UNIVERSIDAD CARLOS III DE MADRID working papers

UC3M Working Papers
Statistics and Econometrics
16-03
ISSN 2387-0303
February 2016

Departamento de Estadística
Universidad Carlos III de Madrid
Calle Madrid, 126
28903 Getafe (Spain)
Fax (34) 91 624-98-48

A Partial Parametric Path Algorithm for Multi-class Classification

Ling Liu[a], Belén Martín-Barragán[b], Francisco Javier Prieto[a]

Abstract

The objective functions of Support Vector Machine methods (SVMs) often include parameters to weigh the relative importance of margins and training accuracies. The values of these parameters have a direct effect both on the optimal accuracies and the misclassification costs. Usually, a grid search is used to find appropriate values for them. This method requires the repeated solution of quadratic programs for different parameter values, and it may imply a large computational cost, especially in a setting of multiclass SVMs and large training datasets. For multi-class classification problems, in the presence of different misclassification costs, identifying a desirable set of values for these parameters becomes even more relevant. In this paper, we propose a partial parametric path algorithm, based on the property that the path of optimal solutions of the SVMs with respect to the preceding parameters is piecewise linear. This partial parametric path algorithm requires the solution of just one quadratic programming problem, and a number of linear systems of equations. Thus it can significantly reduce the computational requirements of the algorithm. To systematically explore the different weights to assign to the misclassification costs, we combine the partial parametric path algorithm with a variable neighborhood search method. Our numerical experiments show the efficiency and reliability of the proposed partial parametric path algorithm.

Keywords: Multi-class SVM, Piecewise Linearity, Partial Parametric Path Algorithm, Variable Neighborhood Search

[a] Department of Statistics, Universidad Carlos III de Madrid.
[b] Business School, University of Edinburgh

A Partial Parametric Path Algorithm for Multi-class Classification

Ling Liu[*] Belén Martín-Barragán[†] Francisco J. Prieto[‡]

February 24, 2016

Abstract

The objective functions of Support Vector Machine methods (**SVMs**) often include parameters to weigh the relative importance of margins and training accuracies. The values of these parameters have a direct effect both on the optimal accuracies and the misclassification costs. Usually, a grid search is used to find appropriate values for them. This method requires the repeated solution of quadratic programs for different parameter values, and it may imply a large computational cost, especially in a setting of multi-class SVMs and large training datasets. For multi-class classification problems, in the presence of different misclassification costs, identifying a desirable set of values for these parameters becomes even more relevant. In this paper, we propose a partial parametric path algorithm, based on the property that the path of optimal solutions of the SVMs with respect to the preceding parameters is piecewise linear. This partial parametric path algorithm requires the solution of just one quadratic programming problem, and a number of linear systems of equations. Thus it can significantly reduce the computational requirements of the algorithm. To systematically explore the different weights to assign to the misclassification costs, we combine the partial parametric path algorithm with a variable neighborhood search method. Our numerical experiments show the efficiency and reliability of the proposed partial parametric path algorithm.

KEYWORDS: Multi-class SVM, Piecewise Linearity, Partial Parametric Path Algorithm, Variable Neighborhood Search

1 Introduction

SVM is a popular approach to solve classification problems. The effectiveness of SVM methods has not only been demonstrated in a very large number of experiments [2, 15, 20], but it is also proven in theory, see [13, 16, 17]. In the classical binary SVM setting for the nonlinearly separable case, the classification problem has two objectives: avoid overfitting and limit any classification errors. This is usually modeled by combining both objectives using a parameter C, $1/2\|\omega\|^2+CF(\sum_{i=1}^{l}\xi_i)$, see [4]. This parameter can be seen to represent a trade-off between training and testing misclassification costs. The aim is to construct a suitable classifier which

[*]L. Liu, Statistics Department, Universidad Carlos III de Madrid, C/ Madrid 126, 28903 Getafe (Madrid), Spain, Email: lliu@est-econ.uc3m.es

[†]B. Martín-Barragán, University of Edinburgh Business School, 29 Buccleuch Place, Edinburgh EH8 9JS, United Kingdom, Email: Belen.Martin@ed.ac.uk

[‡]F.J. Prieto, Universidad Carlos III de Madrid, C/ Madrid 126, 28903 Getafe (Madrid), Spain, Email: fjp@est-econ.uc3m.es

has high classification ability for the whole instance population. Attaining this aim requires the choice of a proper value for parameter C.

Selecting a default value for C may not provide acceptable results, as the optimal values from the SVM have been shown to depend critically on the choice of the value of C [3, 6, 10]. Usually, grid search is used to find an appropriate value of this parameter[6]. This approach is time-consuming, especially when we have big datasets to deal with. In [3], they treat C as a kernel parameter and minimize estimates of generalization errors by gradient descent. This approach depends on the differentiability of the estimates and still needs to solve the optimization problem (a quadratic program) several times. [10] explores the entire path of binary SVMs based on the fact that the corresponding Lagrange multipliers are piecewise-linear in C. They achieved very large savings in computational costs when constructing multiple classifiers for a set of C values.

Notice that the classic binary SVM doesn't take into account any a priori information (such as skewed class distributions, different misclassification costs). However, this information may be critical. For example in medical diagnosis, the difference between the classification costs of misclassifying a healthy person as ill and a diseased patient as healthy is large and can't be ignored. To take into account this information, [1, 18] use two different parameters C^+ and C^- associated to different classes. Also, [1] extends the path algorithm when considering asymmetric misclassification costs. In [12], differences between the weights of instances are taken into account and a corresponding path algorithm is proposed. Their numerical experiments show that these proposed path algorithms significantly reduce the computational cost to find proper parameter values for the binary SVMs.

As in real life we usually have more than two classes, the efficient two-class SVM approach has been extended for multi-class classifications. In [21], they propose a single-objective SVM to handle all the classes simultaneously. However, they consider all misclassification costs to be the same, and they use no a priori information related to the classes' distributions. A direct way to overcome this drawback is to assign different weights to the penalty terms for different misclassification errors in the objective function. Another way is to use multi-objective methods such as [14]. As we discussed before, the first proposal, based on a direct assignment of weight values, is not efficient because of the large computational costs required to choose suitable weights.

L.Wang and X.Shen [19] have proposed a path algorithm for multi-class classification problems based on the L_1 norm. It takes advantage of the property that their optimal solutions are piecewise-linear on a tuning parameter s, which controls the sum of the L_1 norms of all the slope vectors ω_c. They reconstruct the entire optimal path based on finding appropriate features characterizing changes in the active sets. To identify the joints and get the solutions, they need to construct sets of linear equations based on the derivatives of the slopes with respect to s. Our proposal shares its basic motivation with this one, but it aims to take advantage of the simplicity of L_2 optimization problems, and to take into account differences among classification costs. We introduce a partial-parametric-path algorithm (**PPPA**) for multi-class classification inspired by the previous one and by [1, 10]. The partial path begins with a starting solution obtained from a multi-class SVM problem, and it is extended to different values of the weight parameters, while checking if they are acceptable.

In this paper, we provide a general framework for the application of **PPPA**. If we have a sufficiently good classification performance at the starting point (a default set of values for the parameters), we don't need to apply **PPPA**. If the initial classification performance is not acceptable, we take advantage of the piecewise linearity of the optimal solutions to obtain efficient representations of the solution paths for alternative values of the parameters. By

controlling the changes in the active sets, we construct partial solution paths along some chosen parameter directions. To systematically explore the whole parameter space, we combine **PPPA** with a variable neighborhood search method (VNS). When using **PPPA**, we only need to solve one quadratic program to get our starting solution. All other solutions are obtained by solving systems of linear equations. Thus, **PPPA** is computationally efficient. From our experiments in Section 6, we have also verified that **PPPA** is robust, as it provides the same solutions as the ones obtained from the corresponding quadratic programming problems, in nearly all cases.

This paper is organized as follows: In Section 2, we present a single-objective multi-class SVM which takes into account differences in misclassification costs. In Section 3, we characterize the piecewise linear nature of the optimal solutions of the quadratic programs of interest. The components of the partial solution path are presented in Section 4. The combination of **PPPA** and VNS is introduced in Section 5. In Section 6, we describe and comment a set of experimental results showing that **PPPA** is efficient and reliable for multi-class classification problems. Finally, conclusions are presented in Section 7.

2 Our reference multi-class support vector machine

In what follows we assume that we have a training set $I = \{x_i\}_{i=1}^n \subseteq \mathbb{R}^l$, corresponding to m ($m \geq 3$) different classes, and values $y_i \in G = \{1, ..., m\}$ denoting the class membership of the vectors x_i. The aim of multi-class SVMs is to generate classifiers which can help us to predict with high accuracies the class membership of all the objects. For simplicity, we will just consider linear classifiers as nonlinear ones can be seen as linear in a higher dimensional space.

For multi-class classifications, the most commonly used single-objective methods are the all-together, one-against-all and one-against-one methods, [11]. The all-together and one-against-all methods are based on constructing m classifiers. Specifically, the all-together method maximizes the sum of all the margins and minimizes the penalty variables simultaneously within a single quadratic programming. Each of the margins is constructed by considering all instances of one class vs those of all the remaining classes; a similar approach is used in one-against-all method. One-against-all methods solve m quadratic programs to obtain the m classifiers. In all these cases, the presence of asymmetries in the misclassification costs, for example, may have a significant effect on the accuracies of the classifiers.* A one-against-one method constructs $m(m-1)/2$ classifiers and each of them are obtained from a binary SVM which just considers a pair of classes. In [11], experiments show that in general a one-against-all method does not achieve accuracies as high as a one-against-one method. In this paper we prefer to construct $m(m-1)/2$ classifiers and maximize all the pairwise margins. Of course, we could use the path-algorithm (introduced in [1]) on each of the binary SVMs from the one-against-one method to find a satisfactory choice of parameters. But when m is large, this would require tracking a very large number of paths, with very high associated computational costs. In order to take advantage of the high accuracy of a one-against-one method and the compactness of an all-together method, we have chosen to construct a single-objective multi-class SVM maximizing all the pairwise margins and

*A one-against-all classifier is constructed from m binary SVMs. Each of these binary SVMs considers only one class as the positive class and all the remaining classes as the negative class. So we can see that the difference between the sizes of positive and negative classes can be quite large. The classification accuracies would be affected because the classical binary SVM gives the same weight to the penalties of the positive and negative classes misclassification errors.

minimizing all the penalties at the same time.

We construct our classifiers as follows:

- The classifier (discriminating hyperplane) separating the training data from class p and that from class q, is given by:

$$L^{pq} : f^{pq}(x) = (\omega^{pq})^T x + \beta^p - \beta^q = 0, q > p, p, q \in G.$$

Ideally, we would like to have all class p objects lying above hyperplane $L^{pq}, q \neq p, p, q \in G$, and all class q objects lying below L^{pq}. If there exist hyperplanes such that the training objects satisfy this ideal situation, we say that the training objects are linearly separable. But usually we have linearly nonseparable data, and we need to take into account both margins and misclassification errors. So, by maximizing all the margins of the $m(m-1)/2$ classifiers and minimizing the misclassification penalties, we construct a single-objective SVM,

$$\begin{aligned} \min_{W,\beta,\xi} \quad & \frac{1}{2}\sum_{p=1}^{m}\sum_{q>p}(\omega^{pq})^T\omega^{pq} + \sum_{p=1}^{m}\sum_{q\neq p}\sum_{x\in I_p} t_0^{pq}\xi_x^{pq}, \\ s.t. \quad & (\omega^{pq})^T x + \beta^p - \beta^q + \xi_x^{pq} \geq 1, x \in I_p, q > p, p, q \in G, \\ & -(\omega^{pq})^T x - \beta^p + \beta^q + \xi_x^{qp} \geq 1, x \in I_q, q > p, p, q \in G, \\ & \xi_x^{pq} \geq 0, x \in I_p, q \neq p, p, q \in G, \end{aligned} \tag{1}$$

Here, $I_p = \{x \in I | x \text{ belongs to class } p\}$, and t_0^{pq} denotes the weights for each of the penalties. Misclassification penalties ξ_x^{pq} are introduced to avoid overfitting and to guarantee the existence of solutions for (1) in the linearly nonseparable case. To take into account the possible differences in misclassification costs, we introduce different weights $t_0^{pq}, q \neq p, p, q \in G$ for each misclassification penalties associated to different classes and classifiers.

Note that (1) is a quadratic program, whose solution provides the information to define the corresponding classifiers. A difficulty is the choice of acceptable values for the weights t_0^{pq}, that is, values that yield classifiers which have high accuracies and low misclassification costs.

