Homotopy continuation approaches for robust SV classification and regression

Shinya Suzumura¹ · Kohei Ogawa¹ ·
Masashi Sugiyama² · Masayuki Karasuyama¹ · Ichiro Takeuchi¹

Received: 30 June 2015 / Accepted: 9 January 2017 / Published online: 24 February 2017
© The Author(s) 2017

Abstract In support vector machine (SVM) applications with unreliable data that contains a portion of outliers, non-robustness of SVMs often causes considerable performance deterioration. Although many approaches for improving the robustness of SVMs have been studied, two major challenges remain. First, robust learning algorithms are essentially formulated as non-convex optimization problems because the loss function must be designed to alleviate the influence of outliers. It is thus important to develop a non-convex optimization method for robust SVM that can find a good local optimal solution. The second practical issue is how one can tune the hyper-parameter that controls the balance between robustness and efficiency. Unfortunately, due to the non-convexity, robust SVM solutions with slightly different hyper-parameter values can be significantly different, which makes model selection highly unstable. In this paper, we address these two issues simultaneously by introducing a novel *homotopy* approach to non-convex robust SVM learning. Our basic idea is to introduce parametrized formulations of robust SVM which bridge the standard SVM and fully robust SVM via the parameter that represents the influence of outliers. Our homotopy approach allows stable and efficient model selection based on the path of local optimal solutions. Empirical performances

---

Editor: Tapio Elomaa.

✉ Ichiro Takeuchi
takeuchi.ichiro@nitech.ac.jp

Shinya Suzumura
suzumura.mllab.nit@gmail.com

Kohei Ogawa
ogawa@gmail.com

Masashi Sugiyama
sugi@k.u-tokyo.ac.jp

Masayuki Karasuyama
karasuyama@nitech.ac.jp

¹ Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, Japan

² The University of Tokyo, Hongo, Bunkyo-ku, Tokyo, Japan

of the proposed approach are demonstrated through intensive numerical experiments both on robust classification and regression problems.

Keywords Robust support vector machine · Classification · Regression · Parametric programming · Non-convex optimization

1 Introduction

The *support vector machine* (SVM) has been one of the most successful machine learning algorithms (Boser et al. 1992; Cortes and Vapnik 1995; Vapnik 1998). However, in recent practical machine learning applications with less reliable data that contains outliers, non-robustness of the SVM often causes considerable performance deterioration. In robust learning context, outliers indicate observations that have been contaminated, inaccurately recorded, or drawn from different distributions from other normal observations.

Figure 1 shows examples of robust SV classification and regression. In classification problems, we regard the instances whose labels are flipped from the ground truth as outliers. As illustrated in Fig. 1a, standard SV classifier may be highly affected by outliers. On the other hand, robust SV classifier can alleviate the effects of outliers as illustrated in Fig. 1b. In regression problems, we regard instances whose output response are highly contaminated than other normal observations as outliers. Standard SV regression tends to produce an unstable result as illustrated in Fig. 1c, while robust SV regression can alleviate the effects of outliers as illustrated in Fig. 1d.

Fig. 1 Illustrative examples of **a** standard SVC (support vector classification) and **b** robust SVC, **c** standard SVR (support vector regression) and **d** robust SVR on toy datasets. In robust SVM, the classification and regression results are not sensitive to the two *red* outliers in the right-hand side of the plots (Color figure online)

Fig. 2 Robust loss functions for **a** SV classification and **b** SV regression. Note that these robust loss functions are essentially non-convex because they are designed to alleviate the effect of outliers. Furthermore, robust loss functions have additional hyper-parameter for controlling the trade-off between robustness and efficiency. For example, in Ramp loss function, one has to determine the breakpoint (the location of the orange circle) of the robust loss function. In plot (a), the breakpoint at margin -1.0 indicates that we regard the instances whose margin is smaller than -1.0 as outliers. Similarly, in plot (b), the breakpoints at ± 1.0 indicate that we regard the instances whose absolute residuals are greater than 1.0 as outliers (Color figure online)

1.1 Existing robust classification and regression methods

A great deal of efforts have been devoted to improve the robustness of the SVM and other similar learning algorithms (Masnadi-Shiraze and Vasconcelos 2000; Bartlett and Mendelson 2003; Shen et al. 2003; Krause and Singer 2004; Liu et al. 2005; Liu and Shen 2006; Xu et al. 2006; Collobert et al. 2006; Wu and Liu 2007; Masnadi-Shirazi and Vasconcelos 2009; Freund 2009; Yu et al. 2010). In the context of SV classification, so-called *Ramp* loss function (see Fig. 2a) is introduced for alleviating the effects of outliers. Similarly, robust loss function for SV regression (see Fig. 2b) is also introduced. Robustness properties obtained by replacing the loss functions to these robust ones have been intensively studied in the context of robust statistics.

As illustrated in Fig. 2, robust loss functions have two common properties. First, robust loss functions are essentially *non-convex* because they are designed to alleviate the effect of outliers.¹ If one uses a convex loss, even a single outlier can dominate the classification or regression result. Second, robust loss functions have an *additional parameter* for controlling the trade-off between robustness and efficiency. For example, in Ramp loss function in Fig. 2a, one must determine the breakpoint of the loss function (the location of the orange circle). Since such a tuning parameter governs the influence of outliers on the model, it must be carefully tuned based on the property of noise contained in the data set. Unless one has a clear knowledge about the properties of outliers, one has to tune the tuning parameter by using a model selection method such as cross-validation. These two properties suggest that one must solve many non-convex optimization problems for various values of the tuning parameter.

¹ Xu et al. (2006) introduced a robust learning approach that does not require non-convex optimization, in which non-convex loss functions are relaxed to be convex ones. In their approach, the resulting convex problems are formulated as semi-definite programs (SDPs), which are therefore inherently non-scalable. In our experience, the SDP method can be applied to datasets with up to a few hundred instances. Thus, we do not compare our method with the method in Xu et al. (2006) in this paper.

Unfortunately, existing non-convex optimization algorithms for robust SVM learning are not sufficiently efficient nor stable. The most common approach for solving non-convex optimization problems in robust SV classification and regression is *Difference of Convex (DC) programming* (Shen et al. 2003; Krause and Singer 2004; Liu et al. 2005; Liu and Shen 2006; Collobert et al. 2006; Wu and Liu 2007), or its special form called *Concave Convex Procedure (CCCP)* (Yuille and Rangarajan 2002). Since a solution obtained by these general non-convex optimization methods is only one of many local optimal solutions, one has to repeatedly solve the non-convex optimization problems from multiple different initial solutions. Furthermore, due to the non-convexity, robust SVM solutions with slightly different tuning parameter values can be significantly different, which makes the tuning-parameter selection problem highly unstable. To the best of our knowledge, there are no other existing studies that allows us to efficiently obtain stable sequence of multiple local optimal solutions for various tuning parameters values in the context of robust SV classification and regression.

1.2 Our contributions

In this paper, we address efficiency and stability issues of robust SV learning simultaneously by introducing a novel *homotopy* approach.² First, we introduce parameterized formulations of robust SVC and SVR which bridge the standard SVM and fully robust SVM via a parameter that governs the influence of outliers. Second, we use *homotopy* methods (Allgower and George 1993; Gal 1995; Ritter 1984; Best 1996) for tracing a path of solutions when the influence of outliers is gradually decreased. We call the parameter as the *homotopy parameter* and the path of solutions obtained by tracing the homotopy parameter as the *outlier path*. Figs. 3 and 4 illustrate how the robust loss functions for classification and regression problems can be gradually robustified, respectively.

Our first technical contribution is in analyzing the properties of the outlier path for both classification and regression problems. In particular, we derive the necessary and sufficient conditions for SVC and SVR solutions to be locally optimal (note that the well-known Karush-Kuhn Tucker (KKT) conditions are only necessary, but not sufficient). Interestingly, the analyses indicate that the outlier paths contain a finite number of discontinuous points. To the best of our knowledge, the above property of robust learning has not been known previously.

Our second contribution is to develop an efficient algorithm for actually computing the outlier path based on the above theoretical investigation of the geometry of robust SVM solutions. Here, we use parametric programming technique (Allgower and George 1993; Gal 1995; Ritter 1984; Best 1996), which is often used for computing the *regularization path* in machine learning literature. The main technical challenge here is how to handle the discontinuous points in the outlier path. We overcome this difficulty by precisely analyzing the necessary and sufficient conditions for the local optimality. We develop an algorithm that can precisely detect such discontinuous points, and *jump* to find a strictly better local optimal solution.

Experimental results indicate that our proposed method can find better robust SVM solutions more efficiently than alternative method based on CCCP. We conjecture that there are two reasons why favorable results can be obtained with our method. At first, the outlier path

² For regression problems, we study least absolute deviation (LAD) regression. It is straightforward to extend it to original SVR formulation with ε -insensitive loss function. In order to simplify the description, we often call LAD regression as SV regression (SVR). In what follows, we use the term SVM when we describe common properties of SVC and SVR.

Fig. 3 An illustration of the parametrized class of problems for robust SV classification (SVC). A loss function in each plot (red one) is characterized by two parameters $\theta \in [0, 1]$ and $s \in (-\infty, 0]$, which we call *robustness* parameters. When $\theta = 1$ or $s = -\infty$, the loss function is identical with the hinge loss function in standard SVM. On the other hand, when $\theta = 0$ and $s \neq -\infty$, the loss function is identical with the ramp loss function which has been used in several robust SV learning studies. In this paper, we consider a parametrized formulations of robust SVC which bridges the standard SVC and fully robust SVC via the homotopy parameters θ and/or s that governs the influence of outliers. The algorithm we introduce in this paper allows us to compute a path of solutions when we continuously change these homotopy parameters. In the *top row* and the *bottom row*, we consider paths of solutions with respect to the homotopy parameter θ and s , respectively. **a** Homotopy computation with decreasing θ from 1 to 0. **b** Homotopy computation with increasing s from $-\infty$ to 0 (Color figure online)

shares similar advantage as *simulated annealing* (Hromkovic 2001). Simulated annealing is known to find better local solutions in many non-convex optimization problems by solving a sequence of solutions along with so-called *temperature* parameter. If we regard the homotopy parameter as the temperature parameter, our outlier path algorithm can be interpreted as simulated annealing with infinitesimal step size (as we explain in the following sections, our algorithm represents the path of local solutions as a function of the homotopy parameter). Since our algorithm provides the path of local solutions, unlike other non-convex optimization algorithms such as CCCP, two solutions with slightly different homotopy parameter values tend to be similar, which makes the tuning parameter selection stable. According to our experiments, choice of the homotopy parameter is quite sensitive to the generalization performances. Thus, it is important to finely tune the homotopy parameter. Since our algorithm can compute the path of solutions, it is much more computationally efficient than running CCCP many times at different homotopy parameter values.

1.3 Organization of the paper

After we formulate robust SVC and SVR as parametrized optimization problems in Sect. 2, we derive in Sect. 3 the *necessary* and *sufficient* conditions for a robust SVM solution to be locally optimal, and show that there exist a finite number of discontinuous points in the local solution path. We then propose an efficient algorithm in Sect. 4 that can precisely detect such discontinuous points and *jump* to find a strictly better local optimal solution. In Sect. 5, we experimentally demonstrate that our proposed method, named the *outlier path algorithm*,

Fig. 4 An illustration of the parametrized class of problems for robust SV regression (SVR). A loss function in each plot (red one) is characterized by two parameters $\theta \in [0, 1]$ and $|s| \in [0, \infty)$, which we call *robustness* parameters. When $\theta = 1$ or $|s| = 0$, the loss function is identical with the loss function of least absolute deviation (LAD) regression. On the other hand, when $\theta = 0$ and $|s| \neq \infty$, the loss function is a robust function in which the influences of outliers are bounded. In this paper, we also consider a parametrized formulations of robust SVR which bridges the standard SVR and the robust SVR via the homotopy parameters θ and/or $|s|$ that governs the influence of outliers. The algorithm we introduce in this paper allows us to compute a path of solutions when we continuously change these homotopy parameters. In the *top row* and the *bottom row*, we consider paths of solutions with respect to the homotopy parameter θ and s , respectively. **a** Homotopy computation with decreasing θ from 1 to 0. **b** Homotopy computation with decreasing s from $-\infty$ to 0 (Color figure online)

outperforms the existing robust SVM algorithm based on CCCP or DC programming. Finally, we conclude in Sect. 6.

