Mathematical programs with complementarity constraints and application to hyperparameter tuning

Samuel J. Ward [1*], Alain Zemkoho [1] and Selin Damla Ahipaşaoğlu [1]

[1]School of Mathematical Sciences, University of Southampton, Southampton, SO17 1BJ, United Kingdom.

*Corresponding author(s). E-mail(s): s.ward@soton.ac.uk;
Contributing authors: a.b.zemkoho@soton.ac.uk; s.d.ahipasaoglu@soton.ac.uk;

Abstract

We consider the Mathematical Program with Complementarity Constraints (MPCC). One of the main challenges in solving this problem is the systematic failure of standard Constraint Qualifications (CQs). Carefully accounting for the combinatorial nature of the complementarity constraints, tractable versions of the Mangasarian Fromovitz Constraint Qualification (MFCQ) have been designed and widely studied in the literature. This paper looks closely at two such MPCC-MFCQs and their influence on MPCC algorithms. As a key contribution, we prove the convergence of the sequential penalisation and Scholtes relaxation algorithms under a relaxed MPCC-MFCQ that is much weaker than the CQs currently used in the literature. We then form the problem of tuning hyperparameters of a nonlinear Support Vector Machine (SVM), a fundamental machine learning problem for classification, as a MPCC. For this application, we establish that the aforementioned relaxed MPCC-MFCQ holds under a very mild assumption. Moreover, we program robust implementations and comprehensive numerical experimentation on real-world data sets, where we show that the sequential penalisation method applied to the MPCC formulation for tuning SVM hyperparameters can outperform both the Scholtes relaxation technique and the state-of-the-art derivative-free methods from the machine learning literature.

Keywords: Mathematical program with complementarity constraints, Hyperparameter optimisation, Bilevel optimisation, Penalisation, Relaxation, Support vector machines

MSC Classification: 49M37, 65K05, 65K15, 90C30, 90C33

1 Introduction

The Mathematical Program with Complementarity Constraints (MPCC) is now a well-established optimisation problem, with a wide range of applications including in economics [1], chemical engineering [2], energy distribution [3], transportation [4], and machine learning (see Section 4 for an important problem class, as well as some relevant references).

Most standard optimisation algorithms are ineffective at finding solutions of MPCCs, in part due to the systematic failure of classical Constraint Qualifications (CQs) and the combinatorial nature of the problem's feasible set [5, Chapter 3]. To address the issue with standard CQs, many alternative tractable ones tailored to the complementarity constraints have been developed in the literature

(see, e.g. [6–10]). Considering the importance of the Mangasarian Fromovitz Constraint Qualification (MFCQ) in optimisation, it has been particularly at the centre of attention. Two categories of MFCQ–type CQs for the MPCC have emerged in the literature; one is based on the decomposition of the complementarity constraints, especially the constraints associated with the biactive index sets, to build suitable versions of MFCQ that can hold for MPCCs [6, 8, 11]. A second category, constructed from the lens of variational analysis, consists of reformulating the complementarity constraints either with non-smooth operator functions or sets that lead to normal cone operations that restore the fulfilment of such CQs; see, e.g., [10, 12, 13].

In the literature, the MPCC-MFCQ-T, where we use the label "T" to represent the fact that it is built from a tightened decomposition of the complementarity constraints, has emerged as one of the most used CQs for MPCCs; see, e.g., [6, 8, 9, 14]. However, looking at the MPCC-MFCQ-T very closely, one can observe that it boils down to the MPCC-LICQ, the well-known tailored linear independence constraint qualification for MPCCs, when the feasible set does not involve any inequality constraint apart from the ones in the complementarity constraints; cf. Remark 2.7 in Section 2. For this reason, and considering the fact that the MPCC-LICQ is very strong, the MPCC-MFCQ-T fails for many MPCC problems (see Example 2.6 and Proposition 4.4).

Alternatively, there is another version of MPCC-tailored MFCQ, studied in [15, 16], based on a relaxation of the complementarity constraints associated with the biactive index set. Hence, we denote it as MPCC-MFCQ-R with the label "R" representing the corresponding relaxation framework. We show in this paper that the MPCC-MFCQ-R is strictly weaker than the MPCC-MFCQ-T; see Theorem 2.5 and Example 2.6. Based on this observation, we revisit two classical algorithms for MPCCs that rely on MPCC-type CQs for their convergence results, namely, the partial penalisation and Scholtes relaxation methods, which have been widely studied and used in the literature (see, e.g., [16–18] and [11, 19–22], respectively).

The partial penalisation method consists of moving the product of the complementarity constraint functions from the feasible set to the objective. Two options have been pursued for this approach: (i) the *exact* penalisation, where the penalty parameter is fixed [16], and (ii) the *sequential* version, where the penalty parameter increases sequentially throughout the algorithm [18]. As it is well known in constrained optimisation, a big challenge with exact penalisation is identifying a suitable penalty parameter which is highly impactful on the numerical performance. Crucially, for sequential penalisation algorithms, the selection of the initial penalisation parameter is not too critical, as theoretically, under suitable assumptions, the algorithm gets to a point of stability where a solution is found.