This paper focuses on the introduction of efficient procedures to determine acceptable sets of weights. Our approach is based on a two-step approach:

- We select an initial set of weights, and a "search direction" on the space of these weights.
- We find the optimal combination of weights on this one-dimensional space by building a partial path from the optimal solution of (1) as the starting point, and conducting a search on these solutions.
- Finally, we modify the search directions using a variable neighborhood search method (VNS) and repeat this process until we are close enough to an acceptable solution.

To build the partial path we solve the following program as a function of C:

$$\begin{aligned} \min_{W,\beta,\xi} \quad & \frac{1}{2}\sum_{p=1}^{m}\sum_{q>p}(\omega^{pq})^T\omega^{pq} + \sum_{p=1}^{m}\sum_{q\neq p}\sum_{x\in I_p} t_0^{pq}\xi_x^{pq} + C\sum_{p=1}^{m}\sum_{q\neq p}\sum_{x\in I_p} t_1^{pq}\xi_x^{pq}, \\ s.t. \quad & (\omega^{pq})^T x + \beta^p - \beta^q + \xi_x^{pq} \geq 1, x \in I_p, q > p, p, q \in G, \\ & -(\omega^{pq})^T x - \beta^p + \beta^q + \xi_x^{qp} \geq 1, x \in I_q, q > p, p, q \in G, \\ & \xi_x^{pq} \geq 0, x \in I_p, q \neq p, p, q \in G, \end{aligned} \tag{2}$$

Here $t_1^{pq}, q \neq p, p, q \in G$ denotes the direction along which we construct the partial path.

After identifying appropriate parameter values and building the corresponding classifiers, we use majority voting (also known as 'Max Wins') to define our classification rule as in [11]: For observation x, if $\omega^{pq}x + \beta^p - \beta^q > 0$, then the vote for it belonging to the p-th class is increased by one. Otherwise, the vote for the q-th class is increased by one. After this procedure is completed, x is assigned to the class with the largest vote. In the case that two classes have identical votes, the one with smaller index is selected.

3 Optimal classifiers are piecewise affine functions of C

In this Section, we prove that the solutions of problem (2) are piecewise affine wrt C. This is the basic property on which the efficiency of our proposal rests. We also introduce simple characterizations for the solutions of interest.

As the classifiers $L^{pq} : (\omega^{pq})^T x + \beta^p - \beta^q = 0, q > p, p, q \in G$ depend only on the differences $\beta^p - \beta^q$ (instead of the values β^p), without loss of generality we will set $\beta^1 = 0$ in all that follows.

We will use a slightly modified notation, to simplify the representation of the optimal solutions of (2). In particular, we will work with the observations projected onto a higher dimensional space. Let

$$X_x^{pq} = \left(\delta_{x,pq}^{12}, \delta_{x,pq}^{13}, \cdots, \delta_{x,pq}^{(m-1)m}\right), \quad \text{where} \quad \delta_{x,pq}^{ij} = \begin{cases} x, & \text{if } (i,j) = (p,q), \\ -x, & \text{if } (i,j) = (q,p), \\ 0, & \text{otherwise.} \end{cases}$$

Let $\omega = (\omega^{12^T}, \omega^{13^T}, \cdots, \omega^{(m-1)m^T})^T$ and $\beta = (\beta^2, \beta^3, \cdots, \beta^m)^T$, so that we can write $(\omega^{pq})^T x + \beta^p - \beta^q = (\omega)^T X_x^{pq} + \beta^p - \beta^q$. Let X^{pq} denote a matrix with row vectors equal to $X_x^{pq}, x \in I_p$, while ξ^{pq} denotes a vector containing the values $\xi_x^{pq}, x \in I_p$. Define

$$X = \begin{pmatrix} X^{12} \\ X^{21} \\ \vdots \\ X^{(m-1)m} \\ X^{m(m-1)} \end{pmatrix}, \xi = \begin{pmatrix} \xi^{12} \\ \xi^{21} \\ \vdots \\ \xi^{(m-1)m} \\ \xi^{m(m-1)} \end{pmatrix} \text{ and } t_k = \begin{pmatrix} t_k^{12}\vec{1}_{n^1 \times 1} \\ t_k^{21}\vec{1}_{n^2 \times 1} \\ \vdots \\ t_k^{(m-1)m}\vec{1}_{n^{m-1} \times 1} \\ t_k^{m(m-1)}\vec{1}_{n^m \times 1} \end{pmatrix}, k = 0, 1,$$

where $n^p, p \in G$ is the number of class p training instances. Then, we can rewrite (2) as follows:

$$\begin{aligned} \min_{\omega,b,\xi} \quad & \frac{1}{2}\|\omega\|^2 + (t_0)^T\xi + C(t_1)^T\xi, \\ \text{s.t.} \quad & X\omega + R\beta + \xi \geq \vec{1}_{(m-1)n \times 1}, \\ & \xi \geq \vec{0}_{(m-1)n \times 1}, \end{aligned} \tag{3}$$

where R is the coefficient matrix of the variables β in (2).

As the objective function of (3) is quadratic (positive semidefinite) and the constraints are affine functions, the corresponding KKT conditions are necessary and sufficient for optimality. These KKT conditions for (3) are:

$$\omega = X^T\lambda, \tag{4}$$

$$R^T\lambda = \vec{0}_{(m-1)\times 1}, \tag{5}$$
$$\lambda + \mu = t_0 + Ct_1, \tag{6}$$
$$X\omega + R\beta + \xi \geq \vec{1}_{(m-1)n\times 1}, \tag{7}$$
$$\xi \geq \vec{0}_{(m-1)n\times 1}, \tag{8}$$
$$\lambda \geq \vec{0}_{(m-1)n\times 1}, \tag{9}$$
$$\mu \geq \vec{0}_{(m-1)n\times 1}, \tag{10}$$
$$\lambda^T(\vec{1}_{(m-1)n\times 1} - X\omega - R\beta - \xi) = 0, \tag{11}$$
$$\mu^T\xi = 0, \tag{12}$$

where λ denotes the multipliers of $X\omega + R\beta + \xi \geq \vec{1}_{(m-1)n\times 1}$ and μ denotes the multipliers of $\xi \geq \vec{0}_{(m-1)n\times 1}$.

We define the following relevant active sets:

- The indices of the above-margin objects: $A = \{i \mid X_i\omega + R_i\beta > 1\}$,
- The indices of the below-margin objects: $B = \{i \mid X_i\omega + R_i\beta < 1\}$,
- The indices of the on-margin objects: $O = \{i \mid X_i\omega + R_i\beta = 1\}$,

where X_i and R_i denotes the i-th row of X and R respectively. As each constraint is associated to one object, these definitions allow us to separate the objects into three disjoint sets.

Based on the preceding KKT conditions, we refine the preceding definitions as follows:

$$\begin{aligned}
&i \in A, \text{ if and only if } \lambda_i = 0,\\
&i \in B, \text{ if and only if } \lambda_i = t_{0i} + Ct_{1i},\\
&i \in O, \text{ if and only if } \ 0 < \lambda_i < t_{0i} + Ct_{1i}.
\end{aligned}$$

Let X_A denote the sub-matrix of X collecting all rows X_i with $i \in A$. Similarly, we define X_B, X_O, R_A, R_B, and R_O. Also, let λ_O denote the sub-vector of λ corresponding to the components λ_i with $i \in O$. Similarly, we introduce $\lambda_A, \lambda_B, t_{0A}, t_{0B}, t_{0O}, t_{1A}, t_{1B}$ and t_{1O}. We have

$$\begin{aligned}
&\omega = X_O^T\lambda_O + X_B^T t_{0B} + CX_B^T t_{1B},\\
&R_O^T\lambda_O + R_B^T t_{0B} + CR_B^T t_{1B} = \vec{0}_{(m-1)\times 1},\\
&X_O\omega + R_O\beta = \vec{1}_{n_O\times 1},
\end{aligned}$$

where n_O denotes the size of the active set O.

From these definitions and the optimality conditions, the solutions of problem (3) can be computed by solving the system

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & X_O^T \\ \mathbf{0} & \mathbf{0} & R_O^T \\ X_O & R_O & \mathbf{0} \end{pmatrix} \begin{pmatrix} \omega \\ \beta \\ -\lambda_O \end{pmatrix} = \begin{pmatrix} X_B^T t_{0B} \\ R_B^T t_{0B} \\ \vec{1}_{n_O\times 1} \end{pmatrix} + C \begin{pmatrix} X_B^T t_{1B} \\ R_B^T t_{1B} \\ \vec{0}_{n_O\times 1} \end{pmatrix}, \tag{13}$$

where X_O is a $n_O \times \frac{m(m-1)l}{2}$ matrix and R_O is a $n_O \times (m-1)$ matrix.

We will make use of the following auxiliary result characterizing some properties of the coefficient matrix for system (13).

Lemma 3.1. *Given active sets O, A and B, the necessary and sufficient conditions for the coefficient matrix of (13) to be nonsingular is that the column vectors of R_O are linearly independent and the row vectors of $\begin{pmatrix} X_O & R_O \end{pmatrix}$ are also linearly independent.*

Proof. Consider an auxiliary system having the same coefficient matrix as system (13),

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & X_O^T \\ \mathbf{0} & \mathbf{0} & R_O^T \\ X_O & R_O & \mathbf{0} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \vec{0}. \tag{14}$$

We will use it to study the singularity of the coefficient matrix of (13), based on the properties of the solutions of (14).

- Necessity: Suppose the coefficient matrix of (13) and (14) is nonsingular, then the unique solution of (14) is $u = \vec{0}$.

  If the column vectors of R_O are linearly dependent, there exists $u_2 \neq \vec{0}$ such that $R_O u_2 = \vec{0}$. The coefficient matrix of (13) cannot be invertible in this case, as we can choose $u_1 = \vec{0}$, $u_3 = \vec{0}$ and obtain a nonzero vector in the null space of this matrix, contradicting our assumption. It follows that if the coefficient matrix of (13) is nonsingular, the column vectors of R_O must be linearly independent.

  Analogously, if the row vectors of $\begin{pmatrix} X_O & R_O \end{pmatrix}$ are linearly dependent, there exists $u_3 \neq \vec{0}$ such that

  $$\begin{pmatrix} X_O^T \\ R_O^T \end{pmatrix} u_3 = \vec{0}.$$

  Again, in this case we can choose $u_1 = \vec{0}$, $u_2 = \vec{0}$ and obtain a nonzero vector in the null space of this matrix. This contradicts our assumption, implying that we must have linearly independent row vectors for $(X_O \; R_O)$.

- Sufficiency: Assume that the column vectors of R_O and the row vectors of $\begin{pmatrix} X_O & R_O \end{pmatrix}$ are linearly independent. This implies that $M \triangleq X_O X_O^T + R_O R_O^T$ and $R_O^T R_O$ are invertible, and M is positive definite.

  From (14) we have

  $$\begin{aligned} &R_O^T u_3 = \vec{0} \Rightarrow R_O R_O^T u_3 = \vec{0} \\ &u_1 = -X_O^T u_3 \\ &X_O u_1 + R_O u_2 = \vec{0} \Rightarrow R_O u_2 = X_O X_O^T u_3 = (X_O X_O^T + R_O R_O^T) u_3 \Rightarrow u_3 = M^{-1} R_O u_2 \\ &R_O^T u_3 = R_O^T M^{-1} R_O u_2 = \vec{0}. \end{aligned}$$

  As M is positive definite, $R_O^T M^{-1} R_O$ is at least positive semidefinite. As $u_2^T R_O^T M^{-1} R_O u_2 = 0$, from the positive-definiteness of M we have $R_O u_2 = \vec{0}$. As R_O has full column rank, it must hold that $u_2 = \vec{0}$. As $u_3 = M^{-1} R_O u_2$ and $u1 = -X_O^T u_3$, we have $u_1 = \vec{0}$ and $u_3 = \vec{0}$. It follows that the coefficient matrix of (13) and (14) is nonsingular.

□

To present our existence result, we introduce the following regularity condition:

$\mathcal{C}1$: The row vectors of $\begin{pmatrix} X_O & R_O \end{pmatrix}$ are linearly independent.

The next result provides the desired characterization of the solutions.

Theorem 3.2. *Under condition $\mathcal{C}1$, the solution of (13) defines a piecewise-affine optimal solution of (3).*

Proof. As $\beta = (\beta^2, \beta^3, \cdots, \beta^m)^T$, the number of columns in R_O and its maximum column and row rank is $m-1$. Consider an optimal solution of (3) defined by $(\omega_0, \beta_0, \xi_0)$.