This paper is an extended version of our preliminary conference paper presented at ICML 2014 (Suzumura et al. 2014). In this paper, we have extended our robustification path framework to the regression problem, and many more experimental evaluations have been conducted. To the best of our knowledge, the homotopy method (Allgower and George 1993; Gal 1995; Ritter 1984; Best 1996) is first used in our preliminary conference paper in the context of robust learning. So far, homotopy-like methods have been (often implicitly) used for non-convex optimization problems in the context of sparse modeling (Zhang 2010; Mazumder et al. 2011; Zhou et al. 2012) and semi-supervised learning (Ogawa et al. 2013).

2 Parameterized formulation of robust SVM

In this section, we first formulate robust SVMs for classification and regression problems, which we denote by robust SVC (SV classification) and robust SVR (SV regression), respectively. Then, we use parameterized formulation both for robust SVC and SVR, where the parameter governs the influence of outliers to the model. The problem is reduced to ordinary non-robust SVM at one end of the parameter, while the problem corresponds to fully-robust SVM at the other end of the parameter. In the following sections, we develop algorithms for computing the path of local optimal solutions when the parameter is changed from one end to the other.

2.1 Robust SV classification

Let us consider a binary classification problem with n instances and d features. We denote the training set as $\{(\mathbf{x}_i, y_i)\}_{i \in \mathbb{N}_n}$ where $\mathbf{x}_i \in \mathcal{X}$ is the input vector in the input space $\mathcal{X} \subset \mathbb{R}^d$, $y_i \in \{-1, 1\}$ is the binary class label, and the notation $\mathbb{N}_n := \{1, \dots, n\}$ represents the set of natural numbers up to n . We write the decision function as

$$f(\mathbf{x}) := \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \quad (1)$$

where $\boldsymbol{\phi}$ is the feature map implicitly defined by a kernel $\mathbf{K}$, $\mathbf{w}$ is a vector in the feature space, and $^\top$ denotes the transpose of vectors and matrices.

We introduce the following class of optimization problems *parameterized* by θ and s :

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \ell(y_i f(\mathbf{x}_i); \theta, s), \quad (2)$$

where $C > 0$ is the regularization parameter which controls the balance between the first regularization term and the second loss term. The loss function ℓ is characterized by a pair of parameters $\theta \in [0, 1]$ and $s \leq 0$ as

$$\ell(z; \theta, s) := \begin{cases} [0, 1 - z]_+, & z \geq s, \\ 1 - \theta z - s, & z < s, \end{cases} \quad (3)$$

where $[z]_+ := \max\{0, z\}$. We refer to θ and s as the *homotopy parameters*. Figure 3 shows the loss functions for several θ and s . Note that when $\theta = 1$ or $s = \infty$, the optimization problem (2) is equivalent to the standard formulation of SVC, in particular, so-called *soft-margin* SVC. In this case, the loss function is reduced to the well-known *hinge loss* function, which linearly penalizes $1 - y_i f(\mathbf{x}_i)$ when $y_i f(\mathbf{x}_i) \leq 1$, and gives no penalty for $y_i f(\mathbf{x}_i) > 1$ (the left-most column in Fig. 3). The input vector $\mathbf{x}_i$ having $y_i f(\mathbf{x}_i) \leq 1$ at the final solution is called a support vector by which the resulting decision boundary is defined, while the other input vectors do not effect on the boundary because they have no effect on the objective function. The first homotopy parameter θ can be interpreted as the *weight* for an outlier: $\theta = 1$ indicates that the influence of an outlier is the same as an inlier, while $\theta = 0$ indicates that outliers are completely ignored. The second homotopy parameter $s \leq 0$ can be interpreted as the threshold for deciding outliers and inliers. When $\theta = 0$ and $s = 0$, the loss function is reduced to the *ramp loss* function (the right-most column in Fig. 3) to which we have referred as fully robust SVM. A variety of robust loss functions, including the functions that we employed here, have appeared in Bartlett and Mendelson (2003).

In the following sections, we consider two types of homotopy methods. In the first method, we fix $s = 0$, and gradually change θ from 1 to 0 (see the top five plots in Fig. 3). In the second method, we fix $\theta = 0$ and gradually change s from $-\infty$ to 0 (see the bottom five plots in Fig. 3). The optimization 2 is a *convex* problem when $\theta = 1$ or $s = -\infty$, in which ℓ is the hinge loss, while it is *non-convex* when $\theta = 0$ and $s = 0$, in which ℓ is the ramp loss. Therefore, each of the above two homotopy methods can be interpreted as the process of tracing a sequence of solutions when the optimization problem is gradually modified from convex to non-convex. By doing so, we expect to find good local optimal solutions because such a process can be interpreted as *simulated annealing* (Hromkovic 2001). In addition, we can adaptively control the degree of robustness by selecting the best θ or s based on some model selection scheme.

2.2 Robust SV regression

Let us next consider a regression problem. We denote the training set of the regression problem as $\{(\mathbf{x}_i, y_i)\}_{i \in \mathbb{N}_n}$, where the input $\mathbf{x}_i \in \mathcal{X}$ is the input vector as the classification case, while the output $y_i \in \mathbb{R}$ is a real scalar. We consider a regression function $f(\mathbf{x})$ in the form of (1). SV regression is formulated as

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \ell(y_i - f(\mathbf{x}_i); \theta, s), \quad (4)$$

where $C > 0$ is the regularization parameter, and the loss function ℓ is defined as

$$\ell(z; \theta, s) := \begin{cases} |z|, & |z| < s, \\ (|z| - s)\theta + s, & |z| \geq s. \end{cases} \quad (5)$$

The loss function in (5) has two parameters $\theta \in [0, 1]$ and $s \in [0, \infty)$ as the classification case. Figure 4 shows the loss functions for several θ and s .

3 Local optimality

In order to use the homotopy approach, we need to clarify the continuity of the local solution path. To this end, we investigate several properties of local solutions of robust SVM, and derive the necessary and sufficient conditions. Interestingly, our analysis reveals that the local solution path has a finite number of *discontinuous* points. The theoretical results presented here form the basis of our novel homotopy algorithm given in the next section that can properly handle the above discontinuity issue. We first discuss the local optimality of robust SVC in detail in Sects. 3.1 and 3.2, and then present the corresponding result of robust SVR briefly in Sect. 3.3. Here we call a solution to be locally optimal if there is no strictly better feasible solutions in its neighborhood.

3.1 Conditionally optimal solutions (for robust SVC)

The basic idea of our theoretical analysis is to reformulate the robust SVC learning problem as a combinatorial optimization problem. We consider a partition of the instances $\mathbb{N}_n := \{1, \dots, n\}$ into two disjoint sets $\mathcal{I}$ and $\mathcal{O}$. The instances in $\mathcal{I}$ and $\mathcal{O}$ are defined as *Inliers* and *Outliers*, respectively. Here, we restrict that the margin $y_i f(\mathbf{x}_i)$ of an inlier should be larger than or equal to s , while that of an outlier should be smaller than or equal to s . We denote the partition as $\mathcal{P} := \{\mathcal{I}, \mathcal{O}\} \in 2^{\mathbb{N}_n}$, where $2^{\mathbb{N}_n}$ is the power set⁴ of $\mathbb{N}_n$. Given a partition $\mathcal{P}$, the above restrictions define the feasible region of the solution f in the form of a convex polytope:

$$\text{pol}(\mathcal{P}; s) := \left\{ f \mid \begin{array}{l} y_i f(\mathbf{x}_i) \geq s, \ i \in \mathcal{I} \\ y_i f(\mathbf{x}_i) \leq s, \ i \in \mathcal{O} \end{array} \right\}. \quad (6)$$

Using the notion of the convex polytopes, the optimization problem (2) can be rewritten as

$$\min_{\mathcal{P} \in 2^{\mathbb{N}_n}} \left(\min_{f \in \text{pol}(\mathcal{P}; s)} J_{\mathcal{P}}(f; \theta) \right), \quad (7)$$

³ Note that an instance with the margin $y_i f(\mathbf{x}_i) = s$ can be the member of either $\mathcal{I}$ or $\mathcal{O}$.

⁴ The power set means that there are 2^n patterns that each of the instances belongs to either $\mathcal{I}$ or $\mathcal{O}$.

where the objective function $J_{\mathcal{P}}$ is defined as⁵

$$J_{\mathcal{P}}(f; \theta) := \frac{1}{2} \|\mathbf{w}\|_2^2 + C \left(\sum_{i \in \mathcal{I}} [1 - y_i f(\mathbf{x}_i)]_+ + \theta \sum_{i \in \mathcal{O}} [1 - y_i f(\mathbf{x}_i)]_+ \right).$$

Note that the right hand side depends on $\mathcal{P}$ though $\mathcal{I}$ and $\mathcal{O}$.

When the partition $\mathcal{P}$ is fixed, it is easy to confirm that the inner minimization problem of (7) is a convex problem.

Definition 1 (Conditionally optimal solutions) *Given a partition $\mathcal{P}$, the solution of the following convex problem is said to be the conditionally optimal solution:*

$$f_{\mathcal{P}}^* := \operatorname{argmin}_{f \in \operatorname{pol}(\mathcal{P}; s)} J_{\mathcal{P}}(f; \theta). \quad (8)$$

The formulation in (7) is interpreted as a combinatorial optimization problem of finding the best solution from all the 2^n conditionally optimal solutions $f_{\mathcal{P}}^*$ corresponding to all possible 2^n partitions.⁶

Using the representer theorem or convex optimization theory, we can show that any conditionally optimal solution can be written as

$$f_{\mathcal{P}}^*(\mathbf{x}) := \sum_{j \in \mathbb{N}_n} \alpha_j^* y_j K(\mathbf{x}, \mathbf{x}_j), \quad (9)$$

where $\{\alpha_j^*\}_{j \in \mathbb{N}_n}$ are the optimal Lagrange multipliers. The following lemma summarizes the KKT optimality conditions of the conditionally optimal solution $f_{\mathcal{P}}^*$.

Lemma 2 *The KKT conditions of the convex problem (8) is written as*

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) > 1 \Rightarrow \alpha_i^* = 0, \quad (10a)$$

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = 1 \Rightarrow \alpha_i^* \in [0, C], \quad (10b)$$

$$s < y_i f_{\mathcal{P}}^*(\mathbf{x}_i) < 1 \Rightarrow \alpha_i^* = C, \quad (10c)$$

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = s, i \in \mathcal{I} \Rightarrow \alpha_i^* \geq C, \quad (10d)$$

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = s, i \in \mathcal{O} \Rightarrow \alpha_i^* \leq C\theta, \quad (10e)$$

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) < s \Rightarrow \alpha_i^* = C\theta. \quad (10f)$$

The proof is presented in Appendix 7.

3.2 The necessary and sufficient conditions for local optimality (for Robust SVC)

From the definition of conditionally optimal solutions, it is clear that a local optimal solution must be conditionally optimal within the convex polytope $\operatorname{pol}(\mathcal{P}; s)$. However, the conditional optimality does not necessarily indicate the local optimality as the following theorem suggests.

Theorem 3 *For any $\theta \in [0, 1)$ and $s \leq 0$, consider the situation where a conditionally optimal solution $f_{\mathcal{P}}^*$ is at the boundary of the convex polytope $\operatorname{pol}(\mathcal{P}; s)$, i.e., there exists at least an instance such that $y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = s$. In this situation, if we define a new partition*

⁵ Note that we omitted the constant terms irrelevant to the optimization problem.

