.007)
0.01	7.5 (0.5)	40.5 (1.9)	0.846 (0.014)	0.801 (0.026)	9.2 (0.8)	45.5 (2.3)	0.854 (0.007)	0.835 (0.014)
0.1	13.2 (0.7)	55.7 (3.1)	0.875 (0.009)	0.822 (0.014)	20.1 (1.2)	60.7 (5.2)	0.918 (0.007)	0.857 (0.014)
1	33.1 (2.0)	89.9 (8.0)	0.952 (0.009)	0.781 (0.016)	34.4 (2.1)	75.9 (7.0)	0.970 (0.005)	0.869 (0.011)
10	45.1 (2.0)	131.3 (10.7)	1.000 (0.000)	0.744 (0.024)	43.4 (2.0)	96.2 (8.2)	0.997 (0.004)	0.849 (0.011)
100	45.8 (2.1)	146.4 (15.1)	1.000 (0.000)	0.744 (0.021)	44.5 (2.0)	98.4 (14.4)	1.000 (0.000)	0.847 (0.012)
1000	46.5 (2.6)	141.7 (11.9)	1.000 (0.000)	0.736 (0.017)	44.9 (2.9)	104.8 (9.1)	1.000 (0.000)	0.845 (0.011)

As mentioned earlier, SVM required much more time on the original data in our preliminary computational test but showed good performance on the normalized data as summarized in table 2. This means that the performance of SVM can be very sensitive to the numerical values and numeric ranges of the given data at least with respect to the computation time. To assess the effects of the normalization on SSCR for classification, the training and test accuracies of SSCR with the corresponding computation time on the normalized data (the same data used for SVM but translated by adding 1 to each value) is compared to those on the original data translated by adding 1 to each value. Table 6 shows the average training and test accuracies with the corresponding average computation time in seconds of ten independent runs of twofold cross validations for the best value of C with respect to the average test accuracy. From the results, we can see that SSCR is less sensitive than SVM to the numerical values and numeric ranges of the data.

Table 6: The effects of data normalization on SSCR - average training and test accuracies with the corresponding average computation time in seconds for the best value of C (standard deviations in parentheses)

Data sets	Normalized data			Original data		
	Train	Test	Time	Train	Test	Time
Breast Cancer	0.975 (0.002)	0.968 (0.003)	0.735 (0.277)	0.974 (0.001)	0.967 (0.004)	0.871 (0.239)
Bupa	0.797 (0.009)	0.723 (0.016)	2.905 (0.510)	0.803 (0.008)	0.723 (0.016)	1.910 (0.438)
Heart Disease	0.864 (0.004)	0.826 (0.016)	1.496 (0.256)	0.875 (0.009)	0.822 (0.014)	2.314 (0.333)
Housing	0.971 (0.006)	0.870 (0.017)	12.906 (2.287)	0.970 (0.005)	0.869 (0.011)	8.322 (1.516)

4.2 Regression Results

We tested SSCR for regression on three data sets, `Housing`, `Abalone`, and `Machine`, and we compared its performance in terms of prediction errors measured by the mean squared error(MSE) and the mean absolute error(MAE) to those of LSR, MARS and SVR.

For SSCR, we specified the set D defined as (6) with its parameters $d_{min} = -1, d_{max} = 1, T = 100$ and $L = 1$ for all the data sets as in the case of classification. Also, the model parameter C of (7) was varied such that $C = 10^{-3}$, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}. The value of ϵ of (7) was varied such that $\epsilon = 10^{-3}$, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}. For each combination of C and ϵ, we performed ten independent runs of twofold cross validations. In table 7, we summarize the average MSE for the best combination of C and ϵ with respect to the average test MSE and the corresponding average computation time in seconds in table 9. Table 8 shows the average MAE for the best combination of C and ϵ with respect to the average test MAE.