Assume that R_O does not have full rank. Then, there exists a non-zero vector u such that $R_O u = 0$. Define $\beta_* = \beta_0 + \epsilon u$, and

$$\xi^{pq}_{B*} = \vec{1} - X^{pq}_B \omega_0 - (\beta^p_* - \beta^q_*) = \xi^{pq}_{B0} - \epsilon(u^p - u^q).$$

As $R_O u = 0$ it holds that $X_O \omega_0 + R_O \beta_* = \vec{1}$ for any ϵ, and $(\omega_0, \beta_*, \xi_*)$ are feasible as long as $\epsilon_1 \leq \epsilon \leq \epsilon_2$, with

$$\epsilon_1 = \max\left\{ \max_{u^p - u^q < 0, p \neq q, p, q \in G} \frac{X^{pq}_B \omega_0 + \beta^p_0 - \beta^q_0 - \vec{1}}{-(u^p - u^q)}, \max_{u^p - u^q < 0, p \neq q, p, q \in G} \frac{X^{pq}_A \omega_0 + \beta^p_0 - \beta^q_0 - \vec{1}}{u^p - u^q} \right\} < 0,$$

$$\epsilon_2 = \min\left\{ \min_{u^p - u^q > 0, p \neq q, p, q \in G} \frac{X^{pq}_A \omega_0 + \beta^p_0 - \beta^q_0 - \vec{1}}{u^p - u^q}, \min_{u^p - u^q > 0, p \neq q, p, q \in G} \frac{X^{pq}_B \omega_0 + \beta^p_0 - \beta^q_0 - \vec{1}}{-(u^p - u^q)} \right\} > 0.$$

Select $\epsilon = \epsilon_1$ if

$$\sum_{p \neq q, p, q \in G, u^p - u^q < 0} (u^q - u^p)(t^{pq}_0 + C t^{pq}_1) \# B^{pq} > \sum_{p \neq q, p, q \in G, u^p - u^q > 0} (u^p - u^q)(t^{pq}_0 + C t^{pq}_1) \# B^{pq},$$

and $\epsilon = \epsilon_2 > 0$ otherwise. We have that

$$\begin{aligned} \Phi_* &= \frac{1}{2}\|\omega_0\|^2 + (t_0 + C t_1)^T \xi_*, \\ &= \Phi_0 + \epsilon \Bigg[\sum_{p \neq q, p, q \in G, u^p - u^q < 0} (u^q - u^p)(t^{pq}_0 + C t^{pq}_1) \# B^{pq} \\ &\quad - \sum_{p \neq q, p, q \in G, u^p - u^q > 0} (u^p - u^q)(t^{pq}_0 + C t^{pq}_1) \# B^{pq} \Bigg] \leq \Phi_0, \end{aligned}$$

where $\Phi_0 = \frac{1}{2}\|\omega_0\|^2 + (t_0 + C t_1)^T \xi_0$. As a consequence, there must exist an optimal solution including an additional support vector in O, the one defining the selected value for ϵ. This procedure can be repeated until the matrix R_O has full column rank.

As we can always find a solution such that the column vectors of R_O are linearly independent, if condition $\mathcal{C}1$ holds for active sets O, A and B, from Lemma 3.1 the inverse of the coefficient matrix of (13) exists, and (13) implies that the optimal solution $(\omega, \beta, \lambda_O)$ is an affine function of C.

Also, as $\xi_i = \max(0, 1 - X_i \omega - R_i \beta)$, $\lambda_A = \vec{0}$, $\lambda_B = t_{0B} + C t_{1B}$ and $\mu = t_0 + C t_1 - \lambda$, the optimal solutions $(\omega, \beta, \xi, \lambda, \mu)$ are affine with respect to C, under our conditions. Thus, under $\mathcal{C}1$ the optimal classifiers are piecewise affine functions of C. □

From this result, the coefficient matrix of (13) is singular only if the row vectors of $\begin{pmatrix} X_O & R_O \end{pmatrix}$ are linearly dependent. If this is the case, we could project the observations onto a higher dimensional space in which the number of on-margin observations may be smaller and the row vectors of the corresponding $\begin{pmatrix} X_O & R_O \end{pmatrix}$ matrix are linearly independent. Thus, for simplicity we will assume in what follows that the row vectors of $\begin{pmatrix} X_O & R_O \end{pmatrix}$ are linearly independent.

4 The partial parametric path algorithm

To be able to identify good parameter values for the weights of the misclassification errors t, we follow the procedure described in Section 2. The basic step in that procedure consists in the construction of a univariate path along a given parameter direction (selected using a VNS method). In this Section, we study the properties of this path and we present efficient methods to obtain it.

This procedure consists on the following steps:

- For a given starting parameter vector t_0 and a parameter direction t_1, we compute the solution to the SVM problem corresponding to $C = C_0 = 0$.
- Then we determine the largest increase in the parameter C that will not change the active sets, C_1. We obtain this value by making use of the linear structure of the optimal solutions in that interval, using Theorem 3.2.
- We update the active sets at C_k with $k = 1$ (a "joint" in the path), and we repeat the procedure for increasing k until some criterion is optimized, or a stopping criterion is satisfied. In our experiments, the criterion to optimize has been the training classification accuracy, and the termination criterion has been defined as reaching a prespecified maximum value for C.

4.1 Characterization of linear segments of the solution path

From Theorem 3.2, for given active sets the classifiers are affine with respect to parameter C. Define C_k as the k-th joint where there is a change in the active sets, B_k, A_k, O_k are the active sets corresponding to all values $C_k \leq C < C_{k+1}$, and λ_k, μ_k denote the corresponding multipliers.

As the assume that the row vectors of $\begin{pmatrix} X_{O_k} & R_{O_k} \end{pmatrix}$ are linearly independent, we can define

$$\begin{pmatrix} \omega_k^a \\ \beta_k^a \\ -\lambda_{k,O_k}^a \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & X_{O_k}^T \\ \mathbf{0} & \mathbf{0} & R_{O_k}^T \\ X_{O_k} & R_{O_k} & \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} X_{B_k}^T t_{0B_k} \\ R_{B_k}^T t_{0B_k} \\ \vec{1} \end{pmatrix},$$

and

$$\begin{pmatrix} \omega_k^b \\ \beta_k^b \\ -\lambda_{k,O_k}^b \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & X_{O_k}^T \\ \mathbf{0} & \mathbf{0} & R_{O_k}^T \\ X_{O_k} & R_{O_k} & \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} X_{B_k}^T t_{1B_k} \\ R_{B_k}^T t_{1B_k} \\ \vec{0} \end{pmatrix}.$$

Also,

$$\begin{aligned} &\lambda_{k,A_k}^a = \vec{0}, \quad \lambda_{k,A_k}^b = \vec{0}, \quad \lambda_{k,B_k}^a = t_{0B_k}, \quad \lambda_{k,B_k}^b = t_{1B_k}, \\ &\mu_k^a = t_0 - \lambda_k^a, \quad \mu_k^b = t_1 - \lambda_k^b, \\ &\xi_{k,A_k \cup O_k}^a = \vec{0}, \quad \xi_{k,B_k}^a = \vec{1} - X_{B_k}\omega_k^a - R_{B_k}\beta_k^a, \\ &\xi_{k,A_k \cup O_k}^b = \vec{0}, \quad \xi_{k,B_k}^b = -X_{B_k}\omega_k^b - R_{B_k}\beta_k^b. \end{aligned}$$

Then the optimal solution $(\omega, \beta, \xi, \lambda, \mu)$ of (3) with respect to C, $C_k \leq C < C_{k+1}$ is given

by:

$$\begin{pmatrix} \omega \\ \beta \\ \xi \\ \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} \omega_k^a \\ \beta_k^a \\ \xi_k^a \\ \lambda_k^a \\ \mu_k^a \end{pmatrix} + C \begin{pmatrix} \omega_k^b \\ \beta_k^b \\ \xi_k^b \\ \lambda_k^b \\ \mu_k^b \end{pmatrix}. \tag{15}$$

4.2 Finding the joint values C_k

Between joints, the active sets do not change and the optimal solutions of (3) are affine functions of C. But as we increase the value of C, the solution path (15) reaches values that may not satisfy some of the optimality conditions. In this Section, we describe how to obtain efficiently the value of the next joint (the value of C), where the active sets change.

As the value of C changes from the preceding joint, at some point it will become necessary to adjust the active sets. We can classify these adjustments into four cases:

- Case 1: Some above-margin object changes to become an on-margin one;
- Case 2: Some below-margin object changes to become an on-margin one;
- Case 3: Some on-margin object changes to an above-margin one;
- Case 4: Some on-margin object changes to a below-margin one.

We now provide the characterization of the joint values corresponding to each of the four cases, for $C \geq C_k$,

- For the first case, we check $X_i\omega + R_i\beta = 1, \text{for } i \in A_k$. We have

$$X_i(\omega_k^a + C\omega_k^b) + R_i(\beta_k^a + C\beta_k^b) = 1 \Rightarrow C = \frac{1 - (X_i\omega_k^a + R_i\beta_k^a)}{X_i\omega_k^b + R_i\beta_k^b}, i \in A_k.$$

  As $X_i(\omega_k^a + C_k\omega_k^b) + R_i(\beta_k^a + C_k\beta_k^b) > 1$, we only need to check $X_i\omega + R_i\beta = 1$, for $i \in A_k$ and $X_i\omega_k^b + R_i\beta_k^b < 0$. We define

$$C_{k+1}^1 = \min\left\{ \frac{1 - (X_i\omega_k^a + R_i\beta_k^a)}{X_i\omega_k^b + R_i\beta_k^b} | i \in A_k \text{ and } X_i\omega_k^b + R_i\beta_k^b < 0 \right\}.$$

- For the second case, we check $\xi_i = \xi_{k,i}^a + C\xi_{k,i}^b = 0$ for $i \in B_k$. As $\xi_{k,i}^a + C_k\xi_{k,i}^b > 0$ for $i \in B_k$, we only need to check $\xi_{k,i}^a + C\xi_{k,i}^b = 0$ for $i \in B_k$ and $\xi_{k,i}^b < 0$. We define

$$C_{k+1}^2 = \min\left\{ -\frac{\xi_{k,i}^a}{\xi_{k,i}^b} | i \in B_k \text{ and } \xi_{k,i}^b < 0 \right\}.$$

- For the third case, we check $\lambda_i = \lambda_{k,i}^a + C\lambda_{k,i}^b = 0$, for $i \in O_k$. As $\lambda_{k,i}^a + C_k\lambda_{k,i}^b > 0$ for $i \in O_k$, we only need to check $\lambda_{k,i}^a + C\lambda_{k,i}^b = 0$, for $i \in O_k$ and $\lambda_{k,i}^b < 0$. We define

$$C_{k+1}^3 = \min\left\{ -\frac{\lambda_{k,i}^a}{\lambda_{k,i}^b} | i \in O_k \text{ and } \lambda_{k,i}^b < 0 \right\}.$$

- For the last case, we check $\mu_i = \mu^a_{k,i} + C\mu^b_{k,i} = 0$ for $i \in O_k$. As $\mu^a_{k,i} + C_k\mu^b_{k,i} > 0$ for $i \in O_k$, we only need to check $\mu^a_{k,i} + C\mu^b_{k,i} = 0$ for $i \in O_k$ and $\mu^b_{k,i} < 0$. We define

$$C^4_{k+1} = \min\left\{-\frac{\mu^a_{k,i}}{\mu^b_{k,i}} | i \in O_k \text{ and } \mu^b_{k,i} < 0\right\}.$$

The next joint value is defined as:

$$C_{k+1} = \min\{C^1_{k+1}, C^2_{k+1}, C^3_{k+1}, C^4_{k+1}\}.$$

At C_{k+1}, we can update the new values $(\omega_{k+1}, \beta_{k+1}, \xi_{O_{k+1}})$ by observing the changes in the active sets and solving equations (13) for the updated active sets. Then we repeat the procedure with the updated active sets to find the next joint.

It would be possible to build an entire path starting from t_0 close to $\vec{0}$. But we have found that this is not useful in practice in many cases, as (1) would return $\omega^{pq} = \vec{0}, q > p, p, q \in G$ for this value, which is not a very useful choice for data classification. A reasonable initial guess for the values of the parameters would provide much better starting estimates and would reduce the amount of computation to carry out to obtain a good final estimate. Thus, our method is based on constructing partial paths, instead of trying to reconstruct the entire solution path.

5 Combining path-following with a VNS method

The partial path method introduced in Section 4 is used to find the best C^* among the parameters corresponding to the path along parameter direction t_1, starting from t_0. This value depends on the choice of t_0 and t_1. In this Section, we describe a VNS-based procedure [7, 8, 9] to obtain values for t_1 yielding good values for the parameters and good solutions for the multi-class SVM problem, in a systematic manner.