⁶ For some partitions $\mathcal{P}$, the convex problem (8) might not have any feasible solutions.

$\tilde{\mathcal{P}} := \{\tilde{\mathcal{I}}, \tilde{\mathcal{O}}\}$ as

$$\tilde{\mathcal{I}} \leftarrow \mathcal{I} \setminus \{i \in \mathcal{I} | y_i f^*(\mathbf{x}_i) = s\} \cup \{i \in \mathcal{O} | y_i f^*(\mathbf{x}_i) = s\}, \quad (11a)$$

$$\tilde{\mathcal{O}} \leftarrow \mathcal{O} \setminus \{i \in \mathcal{O} | y_i f^*(\mathbf{x}_i) = s\} \cup \{i \in \mathcal{I} | y_i f^*(\mathbf{x}_i) = s\}, \quad (11b)$$

then the new conditionally optimal solution $f_{\tilde{\mathcal{P}}}^*$ is strictly better than the original conditionally optimal solution $f_{\mathcal{P}}^*$, i.e.,

$$J_{\tilde{\mathcal{P}}}(f_{\tilde{\mathcal{P}}}^*; \theta) < J_{\mathcal{P}}(f_{\mathcal{P}}^*; \theta). \quad (12)$$

The proof is presented in Appendix 8. Theorem 3 indicates that if $f_{\mathcal{P}}^*$ is at the boundary of the convex polytope $\text{pol}(\mathcal{P}; s)$, i.e., if there is one or more instances such that $y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = s$, then $f_{\mathcal{P}}^*$ is NOT locally optimal because there is a strictly better solution in the opposite side of the boundary.

The following theorem summarizes the necessary and sufficient conditions for local optimality. Note that, in non-convex optimization problems, the KKT conditions are necessary but not sufficient in general.

Theorem 4 For $\theta \in [0, 1)$ and $s \leq 0$,

$$y_i f^*(\mathbf{x}_i) > 1 \Rightarrow \alpha_i^* = 0, \quad (13a)$$

$$y_i f^*(\mathbf{x}_i) = 1 \Rightarrow \alpha_i^* \in [0, C], \quad (13b)$$

$$s < y_i f^*(\mathbf{x}_i) < 1 \Rightarrow \alpha_i^* = C, \quad (13c)$$

$$y_i f^*(\mathbf{x}_i) < s \Rightarrow \alpha_i^* = C\theta, \quad (13d)$$

$$y_i f^*(\mathbf{x}_i) \neq s, \forall i \in \mathbb{N}_n, \quad (13e)$$

are necessary and sufficient for f^* to be locally optimal.

The proof is presented in Appendix 9. The condition (13e) indicates that the solution at the boundary of the convex polytope is not locally optimal. Figure 5 illustrates when a conditionally optimal solution can be locally optimal with a certain θ or s .

Theorem 4 suggests that, whenever the local solution path computed by the homotopy approach encounters a boundary of the current convex polytope at a certain θ or s , the solution is not anymore locally optimal. In such cases, it is better to search a local optimal solution at that θ or s , and restart the local solution path from the new one. In other words, the local solution path has *discontinuity* at that θ or s . Fortunately, Theorem 3 tells us how to handle such a situation. If the local solution path arrives at the boundary, it can *jump* to the new conditionally optimal solution $f_{\tilde{\mathcal{P}}}^*$ which is located on the opposite side of the boundary. This jump operation is justified because the new solution is shown to be strictly better than the previous one. Figure 5c, d illustrate such a situation.

3.3 Local optimality of SV regression

In order to derive the necessary and sufficient conditions of the local optimality in robust SVR, with abuse of notation, let us consider a partition of the instances $\mathbb{N}_n$ into two disjoint sets $\mathcal{I}$ and $\mathcal{O}$, which represent inliers and outliers, respectively. In regression problems, an instance $(\mathbf{x}_i, y_i)$ is regarded as an outlier if the absolute residual $|y_i - f(\mathbf{x}_i)|$ is sufficiently large. Thus, we define inliers and outliers of regression problem as

$$\mathcal{I} := \{i \in \mathbb{N}_n | |y_i - f(\mathbf{x}_i)| \leq s\},$$

$$\mathcal{O} := \{i \in \mathbb{N}_n | |y_i - f(\mathbf{x}_i)| \geq s\}.$$

Fig. 5 Solution space of robust SVC. **a** The arrows indicate a local solution path when θ is gradually moved from θ_1 to θ_5 (see Sect. 4 for more details). **b** $f_{\mathcal{P}}^*$ is locally optimal if it is at the strict interior of the convex polytope $\text{pol}(\mathcal{P}; s)$. **c** If $f_{\mathcal{P}}^*$ exists at the boundary, then $f_{\mathcal{P}}^*$ is feasible, but not locally optimal. A new convex polytope $\text{pol}(\tilde{\mathcal{P}}; s)$ defined in the opposite side of the boundary is shown in yellow. **d** A strictly better solution exists in $\text{pol}(\tilde{\mathcal{P}}; s)$ (Color figure online)

Given a partition $\mathcal{P} := \{\mathcal{I}, \mathcal{O}\} \in 2^{\mathbb{N}_n}$, the feasible region of the solution f is represented as a convex polytope:

$$\text{pol}(\mathcal{P}; s) := \left\{ f \mid \begin{array}{l} |y_i - f(\mathbf{x}_i)| \leq s, i \in \mathcal{I}, \\ |y_i - f(\mathbf{x}_i)| \geq s, i \in \mathcal{O} \end{array} \right\}. \quad (14)$$

Then, as in the classification case, the optimization problem (4) can be rewritten as

$$\min_{\mathcal{P}} \left(\min_{f \in \text{pol}(\mathcal{P}; s)} J_{\mathcal{P}}(f; \theta) \right), \quad (15)$$

where the objective function $J_{\mathcal{P}}$ is defined as

$$J_{\mathcal{P}}(f; \theta) := \frac{1}{2} \|\mathbf{w}\|_2^2 + C \left(\sum_{i \in \mathcal{I}} |y_i - f(\mathbf{x}_i)| + \theta \sum_{i \in \mathcal{O}} |y_i - f(\mathbf{x}_i)| \right).$$

Since the inner problem of (15) is a convex problem, any conditionally optimal solution can be written as

$$f_{\mathcal{P}}^*(\mathbf{x}) := \sum_{j \in \mathbb{N}_n} \alpha_j^* K(\mathbf{x}, \mathbf{x}_j). \quad (16)$$

The KKT conditions of $f_{\mathcal{P}}^*(\mathbf{x})$ are written as

$$|y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| = 0 \Rightarrow 0 \leq |\alpha_i^*| \leq C, \quad (17a)$$

$$0 \leq |y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| < s \Rightarrow |\alpha_i^*| = C, \quad (17b)$$

$$|y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| = s, i \in \mathcal{I} \Rightarrow |\alpha_i^*| \geq C, \quad (17c)$$

$$|y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| = s, i \in \mathcal{O} \Rightarrow |\alpha_i^*| \leq \theta C, \quad (17d)$$

$$|y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| > s \Rightarrow |\alpha_i^*| = \theta C. \quad (17e)$$

Based on the same discussion as Sect. 3.2, the necessary and sufficient conditions for the local optimality of robust SVR are summarized as the following theorem:

Theorem 5 For $\theta \in [0, 1)$ and $s \geq 0$,

$$|y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| = 0 \Rightarrow 0 \leq |\alpha_i^*| \leq C, \quad (18a)$$

$$0 \leq |y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| < s \Rightarrow |\alpha_i^*| = C, \quad (18b)$$

$$|y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| > s \Rightarrow |\alpha_i^*| = \theta C, \quad (18c)$$

$$|y_i - f_{\mathcal{P}}^*(\mathbf{x}_i)| \neq s. \quad (18d)$$

are necessary and sufficient for f^* to be locally optimal.

We omit the proof of this theorem because they can be easily derived in the same way as Theorem 4.

4 Outlier path algorithm

Based on the analysis presented in the previous section, we develop a novel homotopy algorithm for robust SVM. We call the proposed method the *outlier-path (OP)* algorithm. For simplicity, we consider homotopy path computation involving either θ or s , and denote the former as OP- θ and the latter as OP- s . OP- θ computes the local solution path when θ is gradually decreased from 1 to 0 with fixed $s = 0$, while OP- s computes the local solution path when s is gradually increased from $-\infty$ to 0 with fixed $\theta = 0$.

The local optimality of robust SVM in the previous section shows that the path of local optimal solutions has finite discontinuous points that satisfy (13e) or (18). Below, we introduce an algorithm that appropriately handles those discontinuous points. In this section, we only describe the algorithm for robust SVC. All the methodologies described in this section can be easily extended to robust SVR counterpart.

4.1 Overview

The main flow of the OP algorithm is described in Algorithm 1. The solution f is initialized by solving the standard (convex) SVM, and the partition $\mathcal{P} := \{\mathcal{I}, \mathcal{O}\}$ is defined to satisfy the constraints in (6). The algorithm mainly switches over the two steps called the *continuous step (C-step)* and the *discontinuous step (D-step)*.

In the C-step (Algorithm 2), a continuous path of local solutions is computed for a sequence of gradually decreasing θ (or increasing s) within the convex polytope $\text{pol}(\mathcal{P}; s)$ defined by the current partition $\mathcal{P}$. If the local solution path encounters a boundary of the convex polytope, i.e., if there exists at least an instance such that $y_i f(\mathbf{x}_i) = s$, then the algorithm stops updating θ (or s) and enters the D-step.

Algorithm 1 Outlier Path Algorithm

- 1: Initialize the solution f by solving the standard SVM.
- 2: Initialize the partition $\mathcal{P} := \{\mathcal{I}, \mathcal{O}\}$ as follows:

$$\begin{aligned}\mathcal{I} &\leftarrow \{i \in \mathbb{N}_n | y_i f(\mathbf{x}_i) \leq s\}, \\ \mathcal{O} &\leftarrow \{i \in \mathbb{N}_n | y_i f(\mathbf{x}_i) > s\}.\end{aligned}$$

- 3: $\theta \leftarrow 1$ for OP- θ ; $s \leftarrow \min_{i \in \mathbb{N}_n} y_i f(\mathbf{x}_i)$ for OP- s .
- 4: **while** $\theta > 0$ for OP- θ ; $s < 0$ for OP- s **do**
- 5:   **if** $(y_i f(\mathbf{x}_i) \neq s \ \forall i \in \mathbb{N}_n)$ **then**
- 6:   Run C-step.
- 7:   **else**
- 8:   Run D-step.
- 9:   **end if**
- 10: **end while**

Algorithm 2 Continuous Step (C-step)

- 1: **while** $(y_i f(\mathbf{x}_i) \neq s \ \forall i \in \mathbb{N}_n)$ **do**
- 2:   Solve the sequence of convex problems,

$$\min_{f \in \text{pol}(\mathcal{P}; s)} J_{\mathcal{P}}(f; \theta),$$

for gradually decreasing θ in OP- θ or gradually increasing s in OP- s .

- 3: **end while**

In the D-step (Algorithm 3), a better local solution is obtained for fixed θ (or s) by solving a convex problem defined over another convex polytope in the opposite side of the boundary (see Fig. 5d). If the new solution is again at a boundary of the new polytope, the algorithm repeatedly calls the D-step until it finds the solution in the strict interior of the current polytope.

The C-step can be implemented by any homotopy algorithms for solving a sequence of quadratic problems (QP). In OP- θ , the local solution path can be exactly computed because the path within a convex polytope can be represented as piecewise-linear functions of the homotopy parameter θ . In OP- s , the C-step is trivial because the optimal solution is shown to be constant within a convex polytope. In Sects. 4.2 and 4.3, we will describe the details of our implementation of the C-step for OP- θ and OP- s , respectively.