For SVR, linear, polynomial(with degree 2 and 3) and RBF kernels were all tested with varying model parameter P (similar parameter to C in SSCR) such that $P = 10^{-3}$, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}. For RBF kernels, the kernel parameter σ was varied such that $\sigma = 10^{-3}$, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}. In addition, we varied ϵ for the ϵ-insensitive loss function in SVR such that $\epsilon = 10^{-3}$, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2} and 10^{3}. For each combination of a kernel, P, σ, and ϵ, we also performed ten independent runs of twofold cross validations. Then, the average MSE for the best combination and the corresponding average computation time are summarized in table 7 and table 9, respectively. The average MAE for the best combination of parameters for each data set is summarized in table 8.

For MARS, we also performed ten independent runs of twofold cross validations by varying the maximum number of basis functions, and the best average MSE for each data set and the corresponding computation time are summarized in table 7 and table 9, respectively. The best MAE for each data set is shown in table 8

From table 7 and 8, we can see that regression functions obtained by SSCR show smaller prediction errors in terms of MSE and MAE than the other methods except for the case of `Housing` data set. This is an encouraging result since the parameter selection decision for SSCR may be less complex than those for SVR.

Table 7: Regression results - average training and test error in terms of MSE for the best parameter combination for each data set (standard deviations in parentheses)

Data sets	LSR		MARS		SVR		SSCR	
	Train	Test	Train	Test	Train	Test	Train	Test
Abalone	5.014 (0.007)	5.207 (0.034)	4.435 (0.014)	5.340 (2.270)	4.369 (0.011)	4.641 (0.026)	4.381 (0.017)	4.570 (0.045)
House	23.166 (0.810)	27.287 (2.488)	11.326 (0.550)	17.577 (2.172)	3.487 (0.348)	13.973 (1.647)	6.468 (0.494)	14.087 (2.283)
Machine	3762.855 (334.973)	6579.079 (1402.490)	2205.330 (329.225)	5722.218 (2686.105)	1818.508 (375.475)	3514.737 (1068.819)	1245.005 (67.955)	3084.225 (913.164)

Table 8: Regression results - average training and test error in terms of MAE for the best parameter combination for each data set (standard deviations in parentheses)

Data sets	LSR		MARS		SVR		SSCR	
	Train	Test	Train	Test	Train	Test	Train	Test
Abalone	1.632 (0.001)	1.642 (0.003)	1.523 (0.003)	1.553 (0.016)	1.444 (0.002)	1.486 (0.003)	1.449 (0.002)	1.481 (0.003)
House	3.248 (0.019)	3.483 (0.091)	2.388 (0.045)	2.844 (0.079)	1.419 (0.030)	2.339 (0.101)	1.382 (0.045)	2.477 (0.098)
Machine	38.713 (1.681)	44.344 (2.249)	29.497 (1.901)	37.354 (4.118)	22.727 (0.786)	29.399 (1.983)	18.757 (0.238)	29.230 (2.687)

Table 9: Computation time in seconds for regression - average CPU seconds taken by each method for the best parameter combination in terms of MSE for each data set (standard deviations in parentheses)

Data sets	LSR	MARS	SVR	SSCR
Abalone	0.000 (0.000)	0.999 (0.031)	2.215 (0.061)	104.919 (9.056)
House	0.000 (0.000)	0.051 (0.022)	0.163 (0.020)	52.433 (6.655)
Machine	0.002 (0.005)	0.040 (0.016)	0.003 (0.004)	1.707 (0.265)

Table 10: Computation time in seconds for regression - average CPU seconds taken by SVR and SSCR for all parameter combinations for each data set

Data sets	SVR	SSCR
Abalone	2.454	40.448
Housing	0.079	139.841
Machine	0.008	9.558

Table 9 summarizes the average CPU seconds taken by each regression method for the best parameter combination for each data set. The overall average computation time taken by SVR and SSCR for all parameter combinations for each data set is given in table 10. As in the case of classification, the results shows that SSCR requires more computation time than the other methods.

Computation time taken by SVR usually depends on the number of observations. The number of observations of `Abalone` data set is about 8 times more that that of `Housing` data set. Table 10 shows that SVR took about 30 times more computation time in average on

`Abalone` data set than that on `Housing` data set. However, SSCR took about 3 times more time on `Housing` data set (13 variables for each observation) than that on `Abalone` data set (7 variables for each observation). From these results, we can expect that computation time taken by SSCR depends more on the number of variables of each observations than on the number of observaitons.