Considering this practical advantage of the sequential approach, we focus on it in the analysis in this paper. The article [18] shows that the sequential partial penalisation algorithm, in conjunction with an interior-point method for the subproblems, converges to an S-stationary point, under the MPCC-LICQ, as a key assumption. In this paper, we establish the convergence of the sequential partial penalisation method under the MPCC-MFCQ-R, irrespective of the algorithm for solving the corresponding subproblems; see Theorem 3.1. Not only is MPCC-MFCQ-R far weaker than the MPCC-LICQ, but the subproblems can be solved with any method, and not necessarily with an interior-point method, as required in [18].

As for the Scholtes relaxation algorithm, its convergence has been established under the MPCC-MFCQ-T; see, e.g., [9]. Here, we show that the MPCC-MFCQ-R is sufficient to establish this convergence; cf. Theorem 3.4. Moreover, to make the method more practically relevant, in our proof, we only require the computation of approximate Karush-Kuhn-Tucker (KKT) points for the relaxed problem at each iteration. This is a significant departure from the existing convergence results, where it is usually assumed that the KKT points of the relaxed problem are computed exactly, something that is typically not possible in a practical implementation.

Subsequently, we explore the application of our aforementioned results for general MPCCs to the problem of computing optimal hyperparameters for a Support Vector Machine (SVM). Note that the SVM represents one of the most potent tools for classification in machine learning. Since its introduction in 1995 by Cortes and Vapnik [23, 24], it continues to be prolifically used to solve a

multitude of practical problems, including facial authentication [25], protein sequencing [26] and speech recognition [27]; it also sets a competitive benchmark in all classification problems.

One of the biggest challenges with SVMs is choosing their hyperparameters. Unlike model parameters, which are learnt from patterns in the training data, hyperparameters define the learning process itself and so must be considered separately. The two most common hyperparameters are the *regularisation parameter*, which balances the empirical loss against the complexity of the model, and the *choice of kernel*. There is no one model that will be best suited to all scenarios [28].

Traditionally, in the machine learning literature, derivative free search algorithms such as grid search and Bayesian optimisation have been used to select the hyperparameters that provide the best validation accuracy [29]. However, these methods all assume that training is a black box – that a new model must, each time, be trained in its entirety before being evaluated. They are general methods that neglect the specific structure of the SVM problem and scale very poorly with multiple hyperparameters.

More recently, a new approach to hyperparameter selection using bilevel optimisation has been proposed [30–32]. It enables the simultaneous training of the model while hyperparameter tuning is being conducted. The bilevel optimisation technique has been applied thoroughly and with great success to linear models. For example, the series of papers [33–35] proposes some non-smooth optimisation methods for selecting linear support vector regression and classification hyperparameters. Other authors have created a similar framework but propose different approaches to solving the bilevel program; e.g., a value function-based descent, global relaxation, and stochastic gradient descent methods have been proposed in [36], [37] and [38], respectively. The aforementioned works are on linear models; thus, enabling some mathematical simplicity, but such a framework is less useful in practice, as real-world data sets are rarely linearly separable.

Of particular interest to us is the sequence of papers [39–41], as they suggest an extension of the bilevel optimisation framework to nonlinear kernel models, while proposing an MPCC reformulation for the corresponding problem. However, they do not provide much theoretical insight into the problem. The paper [42] begins addressing this gap. Our paper expands on the latter work by first rigorously constructing the MPCC model of tuning the hyperparameters of nonlinear kernel SVMs. We then prove for the first time that the MPCC-MFCQ-R automatically holds at any feasible point, provided that the corresponding pair of hyperparameters is positive; cf. Proposition 4.2. Note that this positiveness is not a strong requirement, as for SVMs, those values cannot be zero in practice. As a consequence of this result, the sequential penalisation and Scholtes relaxation algorithms, discussed above, converge automatically, almost for free, for the MPCC model. It is important to note that we also show that for the MPCC hyperparameter optimisation model, the MPCC-MFCQ-T fails at any of its feasible points with more than two biactive indices; see Proposition 4.4.

As another key contribution of this paper, we conduct a robust implementation of the partial penalisation and Scholtes relaxation methods and compare them to each other, and also to classical hyperparameter tuning algorithms in the machine learning literature. First, we observe that the sequential penalisation method, on average, not only runs faster but also converges to a stationary point with a lower objective function value than the Scholtes relaxation algorithm. Moreover, we show that the sequential partial penalisation technique can outperform the traditional derivative-free methods such as grid, random, Bayesian and pattern search. Note that for the experiments, we use 18 well-known classification datasets from the literature, where the median problem size is 1773 variables, 1015 constraints (excluding complementarity constraints) and 1010 pairs of complementarity constraints. To the best of our knowledge, these problems are far larger than any others that have been used for experiments in the context of non-convex MPCCs. For illustration, observe that for the MacMPEC collection [43], which is classical for MPCC numerical experimentation, the median problem size is 49 variables, 20 constraints (excluding complementarity constraints) and 12 pairs of complementarity constraints.