In the D-step, we only need to solve a single quadratic problem (QP). Any QP solver can be used in this step. We note that the *warm-start* approach (DeCoste and Wagstaff 2000) is quite helpful in the D-step because the difference between two conditionally optimal solutions in adjacent two convex polytopes is typically very small. In Sect. 4.4, we describe the details of our implementation of the D-step. Figure 6 illustrates an example of the local solution path obtained by OP- θ .

4.2 Continuous-step for OP- θ

In the C-step, the partition $\mathcal{P} := \{\mathcal{I}, \mathcal{O}\}$ is fixed, and our task is to solve a sequence of convex quadratic problems (QPs) parameterized by θ within the convex polytope $\text{pol}(\mathcal{P}; s)$. It has been known in optimization literature that a certain class of parametric convex QP can be exactly solved by exploiting the piecewise linearity of the solution path (Best 1996). We can easily show that the local solution path of OP- θ within a convex polytope is also

Fig. 6 An example of the local solution path by OP- θ on a simple toy data set (with $C = 200$). The paths of four Lagrange multipliers $\alpha_1^*, \dots, \alpha_4^*$ are plotted in the range of $\theta \in [0, 1]$. Open circles represent the discontinuous points in the path. In this simple example, we had experienced three discontinuous points at $\theta = 0.37, 0.67$ and 0.77

Algorithm 3 Discontinuous Step (D-step)

1: Update the partition $\mathcal{P} := \{\mathcal{I}, \mathcal{O}\}$ as follows:

$$\begin{aligned}\mathcal{I} &\leftarrow \mathcal{I} \setminus \{i \in \mathcal{I} | y_i f(\mathbf{x}_i) = s\} \cup \{i \in \mathcal{O} | y_i f(\mathbf{x}_i) = s\}, \\ \mathcal{O} &\leftarrow \mathcal{O} \setminus \{i \in \mathcal{O} | y_i f(\mathbf{x}_i) = s\} \cup \{i \in \mathcal{I} | y_i f(\mathbf{x}_i) = s\}.\end{aligned}$$

2: Solve the following convex problem for fixed θ and s :

$$\min_{f \in \text{pol}(\mathcal{P}; s)} J_{\mathcal{P}}(f; \theta).$$

represented as a piecewise-linear function of θ . The algorithm presented here is similar to the regularization path algorithm for SVM given in [Hastie et al. \(2004\)](#).

Let us consider a partition of the inliers in $\mathcal{I}$ into the following three disjoint sets:

$$\begin{aligned}\mathcal{R} &:= \{i | 1 < y_i f(\mathbf{x}_i)\}, \\ \mathcal{E} &:= \{i | y_i f(\mathbf{x}_i) = 1\}, \\ \mathcal{L} &:= \{i | s < y_i f(\mathbf{x}_i) < 1\}.\end{aligned}$$

For a given fixed partition $\{\mathcal{R}, \mathcal{E}, \mathcal{L}, \mathcal{O}\}$, the KKT conditions of the convex problem (8) indicate that

$$\alpha_i = 0 \quad \forall i \in \mathcal{R}, \quad \alpha_i = C \quad \forall i \in \mathcal{L}, \quad \alpha_i = C\theta \quad \forall i \in \mathcal{O}.$$

The KKT conditions also imply that the remaining Lagrange multipliers $\{\alpha_i\}_{i \in \mathcal{E}}$ must satisfy the following linear system of equations:

$$\begin{aligned}y_i f(\mathbf{x}_i) &= \sum_{j \in \mathbb{N}_n} \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) = 1 \quad \forall i \in \mathcal{E} \\ \Leftrightarrow \mathbf{Q}_{\mathcal{E}\mathcal{E}} \boldsymbol{\alpha}_{\mathcal{E}} &= \mathbf{1} - \mathbf{Q}_{\mathcal{E}\mathcal{L}} \mathbf{1}C - \mathbf{Q}_{\mathcal{E}\mathcal{O}} \mathbf{1}C\theta,\end{aligned}\tag{19}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is an $n \times n$ matrix whose $(i, j)^{\text{th}}$ entry is defined as $Q_{ij} := y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. Here, a notation such as $\mathbf{Q}_{\mathcal{E}\mathcal{L}}$ represents a submatrix of $\mathbf{Q}$ having only the rows in the index set $\mathcal{E}$ and the columns in the index set $\mathcal{L}$. By solving the linear system of equations (19), the Lagrange multipliers $\alpha_i, i \in \mathbb{N}_n$, can be written as an affine function of θ .

Noting that $y_i f(\mathbf{x}_i) = \sum_{j \in \mathbb{N}_n} \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ is also represented as an affine function of θ , any changes of the partition $\{\mathcal{R}, \mathcal{E}, \mathcal{L}\}$ can be exactly identified when the homotopy parameter θ is continuously decreased. Since the solution path linearly changes for each partition of $\{\mathcal{R}, \mathcal{E}, \mathcal{L}\}$, the entire path is represented as a continuous piecewise-linear function of the homotopy parameter θ . We denote the points in $\theta \in [0, 1)$ at which members of the sets $\{\mathcal{R}, \mathcal{E}, \mathcal{L}\}$ change as *break-points* θ_{BP} .

Using the piecewise-linearity of $y_i f(\mathbf{x}_i)$, we can also identify when we should switch to the D-step. Once we detect an instance satisfying $y_i f(\mathbf{x}_i) = s$, we exit the C-step and enter the D-step.

4.3 Continuous-step for OP-s

Since θ is fixed to 0 in OP-s, the KKT conditions (10) yields

$$\alpha_i = 0 \quad \forall i \in \mathcal{O}.$$

This means that outliers have no influence on the solution and thus the conditionally optimal solution $f_{\mathcal{P}}^*$ does not change with s as long as the partition $\mathcal{P}$ is unchanged. The only task in the C-step for OP-s is therefore to find the next s that changes the partition $\mathcal{P}$. Such s can be simply found as

$$s \leftarrow \min_{i \in \mathcal{L}} y_i f(\mathbf{x}_i).$$

4.4 Discontinuous-step (for both OP- θ and OP-s)

As mentioned before, any convex QP solver can be used for the D-step. When the algorithm enters the D-step, we have the conditionally optimal solution $f_{\mathcal{P}}^*$ for the partition $\mathcal{P} := \{\mathcal{I}, \mathcal{O}\}$. Our task here is to find another conditionally optimal solution $f_{\tilde{\mathcal{P}}}^*$ for $\tilde{\mathcal{P}} := \{\tilde{\mathcal{I}}, \tilde{\mathcal{O}}\}$ given by (11).

Given that the difference between the two solutions $f_{\mathcal{P}}^*$ and $f_{\tilde{\mathcal{P}}}^*$ is typically small, the D-step can be efficiently implemented by a technique used in the context of incremental learning (Cauwenberghs and Poggio 2001).

Let us define

$$\begin{aligned} \Delta_{\mathcal{I} \rightarrow \mathcal{O}} &:= \{i \in \mathcal{I} \mid y_i f_{\mathcal{P}}(\mathbf{x}_i) = s\}, \\ \Delta_{\mathcal{O} \rightarrow \mathcal{I}} &:= \{i \in \mathcal{O} \mid y_i f_{\mathcal{P}}(\mathbf{x}_i) = s\}. \end{aligned}$$

Then, we consider the following parameterized problem with parameter $\mu \in [0, 1]$:

$$f_{\tilde{\mathcal{P}}}(\mathbf{x}_i; \mu) := f_{\mathcal{P}}(\mathbf{x}_i) + \mu \Delta f_i \quad \forall i \in \mathbb{N}_n,$$

where

$$\Delta f_i := y_i \begin{bmatrix} \mathbf{K}_{i, \Delta_{\mathcal{I} \rightarrow \mathcal{O}}} & \mathbf{K}_{i, \Delta_{\mathcal{O} \rightarrow \mathcal{I}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{\Delta_{\mathcal{I} \rightarrow \mathcal{O}}}^{(\text{bef})} - \mathbf{1}C\theta \\ \boldsymbol{\alpha}_{\Delta_{\mathcal{O} \rightarrow \mathcal{I}}}^{(\text{bef})} - \mathbf{1}C \end{bmatrix},$$

and $\boldsymbol{\alpha}^{(\text{bef})}$ be the corresponding $\boldsymbol{\alpha}$ at the beginning of the D-Step. We can show that $f_{\tilde{\mathcal{P}}}(\mathbf{x}_i; \mu)$ is reduced to $f_{\mathcal{P}}(\mathbf{x}_i)$ when $\mu = 1$, while it is reduced to $f_{\tilde{\mathcal{P}}}(\mathbf{x}_i)$ when $\mu = 0$ for all $i \in \mathbb{N}_n$. By using a similar technique to incremental learning (Cauwenberghs and Poggio 2001), we can efficiently compute the path of solutions when μ is continuously changed from 1 to 0. This algorithm behaves similarly to the C-step in OP- θ . The implementation detail of the D-step is described in Appendix 10.

5 Numerical experiments

In this section, we compare the proposed outlier-path (OP) algorithm with conventional concave-convex procedure (CCCP) (Yuille and Rangarajan 2002) because, in most of the

Table 1 Benchmark data sets for robust SVC experiments

	Data	n	d
D1	BreastCancerDiagnostic	569	30
D2	AustralianCredit	690	14
D3	GermanNumer	1000	24
D4	SVMGuide1	3089	4
D5	Spambase	4601	57
D6	Musk	6598	166
D7	Gisette	6000	5000
D8	w5a	9888	300
D9	a6a	11220	122
D10	a7a	16100	122

n = # of instances, d = input dimension

Table 2 Benchmark data sets for robust SVR experiments

	Data	n	d
D1	Bodyfat	252	14
D2	yacht_hydrodynamics	308	6
D3	mpg	392	7
D4	Housing	506	13
D5	mg	1385	6
D6	Winequality-red	1599	11
D7	Winequality-white	4898	11
D8	space_ga	3107	6
D9	Abalone	4177	8
D10	cpusmall	8192	12
D11	cadata	20640	8

n = # of instances, d = input dimension

existing robust SVM studies, non-convex optimization for robust SVM training are solved by CCCP or a variant called difference of convex (DC) programming (Shen et al. 2003; Krause and Singer 2004; Liu et al. 2005; Liu and Shen 2006; Collobert et al. 2006; Wu and Liu 2007).

5.1 Setup

We used several benchmark data sets listed in Tables 1 and 2. We randomly divided data set into training (40%), validation (30%), and test (30%) sets for the purposes of optimization, model selection (including the selection of θ or s), and performance evaluation, respectively. For robust SVC, we randomly flipped 15, 20, 25% of the labels in the training and the validation data sets. For robust SVR, we first preprocess the input and output variables; each input variable was normalized so that the minimum and the maximum values are -1 and $+1$, respectively, while the output variable was standardized to have mean zero and variance one. Then, for the 5, 10, 15% of the training and the validation instances, we added a uniform noise $U(-2, 2)$ to input variable, and a Gaussian noise $N(0, 10^2)$ to output variable, where $U(a, b)$ denotes the uniform distribution between a and b and $N(\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2 .