Tables 11 and 12 show how the number of generated columns (`#GenTerms`), that of selected columns (the number of terms in the resulting signomial function) (`#SelTerms`), and training and test errors in terms of MSE (MAE for table 12) change with respect to the value of model parameter C in SSCR. For each data set, the value of ϵ was set to the value for which the average test error was the smallest. The values of each cell of the table show the corresponding average and standard deviation of ten runs of twofold cross validations.

As the value of C increases, SSCR for regression shows similar behavior to that in the case of classification given in table 5. From the results, we can also see that SSCR can provide sparse regression functions with small prediction errors.

Table 11: The behavior of SSCR for regression with respect to the value of C (traing and test errors in terms of MSE, standard deviations in parentheses)

C	Abalone (ϵ = 1)				Housing ($\epsilon = 0.1$)				Machine ($\epsilon = 10$)			
	#SelTerms	#GenTerms	Train	Test	#SelTerms	#GenTerms	Train	Test	#SelTerms	#GenTerms	Train	Test
0.001	0.0	0.0	10.430	10.451	4.8	11.2	41.415	43.116	8.7	13.3	2242.124	3881.155
	(0.0)	(0.0)	(0.105)	(0.170)	(0.4)	(1.9)	(0.703)	(1.047)	(1.1)	(2.4)	(472.603)	(983.191)
0.01	1.0	1.0	7.812	7.823	13.1	33.2	19.362	22.591	18.6	28.5	1245.005	3084.225
	(0.0)	(0.0)	(0.008)	(0.009)	(1.1)	(3.2)	(0.678)	(1.582)	(1.6)	(4.1)	(67.955)	(913.164)
0.1	3.6	12.0	5.458	5.479	32.1	68.7	10.433	14.667	35.2	51.8	547.698	4667.157
	(0.4)	(1.5)	(0.010)	(0.022)	(1.7)	(8.4)	(0.388)	(1.612)	(2.5)	(8.1)	(107.291)	(1675.852)
1	8.0	24.5	4.729	4.776	71.5	134.9	6.468	14.087	57.2	80.5	236.097	7616.161
	(0.6)	(2.7)	(0.018)	(0.025)	(2.1)	(14.2)	(0.494)	(2.283)	(2.6)	(11.7)	(35.291)	(2861.743)
10	12.9	45.9	4.550	4.637	144.7	270.6	2.197	36.417	74.5	109.5	169.102	41737.326
	(0.6)	(3.8)	(0.016)	(0.031)	(5.3)	(31.9)	(0.333)	(7.980)	(2.8)	(12.8)	(34.455)	(31391.184)
100	23.0	66.5	4.381	4.570	208.0	338.7	0.547	185.313	79.0	122.0	155.923	100500.174
	(0.8)	(5.4)	(0.017)	(0.045)	(4.2)	(32.0)	(0.132)	(73.881)	(2.1)	(9.7)	(30.328)	(65153.226)
1000	37.8	97.8	4.331	4.868	241.1	369.0	0.055	1131.669	79.0	133.2	115.114	102702.666
	(1.4)	(10.4)	(0.018)	(0.363)	(2.4)	(42.0)	(0.022)	(460.105)	(2.3)	(14.8)	(31.170)	(63190.713)

Table 12: The behavior of SSCR for regression with respect to the value of C (traing and test errors in terms of MAE, standard deviations in parentheses)