For the remainder of the paper, note that Section 2 covers preliminaries with particular attention on the difference between the two MPCC-tailored CQs that are a key focus in this paper; i.e., the MPCC-MFCQ-T and MPCC-MFCQ-R. In Section 3, we present the sequential inexact penalisation and relaxation methods and prove new convergence results which allow these methods to be

applicable to a broader range of problems. In Section 4, we construct the MPCC optimisation-based hyperparameter tuning model for SVMs and prove that it satisfies MPCC-MFCQ-R but not MPCC-MFCQ-T. Finally, Section 5 presents our numerical experiments and a demonstration that the MPCC model solved with sequential penalisation can outperform traditional derivative-free methods from the machine learning literature. Many technical details, which could further facilitate the comprehension and implementation of some aspects of the paper, are provided in the supplementary materials at the end of this document.

2 Preliminaries on MPCCs

For a number of concepts used throughout the paper, we refer to a general Nonlinear Program (NLP) defined, for given dimensions $n, p, q \in \mathbb{N}$, as

$$\begin{array}{lll} \underset{\boldsymbol{z}\in\mathbb{R}^n}{\text{minimise}} & f(\boldsymbol{z}) & \\ \text{subject to} & g_i(\boldsymbol{z}) \geq 0 & \text{for } i = 1, ..., p, \\ & h_i(\boldsymbol{z}) = 0 & \text{for } i = 1, ..., q, \end{array}$$

where the functions $f : \mathbb{R}^n \to \mathbb{R},\ g : \mathbb{R}^n \to \mathbb{R}^p,\ h : \mathbb{R}^n \to \mathbb{R}^q$ representing the objective, inequality, and equality constraint functions, respectively, are continuously differentiable. However, the main focus of our analysis is the Mathematical Program with complementarity Constraints (MPCC), defined for given dimensions $n, p, q, r \in \mathbb{N}$, as

$$\begin{array}{lll} \underset{\boldsymbol{z}\in\mathbb{R}^n}{\text{minimise}} & f(\boldsymbol{z}) & \\ \text{subject to} & g_i(\boldsymbol{z}) \geq 0 & \text{for } i = 1, ..., p, \\ & h_i(\boldsymbol{z}) = 0 & \text{for } i = 1, ..., q, \\ & G_i(\boldsymbol{z}) \geq 0,\ H_i(\boldsymbol{z}) \geq 0,\ G_i(\boldsymbol{z})H_i(\boldsymbol{z}) = 0 & \text{for } i = 1, ..., r, \end{array}$$

where the functions $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^p$, $h : \mathbb{R}^n \to \mathbb{R}^q$ and $G, H : \mathbb{R}^n \to \mathbb{R}^r$ are continuously differentiable. G and H are referred to as complementarity constraint functions and $G_i(\boldsymbol{z})H_i(\boldsymbol{z})$ as the product term. The last line of problem (MPCC) can be written compactly as $0 \leq G(\boldsymbol{z}) \perp H(\boldsymbol{z}) \geq 0$.

2.1 Constraint qualifications

In continuous constrained optimisation, constraint qualifications (CQs) are usually required to construct necessary optimality conditions and prove the convergence of numerical algorithms. To introduce a CQ for problem (NLP), we define the *index sets*

$$I^g(\bar{\boldsymbol{z}}) \coloneqq \{i \in \{1, ..., p\} :\ g_i(\bar{\boldsymbol{z}}) = 0\} \ \text{ and } \ I^h \coloneqq \{1, ..., q\}\,.$$

Based on this notation, the *Linear Independence Constraint Qualification* (LICQ) is satisfied at a feasible point $\bar{\boldsymbol{z}} \in \mathbb{R}^n$ of the (NLP) iff the family of gradients

$$\left\{\nabla h_i(\bar{\boldsymbol{z}}) : i \in I^h\right\} \cup \{\nabla g_i(\bar{\boldsymbol{z}}) : i \in I^g(\bar{\boldsymbol{z}})\} \ \text{ is linearly independent,}$$

while, the *Mangasarian-Fromovitz Constraint Qualification* (MFCQ) will be said to hold at a feasible point $\bar{\boldsymbol{z}} \in \mathbb{R}^n$ of the (NLP) iff

$$\begin{array}{ll} & \left\{\nabla h_i(\bar{\boldsymbol{z}}) : i \in I^h\right\} \text{ is linearly independent and} \\ \exists\ \boldsymbol{d} \in \mathbb{R}^n : & \begin{cases} \nabla h_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} = 0 & \forall i \in I^h, \\ \nabla g_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} > 0 & \forall i \in I^g(\bar{\boldsymbol{z}}). \end{cases} \end{array}$$

Obviously, the LICQ is stronger than the MFCQ. These CQs play a fundamental role in nonlinear constrained optimisation thanks to their versatility in deriving fundamental properties for problem NLP; see the excellent tutorial in [45] for an overview of these properties.