5.2 Generalization performance

First, we compared the generalization performance. We used the linear kernel and the radial basis function (RBF) kernel defined as $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, where γ is a kernel parameter fixed to $\gamma = 1/d$ with d being the input dimensionality. Model selection was carried out by finding the best hyper-parameter combination that minimizes the validation error. We have a pair of hyper-parameters in each setup. In all the setups, the regularization

Table 3 The mean of test error by 0–1 loss and standard deviation (linear, robust SVC)

Data	C-SVM	CCCP- θ	OP- θ	CCCP- s	OP- s
D1	.056(.016)	.050(.014)	.049(.016)	.055(.018)	.050(.016)
D2	.151(.018)	.145(.007)	.151(.018)	.145(.007)	.152(.010)
D3	.281(.028)	.270(.033)	.270(.023)	.262(.013)	.266(.013)
D4	.066(.007)	.047(.007)	.047(.005)	.053(.010)	.042(.006)
D5	.108(.010)	.088(.009)	.088(.009)	.088(.010)	.084(.007)
D6	.072(.005)	.058(.006)	.064(.003)	.061(.007)	.060(.003)
D7	.185(.013)	.184(.010)	.184(.010)	.184(.010)	.184(.010)
D8	.020(.002)	.020(.003)	.020(.002)	.021(.003)	.020(.003)
D9	.173(.004)	.181(.009)	.173(.005)	.165(.004)	.164(.004)
D10	.173(.008)	.176(.006)	.173(.007)	.160(.004)	.161(.005)
<i>The mean of test error by 0–1 loss and standard deviation (RBF, robust SVC)</i>					
D1	.055(.017)	.043(.022)	.042(.017)	.037(.016)	.038(.013)
D2	.149(.010)	.148(.010)	.147(.010)	.146(.013)	.142(.013)
D3	.276(.024)	.267(.026)	.266(.024)	.271(.015)	.261(.020)
D4	.052(.009)	.048(.009)	.044(.006)	.047(.008)	.040(.005)
D5	.117(.012)	.109(.013)	.107(.012)	.107(.011)	.094(.008)
D6	.046(.007)	.045(.007)	.045(.007)	.045(.007)	.043(.006)
D7	.044(.003)	.044(.003)	.044(.003)	.044(.003)	.044(.003)
D8	.022(.003)	.022(.003)	.022(.003)	.022(.003)	.021(.002)
D9	.169(.003)	.170(.005)	.169(.004)	.168(.005)	.162(.003)
D10	.163(.003)	.163(.003)	.163(.003)	.162(.002)	.160(.004)
<i>The mean of L_1 test error and standard deviation (linear, robust SVR)</i>					
D1	.442(.324)	.337(.347)	.319(.353)	.414(.341)	.276(.321)
D2	.470(.053)	.487(.086)	.474(.087)	.490(.108)	.484(.104)
D3	.414(.038)	.351(.025)	.350(.036)	.414(.105)	.372(.043)
D4	.548(.180)	.520(.193)	.510(.146)	.562(.210)	.596(.297)
D5	.539(.019)	.531(.019)	.530(.017)	.539(.024)	.529(.018)
D6	.685(.028)	.664(.026)	.655(.027)	.685(.044)	.686(.040)
D7	.700(.016)	.691(.017)	.685(.017)	.698(.022)	.692(.014)
D8	.582(.027)	.583(.042)	.570(.031)	.589(.035)	.569(.028)
D9	.518(.015)	.510(.019)	.501(.021)	.522(.026)	.516(.019)
D10	.281(.021)	.278(.016)	.279(.016)	.269(.018)	.269(.021)
D11	.494(.010)	.488(.011)	.487(.012)	.492(.009)	.492(.008)

Table 3 continued

Data	C-SVM	CCCP- θ	OP- θ	CCCP- s	OP- s
<i>The mean of L_1 test error and standard deviation (RBF, robust SVR)</i>					
D1	.077(.049)	.069(.054)	.065(.056)	.070(.053)	.051(.040)
D2	.357(.059)	.346(.045)	.339(.045)	.332(.038)	.327(.040)
D3	.337(.052)	.299(.021)	.302(.019)	.296(.022)	.295(.022)
D4	.390(.046)	.350(.025)	.349(.023)	.357(.022)	.343(.024)
D5	.513(.024)	.519(.028)	.504(.018)	.515(.024)	.503(.019)
D6	.641(.028)	.640(.015)	.635(.017)	.634(.022)	.631(.017)
D7	.671(.011)	.669(.009)	.669(.007)	.674(.011)	.671(.009)
D8	.528(.027)	.504(.027)	.496(.024)	.511(.018)	.510(.020)
D9	.488(.012)	.490(.016)	.486(.012)	.484(.013)	.482(.014)
D10	.198(.015)	.198(.027)	.196(.025)	.194(.015)	.189(.017)
D11	.456(.016)	.441(.005)	.441(.006)	.444(.015)	.446(.015)

The noise level is 15% in SVC, while it is 5% in SVR. Smaller test error is better. The numbers in bold face indicate the better method in terms of the test error

parameter C was chosen from $\{0.01, 0.1, 1, 10, 100\}$, while the candidates of the homotopy parameters were chosen as follows:

- In OP- θ , the set of break-points $\theta_{BP} \in [0, 1]$ was considered as the candidates (note that the local solutions at each break-point have been already computed in the homotopy computation).
- In OP- s , the set of break-points in $[s_C, 0]$ was used as the candidates for robust SVC, where

$$s_C := \min_{i \in \mathbb{N}_n} y_i f_{\text{SVC}}(\mathbf{x}_i)$$

with f_{SVC} being the ordinary non-robust SVC. For robust SVR, the set of break-points in $[s_R, 0.2s_R]$ was used as the candidates, where

$$s_R := \max_{i \in \mathbb{N}_n} |y_i - f_{\text{SVR}}(\mathbf{x}_i)|$$

with f_{SVR} being the ordinary non-robust SVR.

- In CCCP- θ , the homotopy parameter θ was selected from

$$\theta \in \{1, 0.75, 0.5, 0.25, 0\}.$$

- In CCCP- s , the homotopy parameter s was selected from

$$s \in \{s_C, 0.75s_C, 0.5s_C, 0.25s_C, 0\}$$

for robust SVC, while it was selected from

$$s \in \{s_R, 0.8s_R, 0.6s_R, 0.4s_R, 0.2s_R\}$$

for robust SVR.

Note that both OP and CCCP were initialized by using the solution of standard SVM.

Table 3 represents the average and the standard deviation of the test errors on 10 different random data splits. The other results on different noise levels summarized in Appendix 11.

Fig. 7 Elapsed time when the number of (θ, s) -candidates is increased. Changing the number of hyper-parameter candidates affects the computation time of CCCP, but not OP because the entire path of solutions is computed with the infinitesimal resolution. **a** Elapsed time for CCCP and OP (linear, robust SVC). **b** Elapsed time for CCCP and OP (RBF, robust SVC). **c** Elapsed time for CCCP and OP (linear, robust SVR). **d** Elapsed time for CCCP and OP (RBF, robust SVR)

Table 4 The mean of test error by 0–1 loss and standard deviation

$\theta(L = 5)$	Test error	$\theta(L = 10)$	Test error	$\theta(L = 15)$	Test error
.800	.051(.015)	.900	.052(.016)	.933	.052(.016)
.600	.047(.014)	.800	.051(.015)	.867	.052(.015)
.400	.047(.013)	.700	.049(.015)	.800	.051(.015)
.200	.043(.015)	.600	.048(.014)	.733	.050(.014)
.000	.043(.013)	.500	.047(.014)	.667	.049(.014)
		.400	.046(.013)	.600	.048(.014)
		.300	.044(.015)	.533	.048(.015)
		.200	.043(.013)	.467	.047(.013)
		.100	.041(.013)	.400	.046(.013)
		.000	.041(.012)	.333	.045(.014)
				.267	.043(.012)
				.200	.042(.013)
				.133	.041(.012)
				.067	.040(.012)
				.000	.040(.013)

These results indicate that our proposed OP algorithm tends to find better local solutions and the degree of robustness was appropriately controlled. It seems that OP- s is the best one if we only use one of these methods.

5.3 Computation time

Second, we compared the computational costs of the entire model-building process of each method. The results are shown in Fig. 7. Note that the computational cost of the OP algorithm does not depend on the number of hyper-parameter candidates of θ or s , because the entire path of local solutions has already been computed with the infinitesimal resolution in the homotopy computation. On the other hand, the computational cost of CCCP depends on the number of hyper-parameter candidates. In our implementation of CCCP, we used the warm-start approach, i.e., we initialized CCCP with the previous solution for efficiently computing a sequence of solutions. The results indicate that the proposed OP algorithm enables stable and efficient control of robustness, while CCCP suffers a trade-off between model selection performance and computational costs.

5.4 Stability of concave-convex procedure (CCCP)

Finally, we empirically investigate the stability of CCCP algorithm. For simplicity, we only considered a linear classification problem on BreastCancerDiagnostic data with the homotopy parameters $\theta = 0$ and $s = 0$. Remember that, $\theta = 1$ corresponds to the hinge loss function for the standard convex SVM, while $\theta = 0$ corresponds to fully robust ramp loss function. Here, we used warm-start approaches for obtaining a robust SVM solution for $\theta = 0$ by considering a sequence of θ values: $\theta_{(\ell)} := 1 - \frac{\ell}{L}$, $\ell = 0, 1, \dots, L$ (another homotopy parameter s was fixed to be 0). Specifically, we started from the standard convex SVM solution with $\theta_{(0)} = 1$,

Table 5 The mean of test error by 0–1 loss and standard deviation (linear, robust SVC)

Data	CCCP- θ^5	CCCP- θ^{10}	CCCP- θ^{15}	OP- θ	CCCP- s^5	CCCP- s^{10}	CCCP- s^{15}	OP- s
D1	.050(.014)	.049(.019)	.048(.017)	.049(.016)	.055(.018)	.045(.012)	.046(.009)	.050(.016)
D2	.145(.007)	.151(.019)	.151(.019)	.151(.018)	.145(.007)	.154(.018)	.154(.018)	.152(.010)
D3	.270(.033)	.265(.014)	.266(.019)	.270(.023)	.262(.013)	.277(.023)	.270(.020)	.266(.013)
D4	.047(.007)	.048(.005)	.049(.005)	.047(.005)	.053(.010)	.046(.004)	.046(.004)	.042(.006)
D5	.088(.009)	.087(.007)	.088(.009)	.088(.009)	.088(.010)	.087(.012)	.086(.010)	.084(.007)
D6	.058(.006)	.065(.005)	.065(.005)	.064(.003)	.061(.007)	.061(.004)	.060(.003)	.060(.003)
D7	.184(.010)	.184(.010)	.184(.010)	.184(.010)	.184(.010)	.184(.009)	.184(.008)	.184(.010)
D8	.020(.003)	.020(.002)	.020(.002)	.020(.002)	.021(.003)	.020(.002)	.020(.002)	.020(.003)
D9	.181(.009)	.173(.004)	.172(.004)	.173(.005)	.165(.004)	.162(.003)	.163(.003)	.164(.004)
D10	.176(.006)	.175(.006)	.174(.007)	.173(.007)	.160(.004)	.160(.003)	.162(.003)	.161(.005)
<i>The mean of test error by 0–1 loss and standard deviation (RBF, robust SVC)</i>								
D1	.043(.022)	.043(.013)	.042(.015)	.042(.017)	.037(.016)	.047(.016)	.048(.014)	.038(.013)
D2	.148(.010)	.152(.019)	.152(.020)	.147(.010)	.146(.013)	.146(.021)	.146(.021)	.142(.013)
D3	.267(.026)	.271(.022)	.269(.026)	.266(.024)	.271(.015)	.268(.015)	.268(.016)	.261(.020)
D4	.048(.009)	.045(.006)	.045(.005)	.044(.006)	.047(.008)	.044(.004)	.044(.006)	.040(.005)
D5	.109(.013)	.110(.010)	.110(.010)	.107(.012)	.107(.011)	.097(.009)	.099(.009)	.094(.008)
D6	.045(.007)	.051(.004)	.051(.004)	.045(.007)	.045(.007)	.049(.004)	.051(.005)	.043(.006)
D7	.044(.003)	.044(.003)	.044(.003)	.044(.003)	.044(.003)	.044(.003)	.044(.003)	.044(.003)
D8	.022(.003)	.021(.002)	.021(.002)	.022(.003)	.022(.003)	.022(.003)	.021(.003)	.021(.002)
D9	.170(.005)	.169(.003)	.169(.003)	.169(.004)	.168(.005)	.164(.003)	.165(.004)	.162(.003)
D10	.163(.003)	.163(.003)	.163(.003)	.163(.003)	.162(.002)	.160(.004)	.161(.004)	.160(.004)