We consider a good solution to be one such that, if the real misclassification costs are known, yields lower misclassification costs on the training set. If the real misclassification costs are unknown, we search for a solution with fewer classification errors on the training set.

To complete the search procedure, we need to obtain:

- An initial direction t_{10}: If we know the real misclassification costs $\kappa^{pq}, q \neq p.p, q \in G$, we can start the partial path with $t^{pq}_0 = \kappa^{pq}/\kappa$ and $t^{pq}_{10} = t^{pq}_0$, where $\kappa = \sum_{q \neq p, p, q \in G} \kappa^{pq}$.

  If we don't know the real misclassification costs, we suggest taking the initial direction $t_0 = \vec{1}$ and $t_{10} = 0.01 \times \vec{1}$, as it is equivalent to the value used in general multi-class support vector machines [21] with the tradeoff parameter set as $1 + 0.01C$, although it does not consider the different misclassification costs. This is equivalent to assuming all the misclassification costs $\kappa^{pq} = 1, q \neq p, p, q \in G$.

  Starting from the initial point t_0 along t_{10}, we generate the corresponding partial path. To denote all the joints of the partial path, we introduce $PP(t_0, t_{10}) = \{C_k(t_0, t_{10}),\ k \geq 0\}$, where $C_k(t_0, t_{10})$ is the $k-$th joint. From the solutions obtained at the joints of $PP(t_0, t_{10})$, we choose the value C_{0*} corresponding to the best misclassification costs on training set I. Let

$$PI\kappa(C_{0*}, t_{10}) = \sum_{i \in I} \sum_{q \neq p, p, q \in G} \kappa^{pq} \#\{y_i = p, \hat{y}_i = q\}, \tag{16}$$

to denote the lowest misclassification costs on the training set along the partial path $PP(t_0, t_{10})$, where $\hat{y}_i$ is the class membership of x_i determined by the corresponding trained support vector machine. Note that when $\kappa^{pq} = 1$, (16) returns the classification error.

- A neighborhood structure: $N_\iota(t_{10}) = \{t_1 | \left| \frac{t_1^{pq}}{t_{10}^{pq}} - 1 \right| \leq 0.01\iota\}$ and $\iota \leq \iota_{max}$. Although most researchers use $\iota \leq 2$ [8], in the experiments of this paper, we have chosen $\iota_{max} = 10$. This choice has been made to consider a larger number of alternatives given its reduced computational cost, compared to that of other proposals.

- The algorithm will stop if at least one of these conditions is met:
  - SC1: the number of local searches ls reaches the maximum ls^{max}. In our experiments we set $ls^{max} = 100$.
  - SC2: perfect classification, i.e $PI\kappa(C_{0*}, t_{10}) = 0$.

Then we proceed the follows:

- Let $ls = 0$.
- While $ls \leq 100$ and $PI\kappa(C_{0*}, t_{10}) > 0$, repeat the following steps:
  - Step 1: Set $\iota = 1$;
  - Step 2: Until $\iota = \iota_{max}$, repeat the following steps:
    - Step 2.a: Generate a direction $\tilde{t}_{10}$ at random from the ι-th neighborhood of t_{10};
    - Step 2.b: Conduct a local search using $\tilde{t}_{10}$ as the initial solution by doing the following:
      - Step 2.b.1: Randomly generate $\tilde{t}_{1i}, i = 1, \cdots, N$ (we use $N = 10$ in our experiments) from the neighborhood $N_1(\tilde{t}_{10})$;
      - Step 2.b.2: For $\tilde{t}_{1i}$, generate the corresponding partial path $PP(t0, \tilde{t}_{1i})$. Along $PP(t0, \tilde{t}_{1i})$, we find
      $$C_{1i}^* = \arg \min_{C \in PP(t0, \tilde{t}_{1i})} \{PI\kappa(C, \tilde{t}_{1i})\};$$
      - Step 2.b.3: Select
      $$(C_{1*}, \tilde{t}_{1*}) = \arg \min_{(C_{1i}^*, \tilde{t}_{1i}), i=1,\cdots,10} \{PI\kappa(C_{1i}^*, \tilde{t}_{1i})\};$$
      - Step 2.b.4: Let $ls = ls + 1$.
    - Step 2.c: If $PI\kappa(C_{1*}, \tilde{t}_{1*}) < PI\kappa(C_{0*}, t_{10})$, take $t_{10} = \tilde{t}_{1*}, C_{0*} = C_{1*}$ and go back to Step 2.b, otherwise set $\iota = \iota + 1$ and go back to Step 2.a.

6 Numerical experiments

To test the efficiency and reliability of this parametric partial path algorithm, we have conducted several computational experiments. All these experiments have been implemented on a Macbook with 8 gigabytes of memory, using code written in R. The RMOSEK package has been used to solve the quadratic programs (1) defining the start point of the algorithm.

These experiments have been conducted on the following benchmark datasets: IRIS, WINE, SEEDS, VEHICLE, CAR (Car Evaluation), GLASS, SCC (Synthetic Control Chart Time Series) and CTG (Cardiotocography, raw data). All of them are available in the UCI Machine Learning Repository. A summary of the information for these data sets is listed in Table 1.

Table 1. Data set description

Data set	size of the data set	No. of Dim.	No. of classes
IRIS	150	4	3
WINE	178	13	3
SEEDS	210	7	3
VEHICLE	846	18	4
CAR	1728	6	4
GLASS	214	9	6
SCC	600	60	6
CTG	2126	35	10

Table 2 presents the classification performance obtained from the solutions computed at the starting point. As the real costs are unknown, we start with $t_0 = \vec{1}$ and $C = 0$. This means that at the starting point we don't consider the differences among misclassification costs, maximizing the pairwise margins and minimizing the training classification errors using the same weights for all of them.

Table 2. Classification accuracies at the starting point

Data set	IRIS	WINE	SEEDS	VEHICLE	CAR	GLASS	SCC	CTG
tr.ac	0.9831	**1**	0.9464	0.9226	0.8711	0.6488	**1**	**1**
te.ac	0.9667	0.9730	0.9048	0.7457	0.8129	0.6444	0.9833	0.9814

[1] tr.ac= classification accuracy on training set,
[2] te.ac= classification accuracy on testing set.

From Table 2, we can see that for data sets WINE, SCC, CTG and these starting values, we already find the best parameters, with the best training classification accuracies. So for these data sets we don't need to construct partial paths to find better parameters.

For each of the remaining data sets, we construct a partial path starting with $C = 0$ and $t_1 = 0.01 \times \vec{1}$, as we don't know the real misclassification costs; we assume that these misclassification costs are same. We also tried some randomly chosen values for t_0 and t_1. The corresponding results show the reliability of the partial path algorithm; for details, see Appendix B.

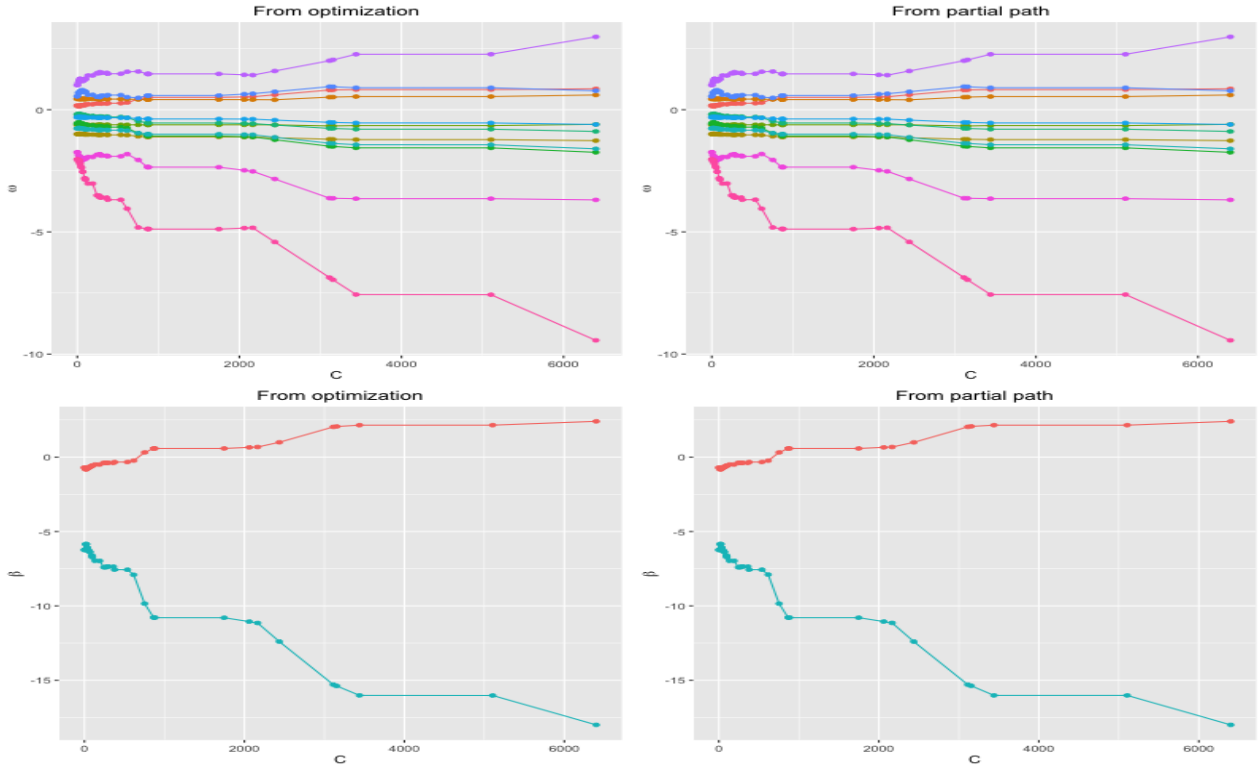
To test the reliability of the partial paths, at each joint we compute the corresponding solutions of (3) using RMOSEK. The following figures (from Figure 1 to Figure 8[†]) compare the solutions obtained from the partial path algorithm and directly from the optimization code. We should mention that, for both the CAR and VEHICLE data sets, the coefficient matrices are singular and do not satisfy condition $\mathcal{C}1$, as defined in Section 3. For these data sets we have projected the original data onto a higher-dimension space in which we obtain

[†]Figure 3 to Figure 8 can be found in Appendix A.

nonsingular coefficient matrices. We have used a modification of a nearest neighbor rule [5] to modify the data and to avoid this problem, in the following way:

- For the CAR data set, we add the euclidean distances from the instances to each of the means of I_1, I_2, I_3 and I_4, as additional coordinates for each instance;
- For the VEHICLE data set, we add the euclidean distances from the instances to the means, 25% quantiles and 75% quantiles of I_1, I_2, I_3 and I_4 as additional coordinates.

Figure 1. Comparison of the solutions from optimization problems and the partial path algorithm for IRIS data



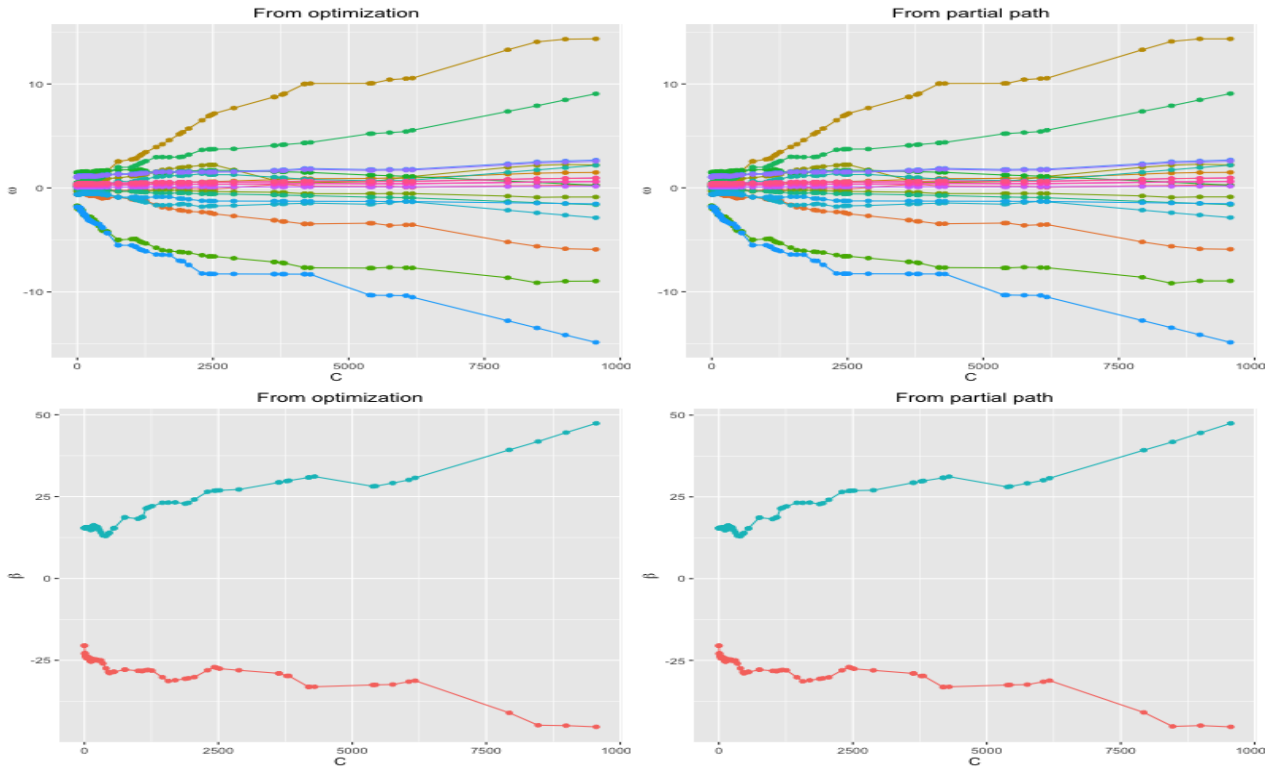
From Figure 1 to Figure 8, we can see that the **PPPA** procedure is very reliable, as the solutions at the joints computed from the optimization problem and **PPPA** are quite similar. Notice that using **PPPA** we need to solve only one quadratic problem (1), while optimization methods repeatedly solve (3) for different values of the parameters. Thus, **PPPA** offers significant savings in computation times. We show the execution times used for constructing the partial paths and completing the optimizations at all the joints of the paths in Table 3.