Unfortunately, it is well known that LICQ and MFCQ are systematically violated at every feasible point of problem MPCC when viewed as a constrained problem with equality and inequality constraints in the format described in NLP [46]. To observe this, consider an index $i = 1, \ldots, r$, a feasible point $\bar{\boldsymbol{z}}$ and the three constraints $G_i(\bar{\boldsymbol{z}}) \geq 0$, $H_i(\bar{\boldsymbol{z}}) \geq 0$ and $G_i(\bar{\boldsymbol{z}})H_i(\bar{\boldsymbol{z}}) = 0$. Assuming, without loss of generality, that $H_i(\bar{\boldsymbol{z}})$ is zero, the gradient of the second constraint $\nabla H_i(\bar{\boldsymbol{z}})$ and the gradient of the third constraint $G_i(\bar{\boldsymbol{z}})\nabla H_i(\bar{\boldsymbol{z}})$ (by the product rule) are not linearly independent.

The main challenge in addressing problem MPCC comes from the combinatorial nature of the complementarity constraints. For each index $i = 1, \ldots, r$, either H_i or G_i may be positive, but not both. To further study this, we partition these cases by defining the following index sets:

$$\begin{aligned} I^G(\bar{\boldsymbol{z}}) &:= \{i \in \{1, ..., r\} : \quad G_i(\bar{\boldsymbol{z}}) = 0, H_i(\bar{\boldsymbol{z}}) > 0\}, \\ I^H(\bar{\boldsymbol{z}}) &:= \{i \in \{1, ..., r\} : \quad G_i(\bar{\boldsymbol{z}}) > 0, H_i(\bar{\boldsymbol{z}}) = 0\}, \\ I^{GH}(\bar{\boldsymbol{z}}) &:= \{i \in \{1, ..., r\} : \quad G_i(\bar{\boldsymbol{z}}) = 0, H_i(\bar{\boldsymbol{z}}) = 0\}. \end{aligned} \tag{2.1}$$

Based on these index sets, at a given feasible point $\bar{\boldsymbol{z}}$ of problem MPCC, we first introduce the corresponding *Tightened Nonlinear Program* (TNLP($\bar{\boldsymbol{z}}$)):

$$\begin{array}{lllll} \underset{\boldsymbol{z} \in \mathbb{R}^n}{\text{minimise}} & f(\boldsymbol{z}) & & & \\ \text{subject to} & g_i(\boldsymbol{z}) \geq 0 & & & \text{for } i = 1, \ldots, p, \\ & h_i(\boldsymbol{z}) = 0 & & & \text{for } i = 1, \ldots, q, \\ & G_i(\boldsymbol{z}) = 0, & H_i(\boldsymbol{z}) \geq 0 & & \text{for } i \in I^G(\bar{\boldsymbol{z}}), \\ & G_i(\boldsymbol{z}) \geq 0, & H_i(\boldsymbol{z}) = 0 & & \text{for } i \in I^H(\bar{\boldsymbol{z}}), \\ & G_i(\boldsymbol{z}) = 0, & H_i(\boldsymbol{z}) = 0 & & \text{for } i \in I^{GH}(\bar{\boldsymbol{z}}). \end{array} \tag{TNLP($\bar{\boldsymbol{z}}$)}$$

It is very important to remember that these index sets depend on the point $\bar{\boldsymbol{z}}$, so the (TNLP($\bar{\boldsymbol{z}}$)) may be a different program when constructed around a different point. The decomposition of the complementarity constraints in MPCC makes the fulfilment of standard CQs more likely. Hence, problem (TNLP($\bar{\boldsymbol{z}}$)) is commonly used to define CQs tailored to problem MPCC as follows.