C	Abalone (ϵ = 0.1)				Housing ($\epsilon = 0.1$)				Machine ($\epsilon = 10$)			
	#SelTerms	#GenTerms	Train	Test	#SelTerms	#GenTerms	Train	Test	#SelTerms	#GenTerms	Train	Test
0.001	0.0	0.0	2.358	2.364	4.8	11.2	4.033	4.198	8.7	13.3	24.171	30.909
	(0.0)	(0.0)	(0.002)	(0.002)	(0.4)	(1.9)	(0.034)	(0.042)	(1.1)	(2.4)	(1.489)	(3.087)
0.01	1.0	1.0	1.907	1.909	13.1	33.2	2.882	3.202	18.6	28.5	18.757	29.230
	(0.0)	(0.0)	(0.000)	(0.002)	(1.1)	(3.2)	(0.028)	(0.086)	(1.6)	(4.1)	(0.238)	(2.687)
0.1	3.9	12.9	1.597	1.601	32.1	68.7	2.040	2.637	35.2	51.8	14.285	34.247
	(0.2)	(1.1)	(0.002)	(0.001)	(1.7)	(8.4)	(0.034)	(0.065)	(2.5)	(8.1)	(0.838)	(3.226)
1	8.5	23.6	1.511	1.520	71.5	134.9	1.382	2.477	57.2	80.5	10.867	43.420
	(1.0)	(4.0)	(0.002)	(0.004)	(2.1)	(14.2)	(0.045)	(0.098)	(2.6)	(11.7)	(0.342)	(6.143)
10	13.8	47.3	1.483	1.499	144.7	270.6	0.683	3.359	74.5	109.5	9.893	78.178
	(0.8)	(4.4)	(0.003)	(0.005)	(5.3)	(31.9)	(0.047)	(0.194)	(2.8)	(12.8)	(0.365)	(14.127)
100	24.3	70.1	1.449	1.481	208.0	338.7	0.273	6.314	79.0	122.0	9.747	122.759
	(0.9)	(8.4)	(0.002)	(0.003)	(4.2)	(32.0)	(0.032)	(0.697)	(2.1)	(9.7)	(0.315)	(25.831)
1000	41.2	102.3	1.433	1.497	241.1	369.0	0.119	12.957	79.0	133.2	9.729	122.990
	(1.4)	(11.1)	(0.002)	(0.010)	(2.4)	(42.0)	(0.008)	(1.650)	(2.3)	(14.8)	(0.300)	(25.601)

As in the case of classification, to assess the effects of the normalization on SSCR for regression, the training and test errors of SSCR with the corresponding computation time on the normalized data (the same data used for SVR but translated by adding 1 to each value) were compared to those on the original data translated by adding 1 to each value. Table 13 shows the average training and test errors in terms of MSE with the corresponding average computation time in seconds of ten independent runs of twofold cross validations for the best combination of C and ϵ with respect to the average test error in terms of MSE. The average training and test errors with the corresponding average computation time in seconds for the best combination of C and ϵ with respect to the average test error in terms of MAE are shown in table 14. From the results, we can also see that SSCR is less sensitive to the numerical values and numeric ranges of the data.

Table 13: The effects of data normalization on SSCR - average training and test accuracies with the corresponding average computation time in seconds for the best combination of C and ϵ in terms of MSE(standard deviations in parentheses)

Data sets	Normalized data			Original data		
	Train	Test	Time	Train	Test	Time
Abalone	4.411 (0.017)	4.581 (0.032)	122.748 (13.614)	4.381 (0.017)	4.570 (0.045)	104.919 (9.056)
Housing	11.222 (0.873)	15.480 (1.351)	29.465 (4.033)	6.468 (0.494)	14.087 (2.283)	52.433 (6.655)
Machine	1963.610 (647.410)	3990.509 (1372.045)	1.605 (0.187)	1245.005 (67.955)	3084.225 (913.164)	1.707 (0.265)

Table 14: The effects of data normalization on SSCR - average training and test accuracies with the corresponding average computation time in seconds for the best combination of C and ϵ in terms of MAE(standard deviations in parentheses)

Data sets	Normalized data			Original data		
	Train	Test	Time	Train	Test	Time
Abalone	1.454 (0.002)	1.484 (0.005)	145.620 (19.146)	1.449 (0.002)	1.481 (0.003)	119.104 (14.376)
Housing	1.979 (0.042)	2.503 (0.078)	30.356 (3.508)	1.382 (0.045)	2.477 (0.098)	52.433 (6.655)
Machine	22.684 (1.457)	30.739 (2.599)	1.605 (0.187)	18.757 (0.238)	29.230 (2.687)	1.707 (0.265)