Table 3. Time to compute solutions at all joints

Data set	IRIS	SEEDS	VEHICLE	CAR	GLASS
t.op(s)	5.102	12.801	2016.6	1438.690	104.864
t.pp(s)	**0.207**	**0.338**	**92.418**	**16.545**	**9.480**

[1] t.op= time used for completing the optimizations at all the joints,
[2] t.pp= time used for constructing the partial path.

Figure 2. Compare the solutions gotten from optimization and partial path algorithm for SEEDS data



Note additionally that a traditional grid search method may miss some good solutions, and will offer in general solutions of lower quality than **PPPA**.

Up to this point, we have constructed paths using $t_1 = 0.01 \times \vec{1}$. We now introduce a variable neighborhood search to find better values of the parameters. The results are shown in Table 4.

Table 4. Results obtained before and after performing a variable neighborhood search

		Unknown misclassification costs			With simulated misclassification costs		
		b.C	ac.tr	ac.te	b.C	cc.tr	cc.te
IRIS	before	22.8497	0.9915	0.9667	22.8497	6	6
	after	22.8497	0.9915	0.9667	22.8497	6	6
SEEDS	before	7930.496	0.9940	0.9286	7930.496	2	7
	after	7930.496	0.9940	0.9286	7930.496	2	7
VEHICLE	before	22.9161	0.8740	0.8129	22.9161	135	65
	after	24.2008	**0.8756**	0.8129	25.0835	**133**	65
CAR	before	7678.431	0.9378	0.8266	7678.431	548	356
	after	7678.431	0.9378	0.8266	2590.884	**452**	329
GLASS	before	4560.535	0.8274	0.62222	4560.535	209	109
	after	6805.878	**0.8333**	0.6222	9935.552	**201**	118

[1] b.C= Best parameter C found among the paths,
[2] cc.tr= classification cost on the training set,
[3] cc.te= classification cost on the testing set.

From Table 4, we can see that for asymmetric training data (VEHICLE, GLASS) the variable neighborhood search provides better solutions with higher classification accuracies.

We have also conducted some experiments to study the case when classification costs are known and different. We simulated values for the real misclassification costs as integers randomly generated in the interval $(1, 10)$. We compare the performance of the proposed method before and after the variable neighborhood search in Table 4. We can see that when there are different misclassification costs, the proposed variable neighborhood search method provides better parameter values. This would imply that when misclassification costs are known and different, to obtain better results we should assign different weights to different misclassification errors.

7 Conclusion

In this paper we have proposed a method, **PPPA**, which starting from an optimal solution given by RMOSEK for multi-class SVMs, is able to identify good values for the weights of the misclassification costs with limited computational cost.

In general, we apply the proposed partial path algorithm in the following manner:

- Obtain a starting set of values for the parameters. Based on the optimization criterion (misclassification costs, for example), decide whether we need to construct a partial path.
- Along a selected parameter direction, construct a partial path from the starting point.
- Along the preceding path, find the best parameter C value. If the classification errors are not acceptable, combine the **PPPA** and a VNS method to systematically search for better parameter values.

Compared with a traditional grid search method, the partial path algorithm only needs to solve one quadratic program (1), while the grid search method repeatedly solves quadratic programs for each one of the parameter values tested. Thus, **PPPA** is much more efficient. Additionally, the quality of the solutions obtained from **PPPA** is higher than those from a traditional grid search method, as the number of potential solutions considered is much higher.

The partial path is constructed by following changes in the active sets. In our experiments, we have shown that the **PPPA** is efficient and reliable, because it gives us solutions which are almost the same as the ones obtained directly from the optimization problem, while requiring at most one tenth of the computational effort. From our experiments, we also see that if the misclassification costs are different (or the training data is asymmetric), a VNS method provides significantly better parameter values.

In summary, we conclude that the **PPPA** is an efficient and reliable procedure to find good parameter values for multi-class SVMs. Combining it with a VNS method helps us to systematically and efficiently explore a very large set of potential parameter weight values.

References

[1] Francis R Bach. Considering Cost Asymmetry in Learning Classifiers. *Jmlr*, 7:1713–1741, 2006.

[2] M P Brown, W N Grundy, D Lin, N Cristianini, C W Sugnet, T S Furey, M Ares, and D Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences of the United States of America*, 97(1):262–267, 2000.

[3] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[5] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[6] Frauke Friedrichs and Christian Igel. Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, 64(Esann):107–117, 2005.

[7] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*, volume 146. 2010.

[8] Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, may 2001.

[9] Pierre Hansen, Nenad Mladenović, and José a. Moreno Pérez. Variable neighbourhood search: Methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[10] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The Entire Regularization Path for the Support Vector Machine. *Test*, 5(2):1391–1415, 2004.

[11] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[12] Masayuki Karasuyama, Naoyuki Harada, Masashi Sugiyama, and Ichiro Takeuchi. Multi-parametric solution-path algorithm for instance-weighted support vector machines. *Machine Learning*, 88(3):297–330, 2012.

[13] Yi Lin. Support vector machines and the Bayes rule in classification. *Data Mining and Knowledge Discovery*, 6(3):259–275, 2002.

[14] K Tatsumi and T Tanino. Support vector machines maximizing geometric margins for multi-class classification. *TOP*, (22):815–840, 2014.

[15] Simon Tong and Daphne Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Journal of Machine Learning Research*, pages 45–66, 2001.

[16] V N Vapnik. *The Nature of Statistical Learning Theory*, volume 8. 1995.

[17] Vladimir N Vapnik. *Statistical Learning Theory*, volume 2. 1998.

[18] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, and Others. Controlling the

sensitivity of support vector machines. In *Proceedings of the international joint conference on artificial intelligence*, pages 55–60, 1999.

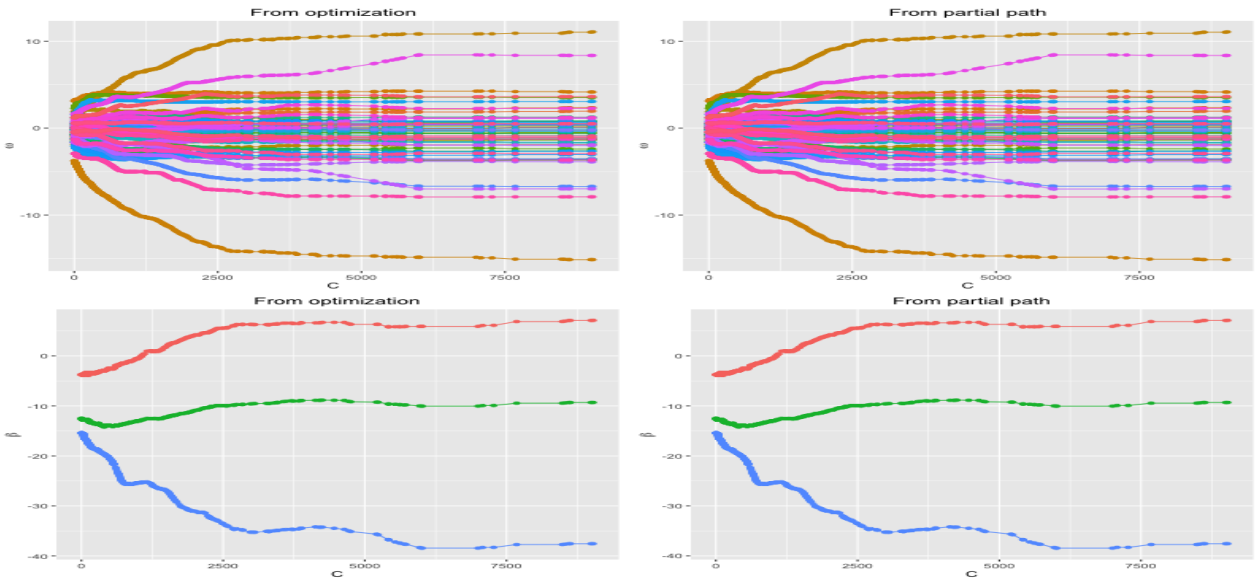
[19] Lifeng Wang and Xiaotong Shen. Multi-Category Support Vector Machines , Feature Selection and Solution Path. *Statistica Sinica*, 16:617–633, 2006.

[20] Xiaodong Wang and Jun Tian. Gene Selection for Cancer Classification using Support Vector Machines. *Computational and mathematical methods in medicine*, 2012:586246, 2012.

[21] J Weston and C Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, 1998.

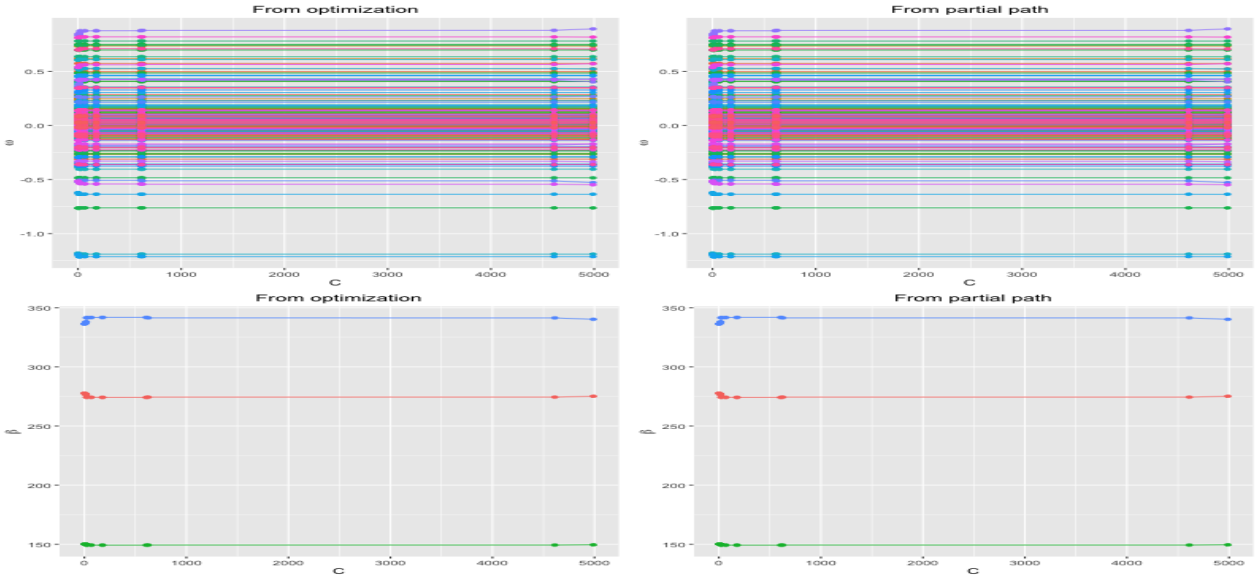
A Comparing The Solutions Obtained from Optimization and the Partial Path Algorithm, $t_0 = \vec{1}, t_1 = 0.01 \times \vec{1}$

Figure 3. Comparing the solutions obtained from optimization and the partial path algorithm for CAR data



‡

Figure 4. Comparing the solutions obtained from optimization and the partial path algorithm for VEHICLE data



‡For GLASS data, we separate the graph to four parts, because when I try to plot all the lines together, the R always quits. Same ways been used for the figures correspond to randomly start points and randomly t_1.