Table 5 continued

Data	CCCP- θ^5	CCCP- θ^{10}	CCCP- θ^{15}	OP- θ	CCCP- s^5	CCCP- s^{10}	CCCP- s^{15}	OP- s
<i>The mean of L_1 test error and standard deviation (linear, robust SVR)</i>								
D1	.337(.347)	.343(.349)	.317(.329)	.319(.353)	.414(.341)	.421(.346)	.400(.371)	.276(.321)
D2	.487(.086)	.486(.083)	.477(.080)	.474(.087)	.490(.108)	.489(.110)	.489(.110)	.484(.104)
D3	.351(.025)	.356(.030)	.354(.027)	.350(.036)	.414(.105)	.405(.085)	.413(.083)	.372(.043)
D4	.520(.193)	.484(.117)	.532(.270)	.510(.146)	.562(.210)	.539(.128)	.559(.193)	.596(.297)
D5	.531(.019)	.534(.020)	.532(.013)	.530(.017)	.539(.024)	.533(.018)	.537(.011)	.529(.018)
D6	.664(.026)	.662(.028)	.668(.025)	.655(.027)	.685(.044)	.676(.034)	.682(.037)	.686(.040)
D7	.691(.017)	.688(.016)	.687(.016)	.685(.017)	.698(.022)	.693(.013)	.700(.021)	.692(.014)
D8	.583(.042)	.573(.028)	.580(.039)	.570(.031)	.589(.035)	.581(.032)	.582(.037)	.569(.028)
D9	.510(.019)	.506(.022)	.510(.022)	.501(.021)	.522(.026)	.518(.025)	.523(.023)	.516(.019)
D10	.278(.016)	.279(.016)	.283(.019)	.279(.016)	.269(.018)	.274(.020)	.271(.020)	.269(.021)
D11	.488(.011)	.489(.010)	.487(.010)	.487(.012)	.492(.009)	.491(.010)	.492(.009)	.492(.008)
<i>The mean of L_1 test error and standard deviation (RBF, robust SVR)</i>								
D1	.069(.054)	.066(.056)	.061(.052)	.065(.056)	.070(.053)	.069(.054)	.070(.055)	.051(.040)
D2	.346(.045)	.341(.044)	.341(.044)	.339(.045)	.332(.038)	.324(.026)	.324(.026)	.327(.040)
D3	.299(.021)	.299(.019)	.300(.020)	.302(.019)	.296(.022)	.296(.020)	.295(.022)	.295(.022)
D4	.350(.025)	.348(.024)	.346(.025)	.349(.023)	.357(.022)	.356(.022)	.358(.023)	.343(.024)
D5	.519(.028)	.508(.019)	.501(.019)	.504(.018)	.515(.024)	.506(.021)	.506(.021)	.503(.019)
D6	.640(.015)	.637(.020)	.643(.021)	.635(.017)	.634(.022)	.634(.020)	.633(.019)	.631(.017)
D7	.669(.009)	.669(.008)	.668(.008)	.669(.007)	.674(.011)	.675(.010)	.672(.010)	.671(.009)
D8	.504(.027)	.504(.028)	.500(.025)	.496(.024)	.511(.018)	.514(.019)	.512(.017)	.510(.020)
D9	.490(.016)	.487(.013)	.486(.010)	.486(.012)	.484(.013)	.486(.013)	.484(.014)	.482(.014)
D10	.198(.027)	.189(.022)	.196(.023)	.196(.025)	.194(.015)	.191(.017)	.192(.015)	.189(.017)
D11	.441(.005)	.441(.005)	.440(.006)	.441(.006)	.444(.015)	.445(.015)	.448(.016)	.446(.015)

The noise level is 15% in SVC, while it is 5% in SVR. The numbers in bold face indicate the better than CCCP- θ^5 or CCCP- s^5

and used it as a warm-start initial solution for $\theta_{(1)}$, and used it as a warm-start initial solution for $\theta_{(2)}$, and so on.

Table 4 shows the average and the standard deviation of the test errors on 30 different random data splits for the three warm-start approaches with $L = 5, 10, 15$ (the performances of CCCP in Table 3 are the results with $L = 5$). The results indicate that the performances (for $\theta = 0$) were slightly better, i.e., better local optimal solutions were obtained when the length of the sequence L is larger. We conjecture that it is because the warm-start approach can be regarded as a simulated annealing, and the performances were better when the annealing step size ($= 1/L$) is smaller.

It is important to note that, if we consider the above warm-start approach with very large L , we would be able to obtain similar results as the proposed homotopy approach. In other words, the proposed homotopy approach can be considered as an efficient method for conducting simulated annealing with infinitesimally small annealing step size. Table 5 shows the same results as Table 3 for three different CCCP approaches with $L = 5, 10, 15$. As we discussed above, the performances tend to be better when L is large, although the computational cost also increases when L gets large.

6 Conclusions

In this paper, we proposed a novel robust SVM learning algorithm based on the homotopy approach that allows efficient computation of the sequence of local optimal solutions when the influence of outliers is gradually decreased. The algorithm is built on our theoretical findings about the geometric property and the optimality conditions of local solutions of robust SVM. Experimental results indicate that our algorithm tends to find better local solutions possibly due to the simulated annealing-like effect and the stable control of robustness. One of the important future works is to adopt scalable homotopy algorithms (Zhou et al. 2012) or approximate parametric programming algorithms (Giesen et al. 2012) for further improving the computational efficiency.

Acknowledgements Funding was provided by Japan Society for the Promotion of Science (Grant Nos. 26280083, 16H00886), and Japan Science and Technology Agency (Grant Nos. 13217726, 15656320).

7 Appendix 1: Proof of Lemma 2

The proof can be directly derived based on standard convex optimization theory (Boyd and Vandenberghe 2004) because the optimization problem (8) is convex if the partition $\mathcal{P} := \{\mathcal{I}, \mathcal{O}\}$ is fixed.

We denote a n dimensional vector as $\mathbf{1}$, $\mathbf{1}_{\mathcal{I}}$ or $\mathbf{1}_{\mathcal{O}} \in \mathbb{R}^n$: the all elements of $\mathbf{1}$ are one, while $\mathbf{1}_{\mathcal{I}}$ or $\mathbf{1}_{\mathcal{O}}$ indicates the all elements corresponding to $\mathcal{I}$ or $\mathcal{O}$ are one and the others are zero, and it can be obtained as $\mathbf{1} := \mathbf{1}_{\mathcal{I}} + \mathbf{1}_{\mathcal{O}}$. In this proof, we use this notation for any vectors such as $\boldsymbol{\alpha} := \boldsymbol{\alpha}_{\mathcal{I}} + \boldsymbol{\alpha}_{\mathcal{O}}$ and $\mathbf{y} := \mathbf{y}_{\mathcal{I}} + \mathbf{y}_{\mathcal{O}}$.

We rewrite the optimization problem in the conditionally optimal solution (8) as

$$\begin{aligned} & \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i \in \mathcal{I}} [1 - y_i f(\mathbf{x}_i)]_+ + C\theta \sum_{i \in \mathcal{O}} [1 - y_i f(\mathbf{x}_i)]_+ \\ & := \underset{\mathbf{w}, \boldsymbol{\xi}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \mathbf{1}_{\mathcal{I}}^{\top} \boldsymbol{\xi}_{\mathcal{I}} + C\theta \mathbf{1}_{\mathcal{O}}^{\top} \boldsymbol{\xi}_{\mathcal{O}} \quad \text{subject to} \end{aligned}$$

$$\begin{aligned} \mathbf{1}_I - \mathbf{y}_I \circ (\Phi \mathbf{w})_I &\leq \xi_I, \quad \xi_I \geq \mathbf{0}, \quad \mathbf{y}_I \circ (\Phi \mathbf{w})_I \geq s \mathbf{1}_I, \\ \mathbf{1}_O - \mathbf{y}_O \circ (\Phi \mathbf{w})_O &\leq \xi_O, \quad \xi_O \geq \mathbf{0}, \quad \mathbf{y}_O \circ (\Phi \mathbf{w})_O \leq s \mathbf{1}_O, \end{aligned}$$

where $\Phi \mathbf{w}$ represents the decision function $f_{\mathcal{P}}(\mathbf{x}_i) := (\Phi \mathbf{w})_i$ as defined (1), and $\circ$ indicates the element-wise product of two vectors.

Let us define Lagrangian as

$$\begin{aligned} L := & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \mathbf{1}_I^\top \xi_I + C \theta \mathbf{1}_O^\top \xi_O + (\alpha_I + \alpha_O)^\top (\mathbf{1} - \mathbf{y} \circ (\Phi \mathbf{w}) - \xi) \\ & - (\eta_I + \eta_O)^\top \xi - (\mathbf{v}_I - \mathbf{v}_O)^\top (\mathbf{y} \circ (\Phi \mathbf{w}) - s \mathbf{1}). \end{aligned}$$

Using the convex optimization theory (Boyd and Vandenberghe 2004), the optimality conditions are

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} &= \mathbf{w}^* - \Phi^\top (\mathbf{y} \circ (\alpha^* + \mathbf{v}_I^* - \mathbf{v}_O^*)) = \mathbf{0}, \\ \frac{\partial L}{\partial \xi_I} &= C \mathbf{1}_I - \alpha_I^* - \eta_I^* = \mathbf{0}, \quad \frac{\partial L}{\partial \xi_O} = C \theta \mathbf{1}_O - \alpha_O^* - \eta_O^* = \mathbf{0}, \\ \mathbf{1}_I - \mathbf{y}_I \circ (\Phi \mathbf{w}^*)_I &\leq \xi_I^*, \quad \xi_I^* \geq \mathbf{0}, \quad \mathbf{y}_I \circ (\Phi \mathbf{w}^*)_I \geq s \mathbf{1}_I, \\ \mathbf{1}_O - \mathbf{y}_O \circ (\Phi \mathbf{w}^*)_O &\leq \xi_O^*, \quad \xi_O^* \geq \mathbf{0}, \quad \mathbf{y}_O \circ (\Phi \mathbf{w}^*)_O \leq s \mathbf{1}_O, \\ \alpha_I^* &\geq \mathbf{0}, \quad \alpha_O^* \geq \mathbf{0}, \quad \eta_I^* \geq \mathbf{0}, \quad \eta_O^* \geq \mathbf{0}, \quad \mathbf{v}_I^* \geq \mathbf{0}, \quad \mathbf{v}_O^* \geq \mathbf{0}, \\ \alpha_I^{*\top} (\mathbf{1}_I - \mathbf{y}_I \circ (\Phi \mathbf{w}^*)_I - \xi_I^*) &= \mathbf{0}, \quad \eta_I^* \circ \xi_I^* = \mathbf{0}, \\ \alpha_O^{*\top} (\mathbf{1}_O - \mathbf{y}_O \circ (\Phi \mathbf{w}^*)_O - \xi_O^*) &= \mathbf{0}, \quad \eta_O^* \circ \xi_O^* = \mathbf{0}, \\ \mathbf{v}_I^{*\top} (\mathbf{y}_I \circ (\Phi \mathbf{w}^*)_I - s \mathbf{1}_I) &= \mathbf{0}, \\ \mathbf{v}_O^{*\top} (\mathbf{y}_O \circ (\Phi \mathbf{w}^*)_O - s \mathbf{1}_O) &= \mathbf{0}. \end{aligned}$$