Definition 2.1 The MPCC-MFCQ-T holds at a feasible point $\bar{\boldsymbol{z}}$ of the MPCC if the corresponding tightened program (TNLP($\bar{\boldsymbol{z}}$)) satisfies the MFCQ at the same point $\bar{\boldsymbol{z}}$. Equivalently, the MPCC-MFCQ-T holds at a feasible point $\bar{\boldsymbol{z}}$ iff

$$\left\{\nabla h_i(\bar{\boldsymbol{z}}) : i \in I^h\right\} \cup \left\{\nabla G_i(\bar{\boldsymbol{z}}) : i \in I^G(\bar{\boldsymbol{z}}) \cup I^{GH}(\bar{\boldsymbol{z}})\right\} \cup \left\{\nabla H_i(\bar{\boldsymbol{z}}) : i \in I^H(\bar{\boldsymbol{z}}) \cup I^{GH}(\bar{\boldsymbol{z}})\right\}$$

is linearly independent and

$$\exists\, \boldsymbol{d} \in \mathbb{R}^n : \quad \begin{cases} \nabla h_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} = 0 & \forall i \in I^h, \\ \nabla G_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} = 0 & \forall i \in I^G(\bar{\boldsymbol{z}}) \cup I^{GH}(\bar{\boldsymbol{z}}), \\ \nabla H_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} = 0 & \forall i \in I^H(\bar{\boldsymbol{z}}) \cup I^{GH}(\bar{\boldsymbol{z}}), \\ \nabla g_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} > 0 & \forall i \in I^g(\bar{\boldsymbol{z}}). \end{cases}$$

Definition 2.1 has been widely used in the literature (see, e.g., [6, Definition 3.1], [14, Definition 4], [8, p. 4] and [9, Definition 5.4]) and is usually referred to as the MPCC-MFCQ. However, we denote this as MPCC-MFCQ-T to emphasise the fact that it is based on the tightened program. In addition, this avoids conflict with the alternative definition that we will present next.

An alternative MPCC-tailored MFCQ relies on the Relaxed Nonlinear Program (RNLP($\bar{\boldsymbol{z}}$)) defined, for a given feasible point $\bar{\boldsymbol{z}}$ of problem MPCC, as

$$\begin{array}{lllll} \underset{\boldsymbol{z}\in\mathbb{R}^n}{\text{minimise}} & f(\boldsymbol{z}) & & & \\ \text{subject to} & g_i(\boldsymbol{z}) \geq 0 & & & \text{for } i = 1, \ldots, p, \\ & h_i(\boldsymbol{z}) = 0 & & & \text{for } i = 1, \ldots, q, \\ & G_i(\boldsymbol{z}) = 0, & H_i(\boldsymbol{z}) \geq 0 & & \text{for } i \in I^G(\bar{\boldsymbol{z}}), \\ & G_i(\boldsymbol{z}) \geq 0, & H_i(\boldsymbol{z}) = 0 & & \text{for } i \in I^H(\bar{\boldsymbol{z}}), \\ & G_i(\boldsymbol{z}) \geq 0, & H_i(\boldsymbol{z}) \geq 0 & & \text{for } i \in I^{GH}(\bar{\boldsymbol{z}}). \end{array} \tag{RNLP($\bar{\boldsymbol{z}}$)}$$

This problem is similar to the tightened program but for the biactive indices $i \in I^{GH}(\bar{\boldsymbol{z}})$, where inequality constraints rather than equality constraints are added for G_i and H_i. Analogously, we use the notation (RNLP($\bar{\boldsymbol{z}}$)) to make clear that this program is defined at a feasible point $\bar{\boldsymbol{z}}$. This leads to a second, different MPCC-tailored constraint qualification.

Definition 2.2 The MPCC-MFCQ-R holds at a feasible point $\bar{\boldsymbol{z}}$ of the MPCC iff the corresponding relaxed program (RNLP($\bar{\boldsymbol{z}}$)) satisfies the MFCQ at the same point $\bar{\boldsymbol{z}}$. Equivalently, the MPCC-MFCQ-R holds at a feasible point $\bar{\boldsymbol{z}}$ of the MPCC iff

$$\Big\{\nabla h_i(\bar{\boldsymbol{z}}) : i \in I^h\Big\} \cup \Big\{\nabla G_i(\bar{\boldsymbol{z}}) : i \in I^G(\bar{\boldsymbol{z}})\Big\} \cup \Big\{\nabla H_i(\bar{\boldsymbol{z}}) : i \in I^H(\bar{\boldsymbol{z}})\Big\}$$
$$\text{is linearly independent and} \tag{2.2}$$
$$\exists\, \boldsymbol{d} \in \mathbb{R}^n : \begin{cases} \nabla h_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} = 0 & \forall i \in I^h, \\ \nabla G_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} = 0 & \forall i \in I^G(\bar{\boldsymbol{z}}), \\ \nabla H_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} = 0 & \forall i \in I^H(\bar{\boldsymbol{z}}), \\ \nabla g_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} > 0 & \forall i \in I^g(\bar{\boldsymbol{z}}), \\ \nabla G_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} > 0,\ \nabla H_i(\bar{\boldsymbol{z}})^\top \boldsymbol{d} > 0 & \forall i \in I^{GH}(\bar{\boldsymbol{z}}). \end{cases} \tag{2.3}$$

Definition 2.2 is less common. It can be found in [15, Definition 2.5] and [16, Definition 2.5]. These papers call it *MPCC-MFCQ* as well, but to avoid confusion, we will call it MPCC-MFCQ-R, denoting that it is based on the relaxed program. We next provide the statement of MPCC-LICQ.