Figure 5. Comparing the solutions obtained from optimization and the partial path algorithm for GLASS data - Part 1

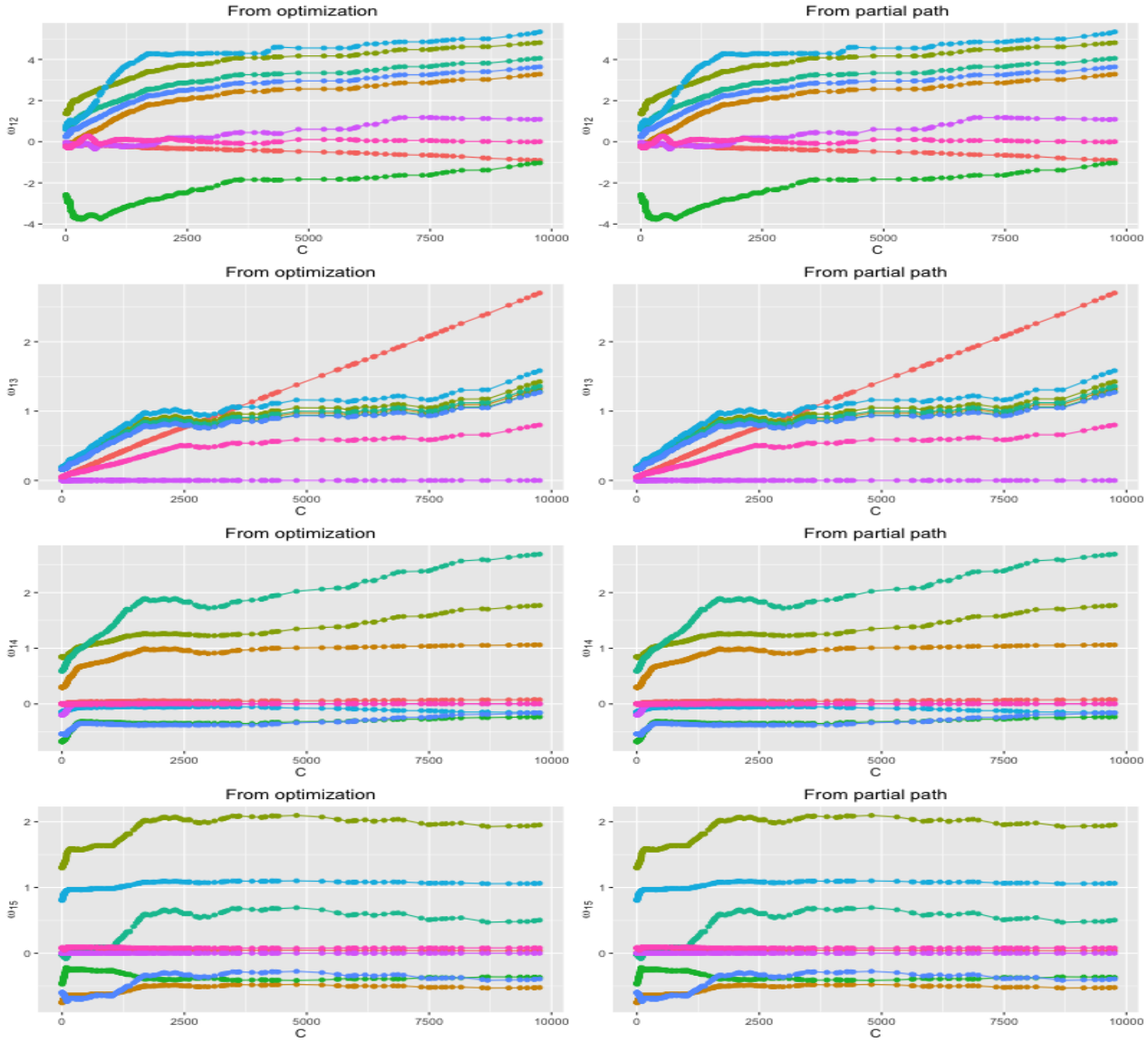


Figure 6. Comparing the solutions obtained from optimization and the partial path algorithm for GLASS data - Part 2

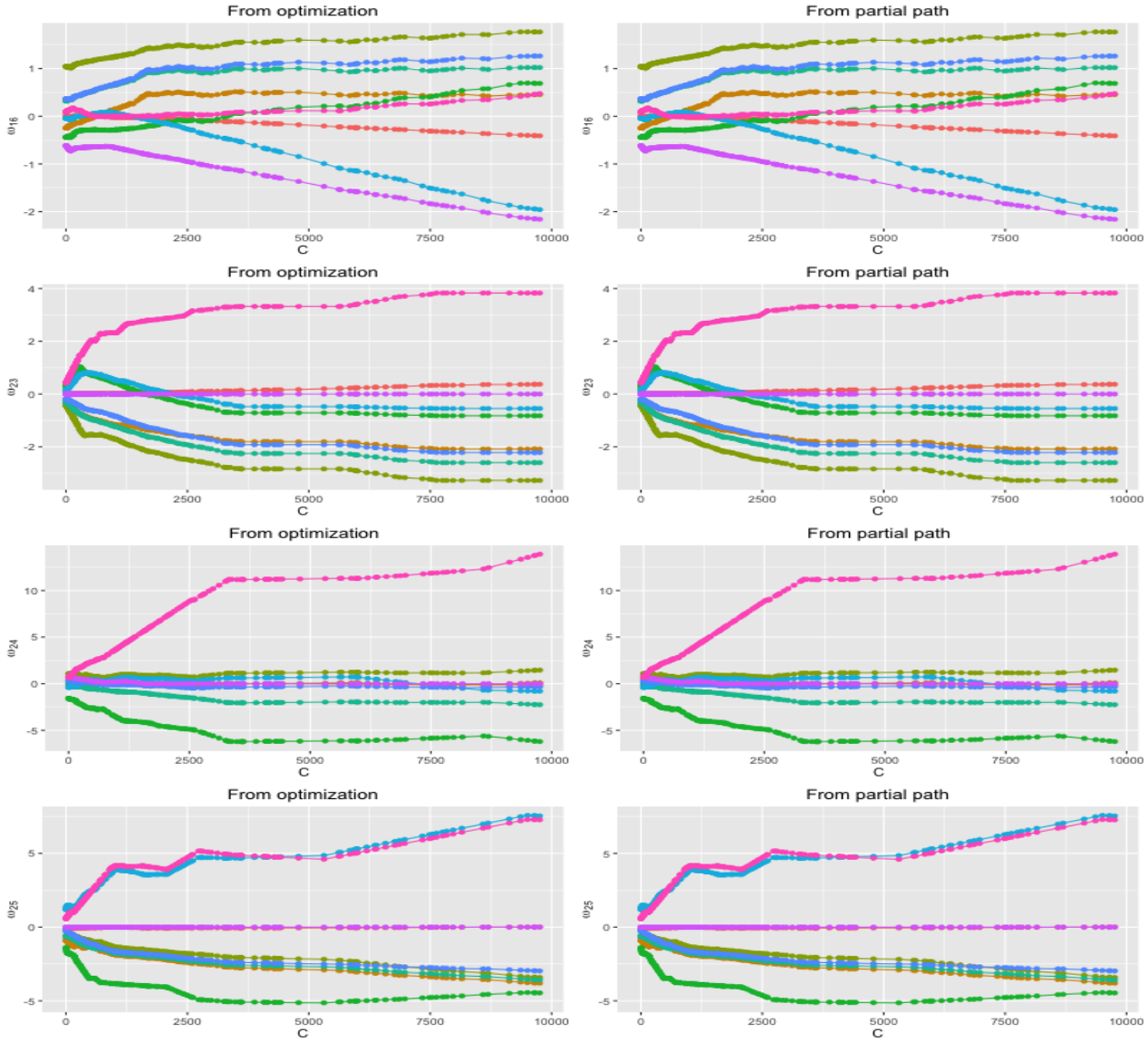


Figure 7. Comparing the solutions obtained from optimization and the partial path algorithm for GLASS data - Part 3

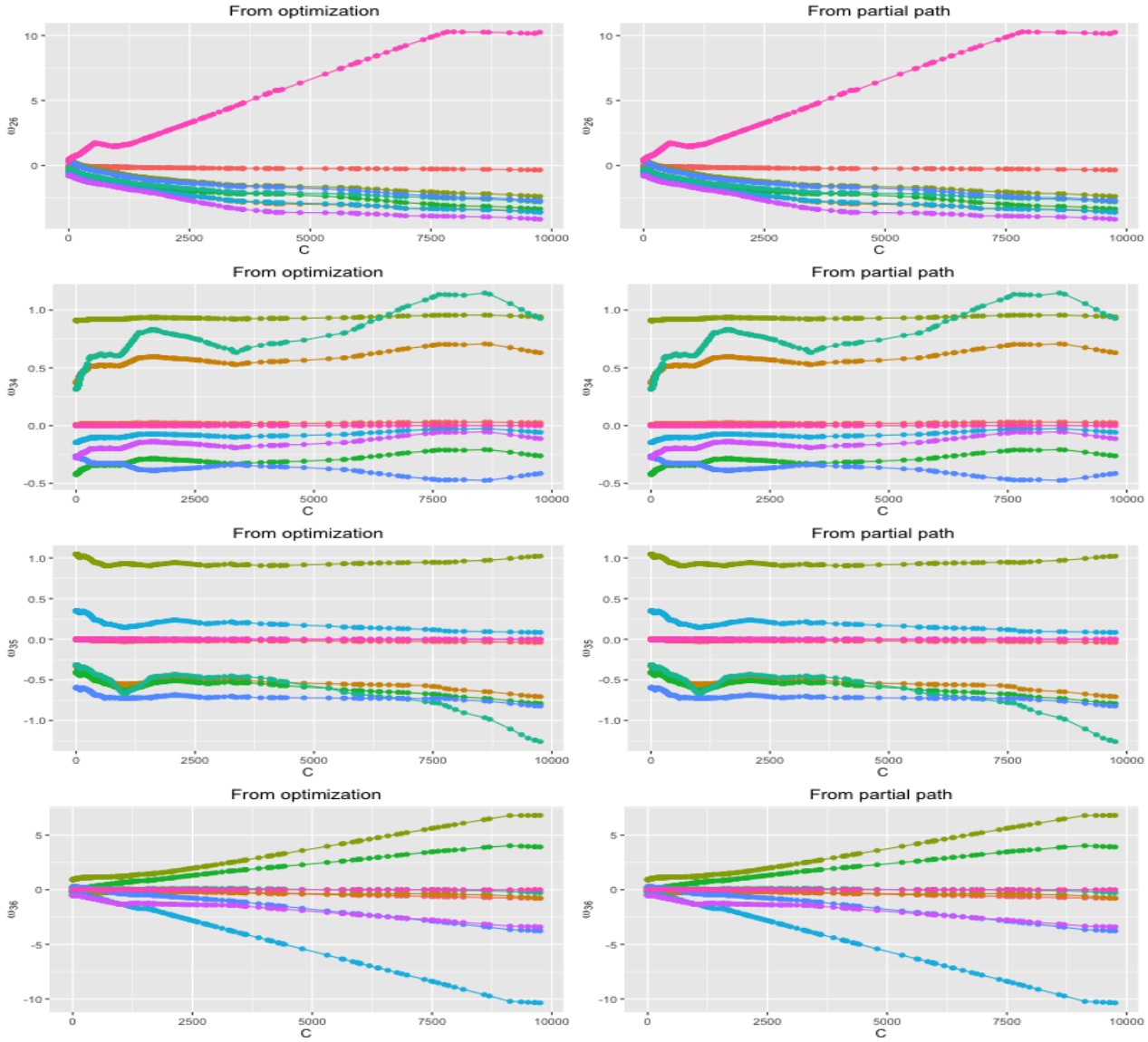
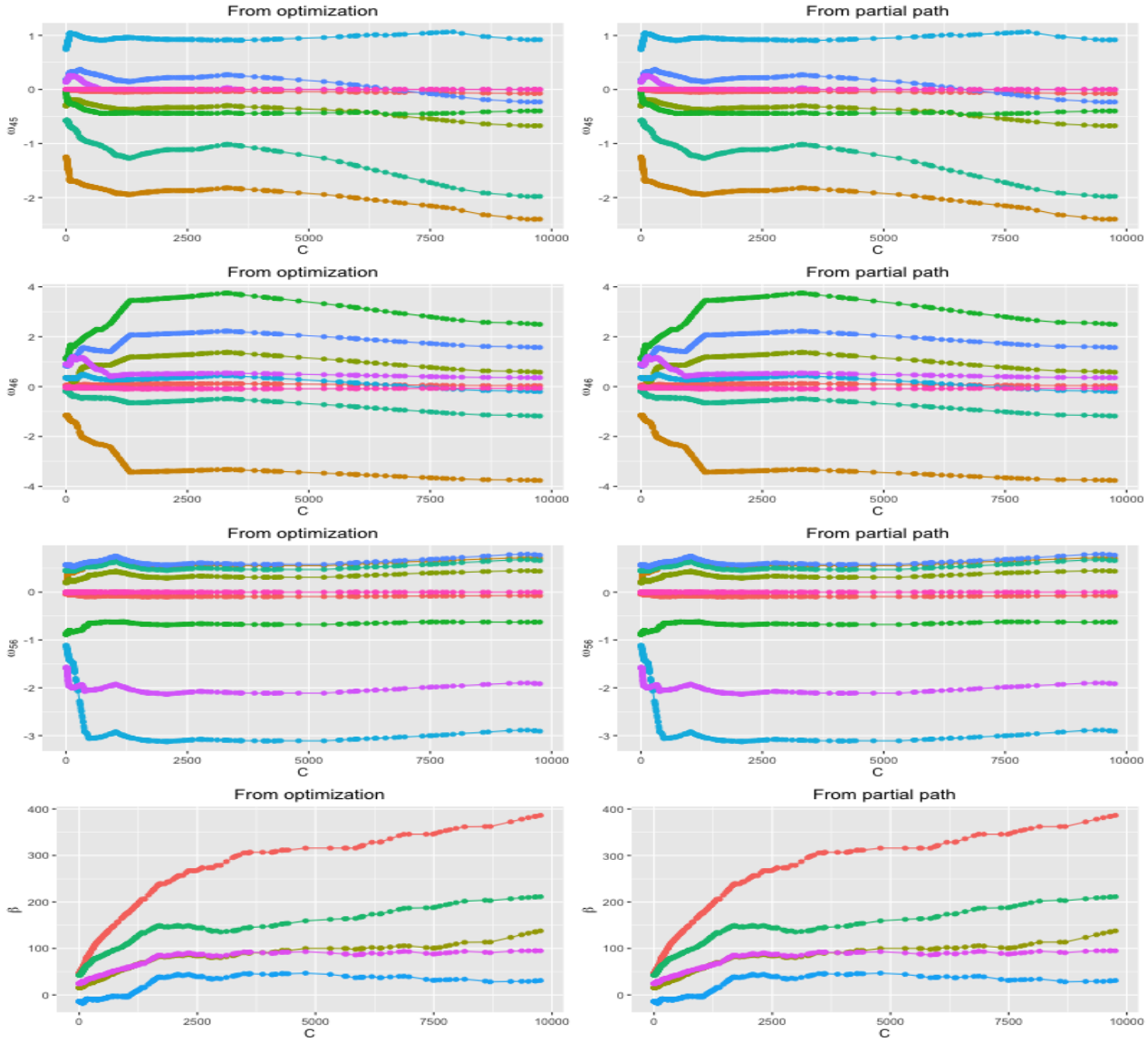