We rewrite the multipliers α^* as $\alpha^* \leftarrow \alpha^* + \mathbf{v}_I^* - \mathbf{v}_O^*$, and then we obtain $\mathbf{w}^* = \Phi^\top (\mathbf{y} \circ \alpha^*)$. Since both $\mathbf{v}_I^*$ and $\mathbf{v}_O^*$ are zero when $\mathbf{y} \circ (\Phi \mathbf{w}^*) \neq s \mathbf{1}$, the optimality conditions can be rewritten as

$$\begin{aligned} y_i f_{\mathcal{P}}^*(\mathbf{x}_i) &> 1 \Rightarrow \alpha_i^* = 0, \\ y_i f_{\mathcal{P}}^*(\mathbf{x}_i) &= 1 \Rightarrow \alpha_i^* \in [0, C], \\ s < y_i f_{\mathcal{P}}^*(\mathbf{x}_i) &< 1 \Rightarrow \alpha_i^* = C, \\ y_i f_{\mathcal{P}}^*(\mathbf{x}_i) &< s \Rightarrow \alpha_i^* = C\theta, \end{aligned}$$

while $\mathbf{v}_i^*$ can be positive when $y_i (\Phi \mathbf{w}^*)_i = s$, and thus the following holds:

$$\begin{aligned} y_i f_{\mathcal{P}}^*(\mathbf{x}_i) &= s, i \in \mathcal{I} \Rightarrow \alpha_i^* \geq C, \\ y_i f_{\mathcal{P}}^*(\mathbf{x}_i) &= s, i \in \mathcal{O} \Rightarrow \alpha_i^* \leq C\theta, \end{aligned}$$

we finalized the proof. $\square$

8 Appendix 2: Proof of Theorem 3

Although $f_{\mathcal{P}}^*$ is a feasible solution, it is not a local optimum for $\theta \in [0, 1)$ and $s \leq 0$ because

$$\alpha_i \leq C\theta \text{ for } i \in \tilde{\mathcal{I}} \cap \mathcal{O}, \quad (20a)$$

$$\alpha_i \geq C \text{ for } i \in \tilde{\mathcal{O}} \cap \mathcal{I}, \quad (20b)$$

violate the KKT conditions (10) for $\tilde{\mathcal{P}}$. These feasibility and *sub*-optimality indicates that

$$J_{\tilde{\mathcal{P}}}(f_{\tilde{\mathcal{P}}}^*; \theta) < J_{\mathcal{P}}(f_{\mathcal{P}}^*; \theta), \quad (21)$$

we arrive at (12). $\square$

9 Appendix 3: Proof of Theorem 4

Sufficiency: If (13e) is true, i.e., if there are NO instances with $y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = s$, then any convex problems defined by different partitions $\tilde{\mathcal{P}} \neq \mathcal{P}$ do not have feasible solutions in the neighborhood of $f_{\mathcal{P}}^*$. This means that if $f_{\mathcal{P}}^*$ is a conditionally optimal solution, then it is locally optimal. (13a)–(13d) are sufficient for $f_{\mathcal{P}}^*$ to be conditionally optimal for the given partition $\mathcal{P}$. Thus, (13) is sufficient for $f_{\mathcal{P}}^*$ to be locally optimal.

Necessity: From Theorem 3, if there exists an instance such that $y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = s$, then $f_{\mathcal{P}}^*$ is a feasible but not locally optimal. Then (13e) is necessary for $f_{\mathcal{P}}^*$ to be locally optimal. In addition, (13a)–(13d) are also necessary for local optimality, because of every local optimal solutions are conditionally optimal for the given partition $\mathcal{P}$. Thus, (13) is necessary for $f_{\mathcal{P}}^*$ to be locally optimal. $\square$

10 Appendix 4: Implementation of D-step

In D-step, we work with the following convex problem

$$f_{\tilde{\mathcal{P}}}^* := \operatorname{argmin}_{f \in \operatorname{pol}(\tilde{\mathcal{P}}; s)} J_{\tilde{\mathcal{P}}}(f; \theta). \quad (22)$$

where, $\tilde{\mathcal{P}}$ is updated from $\mathcal{P}$ as (11).

Let us define a partition $\Pi := \{\mathcal{R}, \mathcal{E}, \mathcal{L}, \tilde{\mathcal{I}}', \tilde{\mathcal{O}}', \hat{\mathcal{O}}''\}$ of $\mathbb{N}_n$ such that

$$i \in \mathcal{R} \Rightarrow y_i f(\mathbf{x}_i) > 1, \quad (23a)$$

$$i \in \mathcal{E} \Rightarrow y_i f(\mathbf{x}_i) = 1, \quad (23b)$$

$$i \in \mathcal{L} \Rightarrow s < y_i f(\mathbf{x}_i) < 1, \quad (23c)$$

$$i \in \tilde{\mathcal{I}}' \Rightarrow y_i f(\mathbf{x}_i) = s \text{ and } i \in \tilde{\mathcal{I}}, \quad (23d)$$

$$i \in \tilde{\mathcal{O}}' \Rightarrow y_i f(\mathbf{x}_i) = s \text{ and } i \in \tilde{\mathcal{O}}, \quad (23e)$$

$$i \in \tilde{\mathcal{O}}'' \Rightarrow y_i f(\mathbf{x}_i) < s. \quad (23f)$$

If we write the conditionally optimal solution as

$$f_{\tilde{\mathcal{P}}}^*(x) := \sum_{j \in \mathbb{N}_n} \alpha_j^* y_j K(x, \mathbf{x}_j), \quad (24)$$

$\{\alpha_j^*\}_{j \in \mathbb{N}_n}$ must satisfy the following KKT conditions

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) > 1 \Rightarrow \alpha_i^* = 0, \quad (25a)$$

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = 1 \Rightarrow \alpha_i^* \in [0, C], \quad (25b)$$

$$s < y_i f_{\mathcal{P}}^*(\mathbf{x}_i) < 1 \Rightarrow \alpha_i^* = C, \quad (25c)$$

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = s, i \in \tilde{\mathcal{I}}' \Rightarrow \alpha_i^* \geq C, \quad (25d)$$

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) = s, i \in \tilde{\mathcal{O}}' \Rightarrow \alpha_i^* \leq C\theta, \quad (25e)$$

$$y_i f_{\mathcal{P}}^*(\mathbf{x}_i) < s, i \in \tilde{\mathcal{O}}'' \Rightarrow \alpha_i^* = C\theta. \quad (25f)$$

At the beginning of the D-step, $f_{\mathcal{P}}^*(\mathbf{x}_i)$ violates the KKT conditions by

$$\Delta f_i := y_i \begin{bmatrix} \mathbf{K}_{i, \Delta_{\mathcal{I} \rightarrow \mathcal{O}}} & \mathbf{K}_{i, \Delta_{\mathcal{O} \rightarrow \mathcal{I}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{\Delta_{\mathcal{I} \rightarrow \mathcal{O}}}^{(\text{bef})} - \mathbf{1}C\theta \\ \boldsymbol{\alpha}_{\Delta_{\mathcal{O} \rightarrow \mathcal{I}}}^{(\text{bef})} - \mathbf{1}C \end{bmatrix}.$$

where $\boldsymbol{\alpha}^{(\text{bef})}$ is the corresponding $\boldsymbol{\alpha}$ at the beginning of the D-step, while $\Delta_{\mathcal{I} \rightarrow \mathcal{O}}$ and $\Delta_{\mathcal{O} \rightarrow \mathcal{I}}$ denote the difference in $\tilde{\mathcal{P}}$ and $\mathcal{P}$ defined as

$$\Delta_{\mathcal{I} \rightarrow \mathcal{O}} := \{i \in \mathcal{I} \mid y_i f_{\mathcal{P}}(\mathbf{x}_i) = s\},$$

$$\Delta_{\mathcal{O} \rightarrow \mathcal{I}} := \{i \in \mathcal{O} \mid y_i f_{\mathcal{P}}(\mathbf{x}_i) = s\}.$$

Then, we consider the following another parametrized problem with a parameter $\mu \in [0, 1]$:

$$f_{\tilde{\mathcal{P}}}(\mathbf{x}_i; \mu) := f_{\mathcal{P}}(\mathbf{x}_i) + \mu \Delta f_i \quad \forall i \in \mathbb{N}_n.$$

In order to always satisfy the KKT conditions for $f_{\tilde{\mathcal{P}}}(\mathbf{x}_i; \mu)$, we solve the following linear system

$$\begin{aligned} \mathbf{Q}_{\mathcal{A}, \mathcal{A}} \begin{bmatrix} \boldsymbol{\alpha}_{\mathcal{E}} \\ \boldsymbol{\alpha}_{\tilde{\mathcal{I}}'} \\ \boldsymbol{\alpha}_{\tilde{\mathcal{O}}'} \end{bmatrix} &= \begin{bmatrix} \mathbf{1} \\ s \\ s \end{bmatrix} - \mathbf{Q}_{\mathcal{A}, \mathcal{L}} \mathbf{1}C - \mathbf{Q}_{\mathcal{A}, \tilde{\mathcal{O}}''} \mathbf{1}C\theta \\ &\quad - \begin{bmatrix} \mathbf{Q}_{\mathcal{A}, \Delta_{\mathcal{I} \rightarrow \mathcal{O}}} & \mathbf{Q}_{\mathcal{A}, \Delta_{\mathcal{O} \rightarrow \mathcal{I}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{\Delta_{\mathcal{I} \rightarrow \mathcal{O}}}^{(\text{bef})} - \mathbf{1}C\theta \\ \boldsymbol{\alpha}_{\Delta_{\mathcal{O} \rightarrow \mathcal{I}}}^{(\text{bef})} - \mathbf{1}C \end{bmatrix} \mu, \end{aligned}$$

where $\mathcal{A} := \{\mathcal{E}, \tilde{\mathcal{I}}', \tilde{\mathcal{O}}'\}$. This linear system can also be solved by using the piecewise-linear parametric programming while the scalar parameter μ is continuously moved from 1 to 0.

In this parametric problem, we can show that $f_{\tilde{\mathcal{P}}}^*(\mathbf{x}_i; \mu) = f_{\mathcal{P}}^*(\mathbf{x}_i)$ if $\mu = 1$ and $f_{\tilde{\mathcal{P}}}^*(\mathbf{x}_i; \mu) = f_{\mathcal{P}}^*(\mathbf{x}_i)$ if $\mu = 0$ for all $i \in \mathbb{N}_n$.

Since the number of elements in $\Delta_{\mathcal{I} \rightarrow \mathcal{O}}$ and $\Delta_{\mathcal{O} \rightarrow \mathcal{I}}$ are typically small, the D-step can be efficiently implemented by a technique used in the context of incremental learning (Cauwenberghs and Poggio 2001).

11 Appendix 5: Generalization performance on different noise levels

Tables 6 and 7 represent the average and the standard deviation of the test errors on 10 different random data splits with more higher noise level. It seems that our proposed OP algorithm tends to find better local solutions even if the noisy level is more higher.