Remark 2.3 The MPCC-LICQ holds at point $\bar{\boldsymbol{z}}$ of problem MPCC iff any of the following three equivalent statements holds:

(i) The corresponding tightened program (TNLP($\bar{\boldsymbol{z}}$)) satisfies the LICQ at the same point $\bar{\boldsymbol{z}}$.

(ii) The corresponding relaxed program (RNLP($\bar{\boldsymbol{z}}$)) satisfies the LICQ at the same point $\bar{\boldsymbol{z}}$.

(iii) The following family of vectors is linearly independent:

$$\begin{aligned} \Big\{\nabla h_i(\bar{\boldsymbol{z}}) : i \in I^h\Big\} \cup \Big\{\nabla g_i(\bar{\boldsymbol{z}}) : i \in I^g(\bar{\boldsymbol{z}})\Big\} \cup \Big\{\nabla G_i(\bar{\boldsymbol{z}}) : i \in I^G(\bar{\boldsymbol{z}}) \cup I^{GH}(\bar{\boldsymbol{z}})\Big\} \\ \cup \Big\{\nabla H_i(\bar{\boldsymbol{z}}) : i \in I^H(\bar{\boldsymbol{z}}) \cup I^{GH}(\bar{\boldsymbol{z}})\Big\}. \end{aligned} \tag{2.4}$$

This means that the concept of the MPCC-LICQ is the same regardless of whether it is defined from the tightened or relaxed program.

Observe that the MPCC-MFCQ-T and MPCC-MFCQ-R are defined above in primal space due to the involvement of feasible directions. Next, we introduce the dual forms of these CQs, which are usually more handy in practice. To proceed, consider two sets of vectors $A := \{a_i : i \in I_1\}$ and

$B \coloneqq \{b_i : i \in I_2\}$. We say that (A, B) is *positive-linearly dependent* iff there exist scalars $\{\lambda_i\}_{i\in I_1}$ and $\{\mu_i\}_{i\in I_2}$, not all of them zero, such that

$$\sum_{i\in I_1} \lambda_i a_i + \sum_{i\in I_2} \mu_i b_i = 0, \text{ where } \lambda_i \geq 0 \text{ for } i \in I_1 \text{ and } \mu_i \text{ free for } i \in I_2.$$

If no such scalars exist, we say (A, B) is *positive-linearly independent*. The concept of positive-linear independence, which appears to have been introduced in [47, p. 965, Definition 2.1], has since been widely used, in particular in the MPCC literature; see, e.g., [20, p. 260, Definition 2.1] and [9, page 63, Definition 4.5]. We now state a lemma that will be useful for many of the proofs below.

Lemma 2.4 *The following statements hold true:*

(i) The MFCQ holds at a feasible point $\bar{\boldsymbol{z}}$ of problem (NLP) if and only if (A, B) is positive-linearly independent where:

$$A = \{\nabla g_i(\bar{\boldsymbol{z}}) : i \in I^g(\bar{\boldsymbol{z}})\}, \qquad B = \{\nabla h_i(\bar{\boldsymbol{z}}) : i \in I^h\}.$$

(ii) The MPCC-MFCQ-T is satisfied at a feasible point $\bar{\boldsymbol{z}}$ of problem MPCC if and only if (A,B) is positive-linearly independent where:

$$\begin{aligned} A &= \{\nabla g_i(\bar{\boldsymbol{z}}) : i \in I^g(\bar{\boldsymbol{z}})\}, \\ B &= \left\{\nabla h_i(\bar{\boldsymbol{z}}) : i \in I^h\right\} \cup \left\{\nabla G_i(\bar{\boldsymbol{z}}) : i \in I^G(\bar{\boldsymbol{z}}) \cup I^{GH}(\bar{\boldsymbol{z}})\right\} \cup \left\{\nabla H_i(\bar{\boldsymbol{z}}) : i \in I^H(\bar{\boldsymbol{z}}) \cup I^{GH}(\bar{\boldsymbol{z}})\right\}. \end{aligned} \tag{2.5}$$

(iii) The MPCC-MFCQ-R is satisfied at a feasible point $\bar{\boldsymbol{z}}$ of problem MPCC if and only if (A, B) is positive-linearly independent where:

$$\begin{aligned} A &= \{\nabla g_i(\bar{\boldsymbol{z}}) : i \in I^g(\bar{\boldsymbol{z}})\} \cup \left\{\nabla G_i(\bar{\boldsymbol{z}}), \nabla H_i(\bar{\boldsymbol{z}}) : i \in I^{GH}(\bar{\boldsymbol{z}})\right\}, \\ B &= \left\{\nabla h_i(\bar{\boldsymbol{z}}) : i \in I^h\right\} \cup \left\{\nabla G_i(\bar{\boldsymbol{z}}) : i \in I^G(\bar{\boldsymbol{z}})\right\} \cup \left\{\nabla H_i(\bar{\boldsymbol{z}}) : i \in I^H(\bar{\boldsymbol{z}})\right\}. \end{aligned} \tag{2.6}$$