5 Concluding Remarks

In this paper, we discussed a new method to solve classification and regression problems, which is called SSCR. SSCR gives a sparse signomial function by solving a linear program based on the column generation technique. It gives an explicit function description in the original input space while exploring the nonlinearity of the given data without using nonlinear kernels, which can be used for prediction as well as interpretation. While more computational test is clearly needed, we believe that the preliminary computational evidence on widely used real data sets shows SSCR is promising and worthwhile to study further because SSCR has consistently showed competitive or better performance compared to current popular methods on these data sets. Especially, the training and test accuracies of SSCR are comparable to those of SVMs. This is an encouraging result since SSCR, unlike SVMs, does not require a kernel type to be selected, so that the parameter selection decision for SSCR may be less complex than those for SVMs.

From the implementation point of view, there are some issues needed to be studied further. First, an efficient implementation of SSCR is needed to reduce the computation time and to handle large number of observations (at least, hundreds of thousands of observations). For this purpose, a column generation approach combined with the constraint generation technique [19] could be considered as a viable option. Second, a more efficient and effective algorithm to solve the column generation problem (CGP) would be needed to enhance the performance of SSCR.

Since SSCR seeks an explicit function description in the original input space, the variable selection decision can be naturally incorporated into an integrated optimization model that would be a mixed-integer program (MIP), which we believe worthwhile to study. Also, we believe that SSCR can be applied to the problem of data description [21], which also remains to be studied further.

References

[1] L. Brieman, J. Friedman, R. Olshen and C. Stone, Classification and Regression Trees, Wadsworth International, Belmont, 1984.

[2] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery **2** (1998) 121–167.

[3] C. C. Chang and C. J. Lin, LIBSVM: a Library for Support Vector Machines, http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2001.

[4] V. Chvátal, Linear Programming, W. H. Freeman and Company, New York, 1983.

[5] M. Frank and P. Wolfe, An algorithm for quadratic programming, Naval Research Logistics Quarterly **3** (1956) 95–110.

[6] J. Friedman, Multivariate adaptive regression splines, Annals of Statistics **19** (1991) 1–67.

[7] S. R. Gunn, Support vector machines for classification and regression, Technical Report, School of Electronics and Computer Science, University of Southampton, 1998.

[8] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning. Springer, New York, 2001.

[9] D. Hosmer and S. Lemeshow, Applied logistic regression (2nd ed.), John Wiley & Sons, New York, 2000.

[10] K. Huang, D. Zheng, I. King, and M. R. Lyu, Arbitrary norm support vector machines, Neural Computation **21** (2009) 560–582.

[11] H. Kim and W. Y. Loh, Classification tree with unbiased multiway splits, Journal of American Statistical Association **96** (2001) 598–604.

[12] O. L. Mangasarian, Arbitrary-norm separating plane, Operations Research Letters **24** (1999) 15–23.

[13] O. L. Mangasarian, Exact 1-norm support vector machines via unconstrained convex differentiable minimization, Journal of Machine Learning Research **7** (2006) 1517–1530.

[14] O. L. Mangasarian and M. E. Thomson, Chunking for massive nonlinear kernel classification, Optimization Methods and Software **23** (2008) 265–274.

[15] Matlab Statistics Toolbox, http://www.mathworks.com, 2008.

[16] Mixture and Flexible Discriminant Analysis Package, http://cran.r-project.org/web/packages/mda, 2009.

[17] D. C. Montgomery, E. A. Peck and G. G. Vining, Introduction to Linear Regression Analysis (4th ed.), John Wiley & Sons, New York, 2006.

[18] P. M. Murphy and D. W. Aha, UCI machine learning repository, www.ics.uci.edu/∼mlearn/MLRepository.html, 1992.

[19] G. L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, New York, 1988.

[20] A. J. Smola and B. Schölkopf, A tutorial on support vector regression, Statistics and Computing **14** (2004) 199–222.

[21] D. M. J. Tax and R. P. W. Duin, Support vector data description, Machine Learning **54** (2004) 45–66.

[22] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[23] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, Use of zero-norm with linear models and kernel methods, Journal of Machine Learning Research **3** (2003) 1439–1461.

[24] Xpress-MP 2008B, http://www.dashoptimization.com, 2008.