Table 6 The mean of test error by 0–1 loss and standard deviation (linear, robust SVC)

Data	C-SVM	CCCP- θ	OP- θ	CCCP- s	OP- s
D1	.053(.013)	.052(.022)	.048(.024)	.052(.018)	.048(.021)
D2	.152(.018)	.152(.018)	.152(.018)	.157(.021)	.163(.013)
D3	.277(.025)	.274(.020)	.266(.029)	.279(.026)	.268(.024)
D4	.077(.015)	.051(.014)	.044(.012)	.054(.015)	.043(.009)
D5	.124(.014)	.102(.014)	.099(.013)	.099(.009)	.097(.010)
D6	.067(.003)	.064(.003)	.063(.004)	.064(.002)	.063(.004)
D7	.218(.012)	.218(.012)	.218(.012)	.216(.010)	.217(.010)
D8	.021(.003)	.021(.003)	.021(.003)	.022(.003)	.022(.002)
D9	.177(.014)	.178(.013)	.178(.014)	.163(.004)	.165(.005)
D10	.187(.018)	.188(.018)	.188(.019)	.163(.004)	.159(.004)

The mean of test error by 0–1 loss and standard deviation (RBF, robust SVC)

D1	.046(.009)	.044(.012)	.048(.024)	.044(.013)	.050(.031)
D2	.157(.023)	.161(.023)	.161(.023)	.161(.028)	.166(.026)
D3	.279(.027)	.274(.029)	.271(.023)	.283(.022)	.280(.021)
D4	.058(.011)	.050(.006)	.046(.012)	.049(.012)	.043(.008)
D5	.129(.016)	.124(.015)	.124(.015)	.126(.016)	.105(.009)
D6	.048(.004)	.049(.004)	.049(.004)	.049(.005)	.048(.006)
D7	.045(.003)	.046(.003)	.046(.003)	.045(.003)	.045(.003)
D8	.024(.003)	.023(.003)	.023(.003)	.024(.003)	.024(.003)
D9	.169(.004)	.169(.004)	.168(.005)	.164(.005)	.163(.003)
D10	.163(.004)	.163(.004)	.163(.003)	.160(.004)	.159(.004)

The mean of L_1 test error and standard deviation (linear, robust SVR)

D1	.551(.233)	.246(.293)	.225(.271)	.357(.264)	.421(.273)
D2	.547(.183)	.496(.074)	.511(.108)	.542(.186)	.486(.041)
D3	.493(.105)	.440(.125)	.447(.122)	.586(.319)	.568(.304)
D4	.546(.079)	.473(.088)	.469(.086)	.476(.067)	.472(.072)
D5	.561(.034)	.549(.029)	.543(.029)	.547(.028)	.548(.027)
D6	.759(.042)	.706(.053)	.696(.059)	.712(.047)	.703(.043)
D7	.710(.008)	.704(.010)	.693(.016)	.719(.025)	.717(.024)
D8	.595(.030)	.571(.014)	.568(.012)	.583(.021)	.576(.017)
D9	.548(.035)	.530(.026)	.530(.026)	.539(.034)	.535(.037)
D10	.301(.018)	.290(.016)	.292(.016)	.288(.013)	.290(.021)
D11	.505(.020)	.501(.008)	.501(.007)	.502(.016)	.501(.016)

The mean of L_1 test error and standard deviation (RBF, robust SVR)

D1	.087(.040)	.064(.051)	.059(.048)	.074(.047)	.067(.044)
D2	.394(.049)	.369(.049)	.361(.051)	.377(.055)	.332(.062)
D3	.374(.039)	.345(.044)	.346(.043)	.342(.043)	.328(.029)
D4	.394(.059)	.399(.050)	.393(.048)	.385(.049)	.381(.046)
D5	.524(.015)	.520(.015)	.520(.013)	.516(.022)	.518(.026)
D6	.667(.027)	.655(.025)	.656(.023)	.668(.028)	.661(.024)
D7	.676(.011)	.674(.008)	.673(.008)	.674(.008)	.672(.007)

Table 6 continued

Data	C-SVM	CCCP- θ	OP- θ	CCCP- s	OP- s
D8	.560(.018)	.541(.022)	.535(.026)	.542(.015)	.539(.027)
D9	.499(.016)	.493(.014)	.491(.012)	.495(.017)	.492(.015)
D10	.228(.030)	.220(.017)	.219(.017)	.207(.010)	.208(.010)
D11	.466(.018)	.453(.020)	.454(.020)	.466(.018)	.464(.017)

The noise level is 20% in SVC, while it is 10% in SVR. The numbers in bold face indicate the better method in terms of the test error

Table 7 The mean of test error by 0–1 loss and standard deviation (linear, robust SVC)

Data	C-SVM	CCCP- θ	OP- θ	CCCP- s	OP- s
D1	.072(.028)	.065(.028)	.064(.031)	.066(.024)	.068(.029)
D2	.151(.018)	.151(.018)	.151(.018)	.153(.019)	.154(.019)
D3	.275(.020)	.272(.018)	.274(.015)	.271(.017)	.275(.013)
D4	.088(.014)	.060(.011)	.057(.010)	.062(.011)	.046(.005)
D5	.128(.013)	.105(.013)	.103(.011)	.103(.014)	.096(.014)
D6	.071(.003)	.067(.005)	.066(.006)	.067(.004)	.066(.005)
D7	.241(.009)	.239(.009)	.240(.009)	.239(.009)	.241(.010)
D8	.022(.003)	.022(.003)	.022(.003)	.022(.003)	.022(.003)
D9	.201(.019)	.201(.019)	.201(.018)	.169(.007)	.168(.007)
D10	.198(.015)	.198(.014)	.199(.015)	.169(.004)	.165(.004)

The mean of test error by 0–1 loss and standard deviation (RBF, robust SVC)

Data	C-SVC	CCCP- θ	OP- θ	CCCP- s	OP- s
D1	.061(.030)	.057(.019)	.062(.021)	.063(.023)	.061(.025)
D2	.151(.018)	.152(.018)	.151(.018)	.152(.023)	.149(.023)
D3	.277(.020)	.275(.018)	.275(.017)	.270(.022)	.263(.017)
D4	.065(.010)	.052(.009)	.052(.007)	.055(.008)	.044(.007)
D5	.131(.012)	.125(.014)	.123(.011)	.125(.010)	.106(.007)
D6	.059(.007)	.058(.007)	.057(.006)	.061(.007)	.061(.007)
D7	.050(.007)	.049(.006)	.049(.006)	.049(.005)	.049(.006)
D8	.025(.003)	.024(.003)	.024(.003)	.025(.003)	.024(.003)
D9	.172(.007)	.173(.007)	.172(.006)	.168(.004)	.166(.006)
D10	.167(.005)	.168(.005)	.168(.005)	.164(.004)	.166(.007)

The mean of L_1 test error and standard deviation (linear, robust SVR)

D1	.581(.275)	.421(.326)	.499(.372)	.524(.383)	.462(.289)
D2	.607(.405)	.584(.411)	.570(.415)	.745(.572)	.762(.548)
D3	.547(.190)	.531(.210)	.492(.186)	.716(.360)	.640(.242)
D4	.637(.206)	.531(.212)	.631(.386)	.752(.501)	.767(.550)
D5	.557(.025)	.547(.016)	.542(.023)	.568(.025)	.551(.019)
D6	.757(.038)	.672(.054)	.671(.064)	.722(.064)	.711(.047)

Table 7 continued

Data	C-SVM	CCCP- θ	OP- θ	CCCP- s	OP- s
D7	.720(.016)	.710(.016)	.703(.024)	.713(.017)	.713(.017)
D8	.618(.051)	.585(.025)	.580(.022)	.599(.018)	.596(.023)
D9	.561(.025)	.515(.020)	.519(.021)	.552(.047)	.540(.036)
D10	.303(.017)	.285(.014)	.283(.012)	.282(.018)	.283(.017)
D11	.511(.009)	.503(.012)	.503(.010)	.512(.023)	.508(.023)
<i>The mean of L_1 test error and standard deviation (RBF, robust SVR)</i>					
D1	.237(.151)	.164(.140)	.153(.138)	.202(.163)	.190(.155)
D2	.392(.067)	.355(.059)	.348(.072)	.353(.059)	.358(.056)
D3	.387(.036)	.333(.037)	.339(.042)	.340(.037)	.325(.039)
D4	.435(.069)	.428(.077)	.420(.075)	.415(.072)	.416(.058)
D5	.532(.016)	.532(.017)	.529(.020)	.525(.018)	.523(.020)
D6	.665(.034)	.648(.036)	.644(.033)	.649(.036)	.643(.034)
D7	.700(.027)	.687(.024)	.683(.025)	.682(.013)	.682(.016)
D8	.553(.025)	.547(.020)	.544(.019)	.545(.026)	.537(.025)
D9	.499(.008)	.490(.011)	.492(.012)	.498(.015)	.494(.013)
D10	.243(.021)	.241(.016)	.238(.015)	.228(.023)	.228(.022)
D11	.482(.024)	.463(.020)	.464(.020)	.470(.011)	.470(.011)

The noise level is 25% in SVC, while it is 15% in SVR. The numbers in bold face indicate the better method in terms of the test error

References

- Allgower, E. L., & George, K. (1993). Continuation and path following. *Acta Numerica*, 2, 1–63.
- Bartlett, P. L., & Mendelson, S. (2003). Rademacher and gaussian complexities: Risk bounds and structural results. *The Journal of Machine Learning Research*, 3, 463–482.
- Best, M. J. (1996). An algorithm for the solution of the parametric quadratic programming problem. *Applied Mathematics and Parallel Computing*, pp. 57–76.
- Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual ACM workshop on computational learning theory*, pp. 144–152.
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.
- Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems*, 13, 409–415.
- Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Trading convexity for scalability. In *Proceedings of the 23rd international conference on machine learning*, pp. 201–208.
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273–297.
- DeCoste, D., & Wagstaff, K. (2000). Alpha seeding for support vector machines. In *Proceeding of the sixth ACM SIGKDD international conference on knowledge discovery and data mining*.
- Freund, Y. (2009). A more robust boosting algorithm. [arXiv:0905.2138](https://arxiv.org/abs/0905.2138).
- Gal, T. (1995). *Postoptimal analysis, parametric programming, and related topics*. Walter de Gruyter.
- Giesen, J., Jaggi, M., & Laue, S. (2012). Approximating parameterized convex optimization problems. *ACM Transactions on Algorithms*, 9.
- Hastie, T., Rosset, S., Tibshirani, R., & Zhu, J. (2004). The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5, 1391–415.
- Hromkovic, J. (2001). *Algorithmics for hard problems*. Berlin: Springer.
- Krause, N., & Singer, Y. (2004). Leveraging the margin more carefully. In *Proceedings of the 21st international conference on machine learning*, 2004, pp. 63–70.
- Liu, Y., & Shen, X. (2006). Multicategory ψ -learning. *Journal of the American Statistical Association*, 101, 98.

- Liu, Y., Shen, X., & Doss, H. (2005). Multicategory ψ -learning and support vector machine: Computational tools. *Journal of Computational and Graphical Statistics*, 14, 219–236.
- Masnadi-Shiraze, H., & Vasconcelos, N. (2000). Functional gradient techniques for combining hypotheses. In *Advances in large margin classifiers*. MIT Press, pp. 221–246.
- Masnadi-Shirazi, H., & Vasconcelos, N. (2009). On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in neural information processing systems*, 22, 1049–1056.
- Mazumder, R., Friedman, J. H., & Hastie, T. (2011). Sparsenet: Coordinate descent with non-convex penalties. *Journal of the American Statistical Association*, 106, 1125–1138.
- Ogawa, K., Imamura, M., Takeuchi, I., & Sugiyama, M. (2013). Infinitesimal annealing for training semi-supervised support vector machines. In *Proceedings of the 30th international conference on machine learning*.
- Ritter, K. (1984). On parametric linear and quadratic programming problems. In *Mathematical programming: proceedings of the international congress on mathematical programming*, pp. 307–335.
- Shen, X., Tseng, G., Zhang, X., & Wong, W. H. (2003). On ψ -learning. *Journal of the American Statistical Association*, 98(463), 724–734.
- Suzumura, S., Ogawa, K., Sugiyama, M., & Takeuchi, I. (2014). Outlier path: A homotopy algorithm for robust SVM. In *Proceedings of the 31st international conference on machine learning (ICML-14)*, pp. 1098–1106.
- Vapnik, V. N. (1998). *Statistical learning theory*. Wiley Inter-Science.
- Wu, Y., & Liu, Y. (2007). Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102, 974–983.
- Xu, L., Crammer, K., & Schuurmans, D. (2006). Robust support vector machine training via convex outlier ablation. In *Proceedings of the national conference on artificial intelligence (AAAI)*.
- Yu, Y., Yang, M., Xu, L., White, M., & Schuurmans, D. (2010). Relaxed clipping: A global training method for robust regression and classification. In *Advances in neural information processing systems*, 23.
- Yuille, A. L., & Rangarajan, A. (2002). The concave-convex procedure (cccp). In *Advances in neural information processing systems*, 14.
- Zhang, C. H. (2010). Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, 38, 894–942.
- Zhou, H., Armagan, A., & Dunson, D. B. (2012). Path following and empirical Bayes model selection for sparse regression. [arXiv:1201.3528](https://arxiv.org/abs/1201.3528).