Proof This follows directly from applying Motzkin's theorem of the alternative [48, p. 27] to the definitions of MFCQ, MPCC-MFCQ-T and MPCC-MFCQ-R for the corresponding problems. □

We have the following sequence of implications, where the first is well known. The second implication is presented for the first time. We include both proofs for completeness.

Theorem 2.5 *For any feasible point $\boldsymbol{z}$ of problem MPCC, the following implications hold:*

$$\textit{MPCC-LICQ} \implies \textit{MPCC-MFCQ-T} \implies \textit{MPCC-MFCQ-R}$$

Proof For the first implication see [6, Corollary 3.2]. For the second implication, assume that the MPCC-MFCQ-R does not hold at $\bar{\boldsymbol{z}}$. By Lemma 2.4, we can conclude the positive-linear dependence of (2.6). That is there exist scalars $\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\nu}}, \hat{\boldsymbol{\xi}} \geq 0$ and $\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi}$ free, not all of them zero, such that

$$\begin{aligned} &\sum_{i\in I^g(\bar{\boldsymbol{z}})} \hat{\lambda}_i \nabla g_i(\bar{\boldsymbol{z}}) + \sum_{i\in I^{GH}(\bar{\boldsymbol{z}})} \hat{\nu}_i \nabla G_i(\bar{\boldsymbol{z}}) + \sum_{i\in I^{GH}(\bar{\boldsymbol{z}})} \hat{\xi}_i \nabla H_i(\bar{\boldsymbol{z}}) \\ &+ \sum_{i=1,\ldots,q} \mu_i \nabla h_i(\bar{\boldsymbol{z}}) + \sum_{i\in I^G(\bar{\boldsymbol{z}})} \nu_i \nabla G_i(\bar{\boldsymbol{z}}) + \sum_{i\in I^H(\bar{\boldsymbol{z}})} \xi_i \nabla H_i(\bar{\boldsymbol{z}}) = 0. \end{aligned}$$

This contradicts the positive-linear independence of 2.5. Therefore, MPCC-MFCQ-T also fails. □

MPCC-MFCQ-R is strictly weaker than MPCC-MFCQ-T as shown in the following example.

Example 2.6 Consider the following (MPCC):

$$\begin{aligned} \underset{\boldsymbol{z}\in\mathbb{R}^2}{\text{minimise}} \quad & z_1 + z_2 \\ \text{subject to} \quad & z_1 + z_2 \geq 0, \\ & 0 \leq z_1 \perp z_2 \geq 0 \end{aligned}$$

with $g(\boldsymbol{z}) \coloneqq z_1 + z_2$, $G(\boldsymbol{z}) \coloneqq z_1$ and $H(\boldsymbol{z}) \coloneqq z_2$. At the point $\bar{\boldsymbol{z}} = [0,0]^\top$, we are in the bi-active case where $G(\bar{\boldsymbol{z}}) = H(\bar{\boldsymbol{z}}) = 0$. Consider the derivatives of the constraints: $\nabla g(\bar{\boldsymbol{z}}) = [1,1]^\top$, $\nabla G(\bar{\boldsymbol{z}}) = [1,0]^\top$ and $\nabla H(\bar{\boldsymbol{z}}) = [0,1]^\top$. Choosing, for example, $d = [3,4]^\top$, we get $\boldsymbol{d}^\top \nabla g(\bar{\boldsymbol{z}}) = 7 > 0$ and $\boldsymbol{d}^\top \nabla G(\bar{\boldsymbol{z}}) = 3 > 0$ and $\boldsymbol{d}^\top \nabla H(\bar{\boldsymbol{z}}) = 4 > 0$. Thus, the MPCC-MFCQ-R holds at $\bar{\boldsymbol{z}}$. However, there does not exist any direction $\boldsymbol{d} \in \mathbb{R}^2$ such that $\boldsymbol{d}^\top \nabla g(\bar{\boldsymbol{z}}) > 0$ and $\boldsymbol{d}^\top \nabla G(\bar{\boldsymbol{z}}) = 0$ and $\boldsymbol{d}^\top \nabla H(\bar{\boldsymbol{z}}) = 0$. This implies the failure of the MPCC-MFCQ-T at $\bar{\boldsymbol{z}}$.

Another important example that satisfies MPCC-MFCQ-R but not MPCC-MFCQ-T is the main topic of Section 4. For further evidence of the very strong nature of the MPCC-MFCQ-T, we end on the following remark.