Figure 8. Comparing the solutions obtained from optimization and the partial path algorithm for GLASS data - Part 4



B Comparing the solutions obtained from optimization and the partial path algorithm, t_0, t_1 randomly chosen

Figure 9. Comparing the solutions obtained from optimization and the partial path algorithm for IRIS data

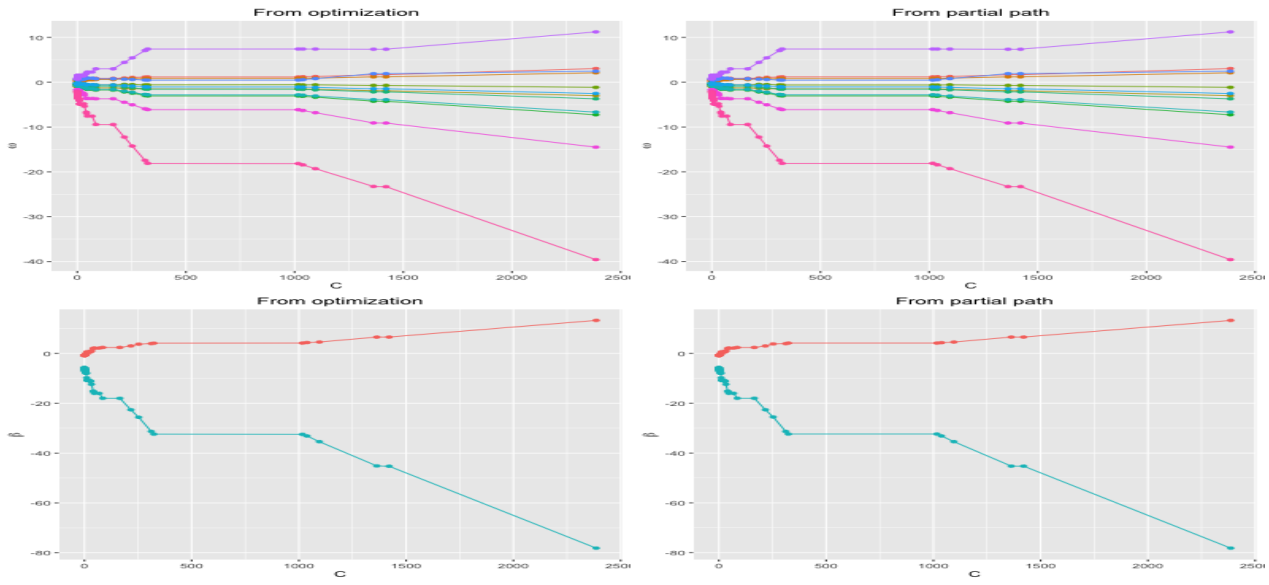


Figure 10. Comparing the solutions obtained from optimization and the partial path algorithm for SEEDS data

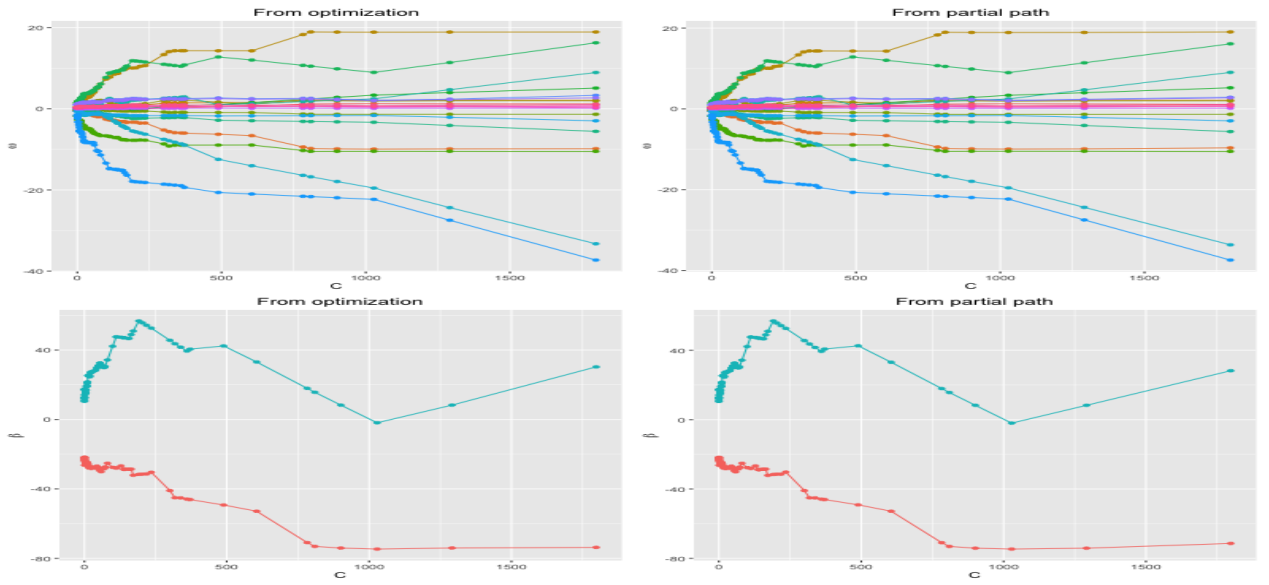


Figure 11. Comparing the solutions obtained from optimization and the partial path algorithm CAR data

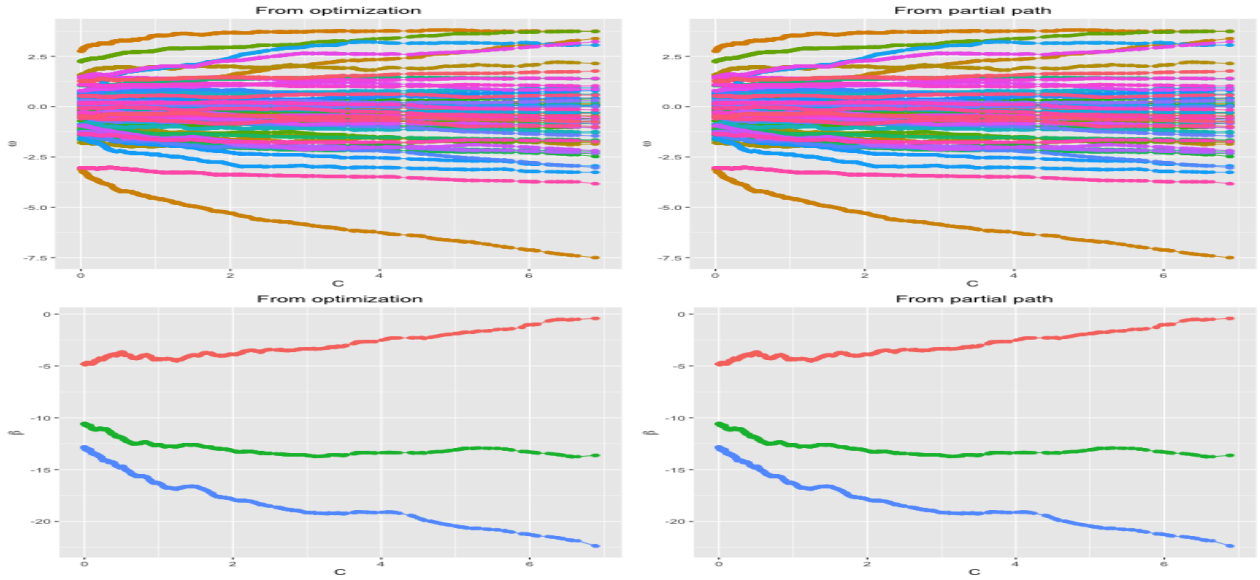


Figure 12. Comparing the solutions obtained from optimization and the partial path algorithm VEHICLE data

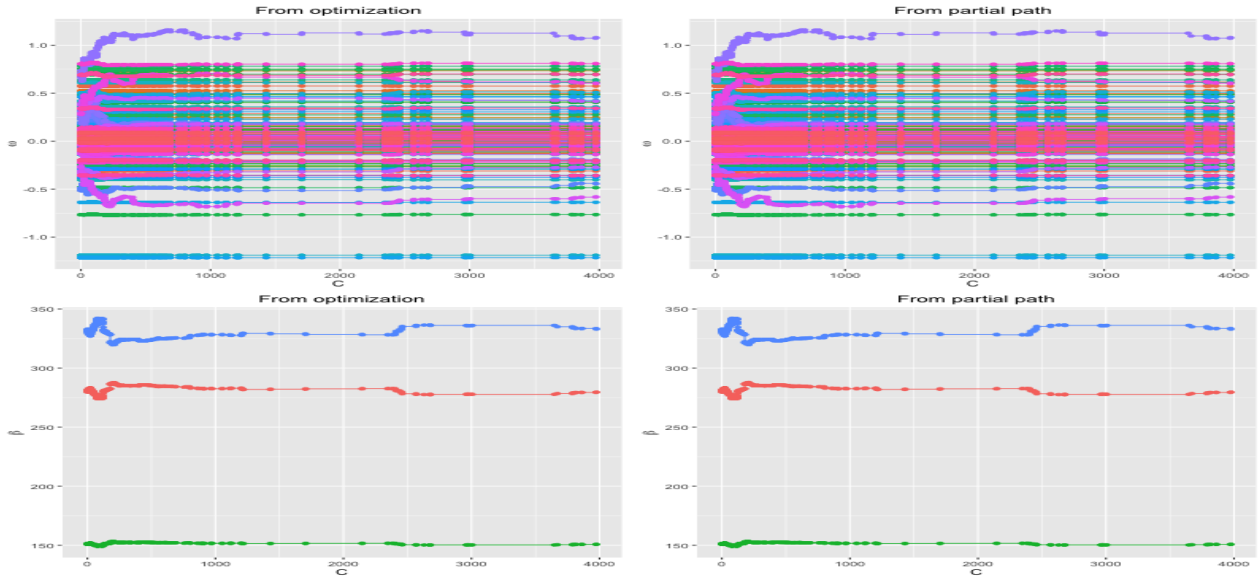


Figure 13. Comparing the solutions obtained from optimization and the partial path algorithm for GLASS data - Part 1

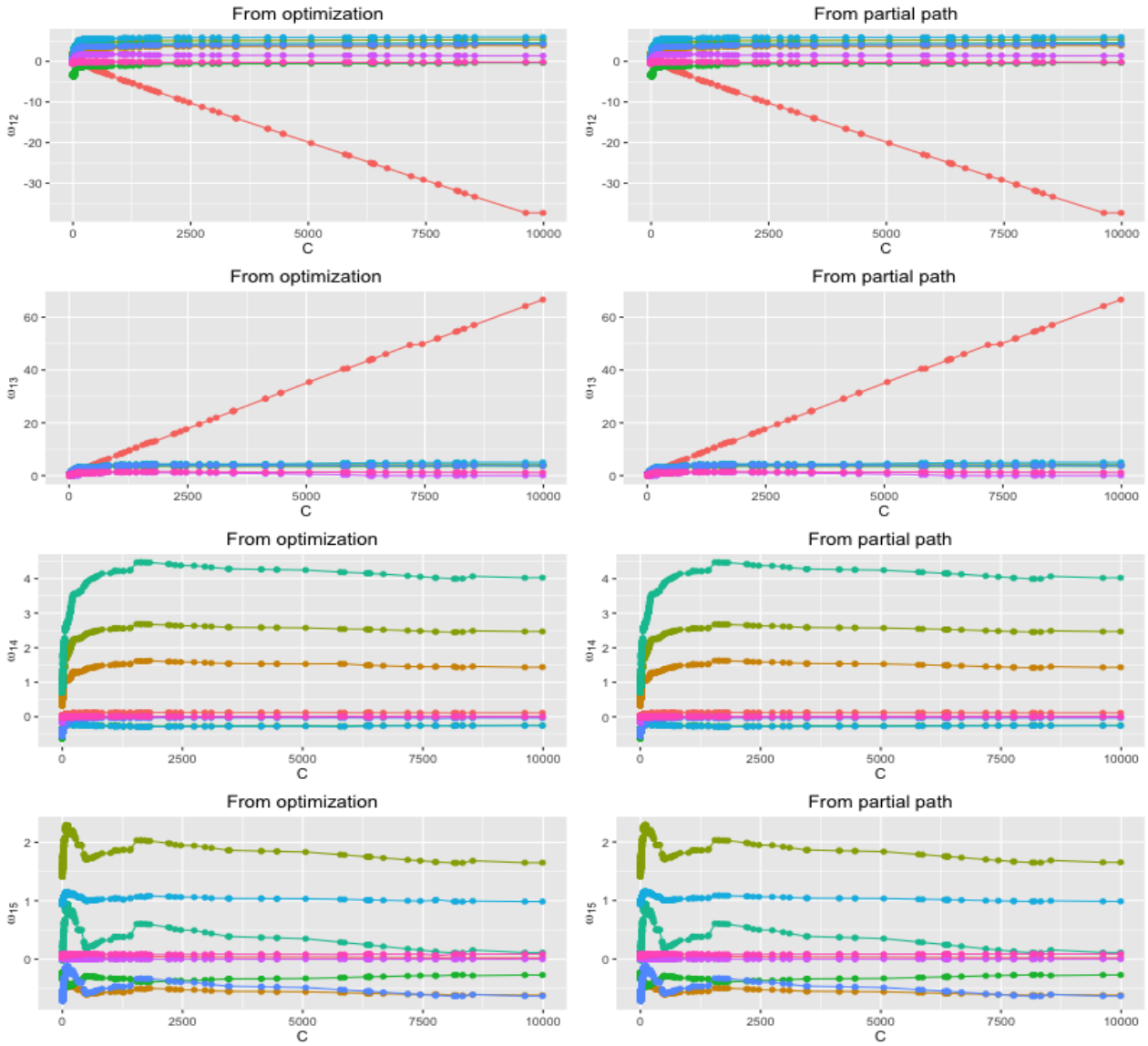


Figure 14. Comparing the solutions obtained from optimization and the partial path algorithm for GLASS data - Part 2

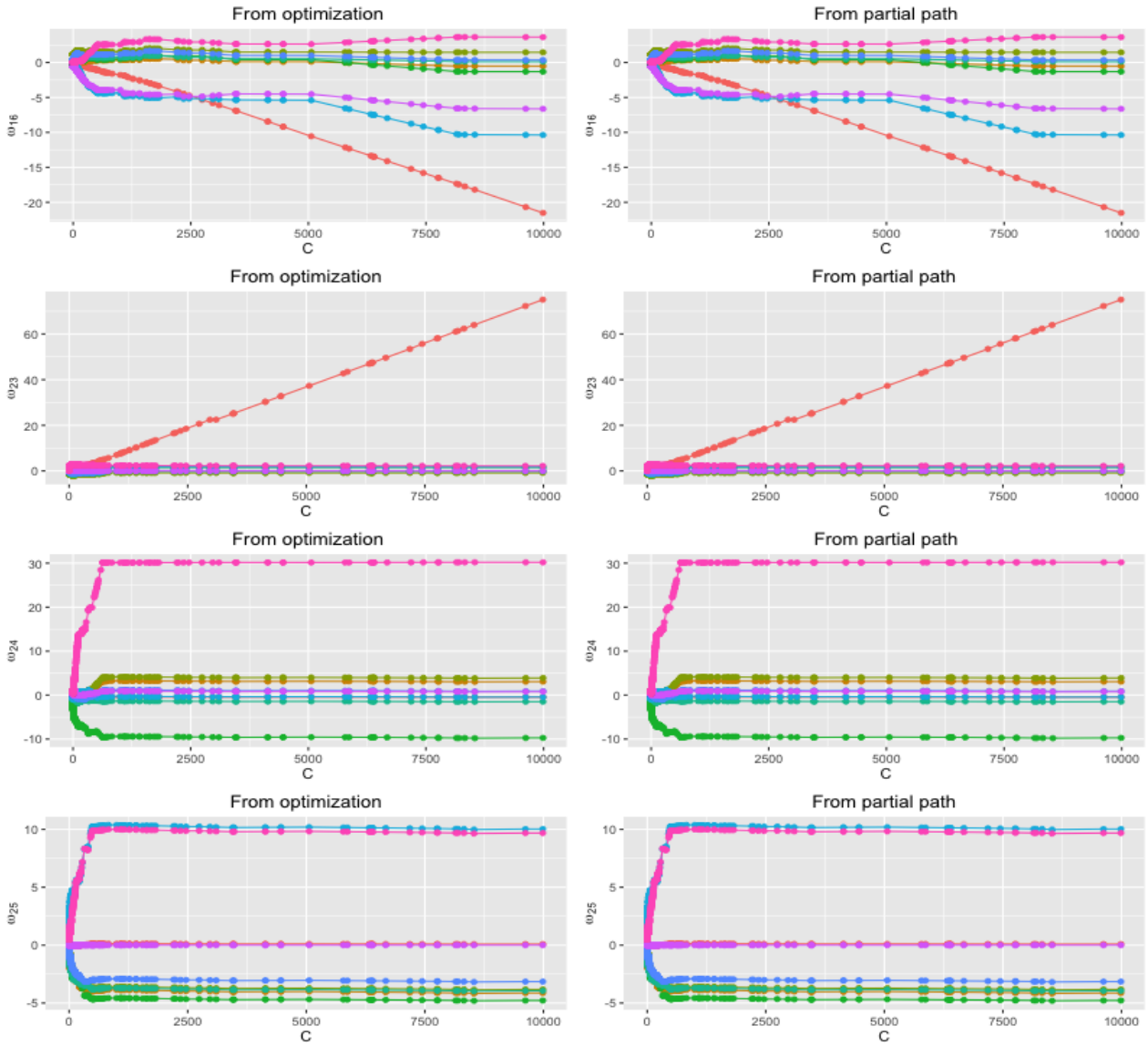


Figure 15. Comparing the solutions obtained from optimization and the partial path algorithm for GLASS data - Part 3

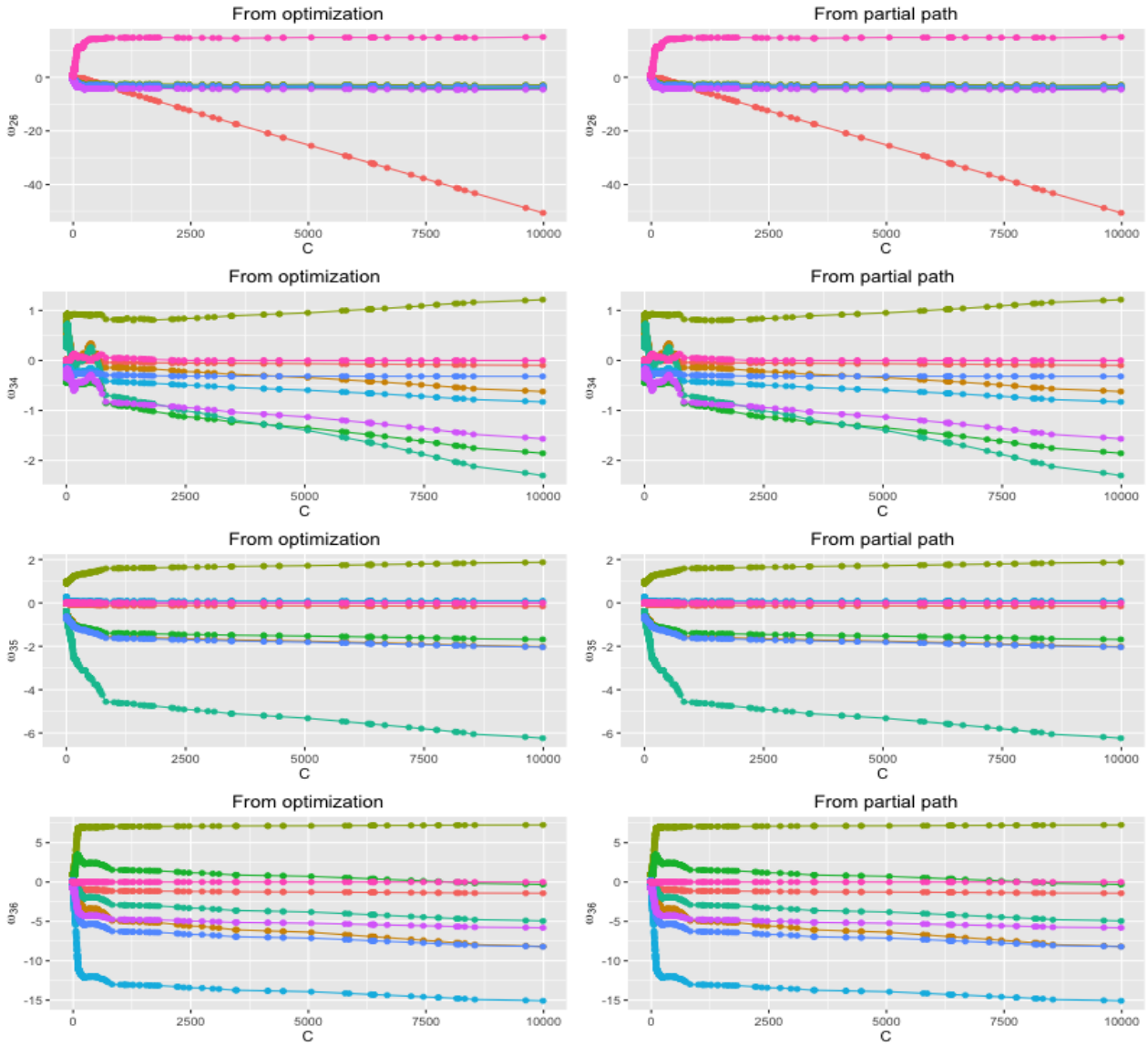


Figure 16. Comparing the solutions obtained from optimization and the partial path algorithm for GLASS data - Part 4