Remark 2.7 For any problem of the form (MPCC) with $p = 0$ (in other words with no inequality $g_i(\boldsymbol{z}) \geq 0$ constraints but still possibly with equality $h_i(\boldsymbol{z}) = 0$ and complementarity $0 \leq H_i(\boldsymbol{z}) \perp G_i(\boldsymbol{z}) \geq 0$ constraints), the MPCC-MFCQ-T is equivalent to the MPCC-LICQ.

2.2 Stationarity

This subsection states some well-known stationarity concepts for problem MPCC, which have been extensively studied in the literature; see, e.g. [8], which seems to be one of the early papers on the subject. We begin by introducing a problem-specific Lagrangian function

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi}) \coloneqq f(\boldsymbol{z}) - \sum_{i=1}^{p} \lambda_i g_i(\boldsymbol{z}) - \sum_{i=1}^{q} \mu_i h_i(\boldsymbol{z}) - \sum_{i=1}^{r} \nu_i G_i(\boldsymbol{z}) - \sum_{i=1}^{r} \xi_i H_i(\boldsymbol{z}). \tag{2.7}$$

Observe that this function does not include the product terms $G_i(\boldsymbol{z})H_i(\boldsymbol{z})$ for $i = 1, \ldots, r$.

Definition 2.8 The feasible point $\bar{\boldsymbol{z}}$ of problem MPCC is said to be Weakly (W)-stationary if there exist multipliers $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi})$ such that the following conditions hold:

$$\nabla_{\boldsymbol{z}} \mathcal{L}(\bar{\boldsymbol{z}}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi}) = 0 \tag{2.8a}$$

$$\lambda_i \geq 0 \text{ for } i \in I^g(\bar{\boldsymbol{z}}), \quad \lambda_i = 0 \text{ for } i \notin I^g(\bar{\boldsymbol{z}}), \quad \xi_i = 0 \text{ for } i \in I^G(\bar{\boldsymbol{z}}), \quad \nu_i = 0 \text{ for } i \in I^H(\bar{\boldsymbol{z}}). \tag{2.8b}$$

The point $\bar{\boldsymbol{z}}$ will be said to be Alternative (A), Clarke (C), Mordukhovich (M) or Strongly (S)-stationary if additionally to (2.8b), we respectively have the conditions

$$\text{A-stationary:} \quad \nu_i \geq 0 \text{ or } \xi_i \geq 0 \quad \text{for } i \in I^{GH}(\bar{\boldsymbol{z}}); \tag{2.8A}$$

$$\text{C-stationary:} \quad \nu_i \xi_i \geq 0 \quad \text{for } i \in I^{GH}(\bar{\boldsymbol{z}}); \tag{2.8C}$$

$$\text{M-stationary:} \quad \nu_i, \xi_i \geq 0 \text{ or } \nu_i \xi_i = 0 \quad \text{for } i \in I^{GH}(\bar{\boldsymbol{z}}); \tag{2.8M}$$

$$\text{S-stationary:} \quad \nu_i, \xi_i \geq 0 \quad \text{for } i \in I^{GH}(\bar{\boldsymbol{z}}). \tag{2.8S}$$

It is important to note a few properties of these stationary conditions. They only differ by the nature of multipliers on the biactive index set I^{GH}. Figure SM9 highlights these differences. So, if the problem is strictly complementary at $\bar{\boldsymbol{z}}$, that is $G_i(\boldsymbol{z}) > 0, H_i(\boldsymbol{z}) = 0$ or $G_i(\boldsymbol{z}) = 0, H_i(\boldsymbol{z}) > 0$ for all $i = 1, \ldots, r$, then all five concepts are equivalent. S-stationary is the strongest and is equivalent to the

KKT conditions of the MPCC if the complementarity constraints are viewed as ordinary inequality and equality constraints. The following relationships hold [6, p. 60]:

$$\text{S-stationary} \implies \text{M-stationary} \implies \begin{matrix} \text{A-stationary} \\ \text{C-stationary} \end{matrix} \implies \text{W-stationary}$$

3 Solution methods

In this section, we study the partial penalisation and Scholtes relaxation methods. There are at least two motivations to study these algorithms. First, they are among the most commonly used in the literature on MPCCs (see, e.g., [14]). Secondly, there is a nice symmetry between the two approaches. In the penalisation method, the constraint $G(\boldsymbol{z})^\top H(\boldsymbol{z}) = 0$ is moved from the feasible set to the objective function, while this same constraint is instead relaxed to form a larger feasible set in the context of the relaxation approach.

As it will be clear in the next subsection, a common point between the two methods, which does not seem to have been discovered before, is that they converge under the same CQ; i.e., the MPCC-MFCQ-R, which is strictly weaker than any CQ that has been used so far to prove the convergence of these algorithms. However, a key departure point on the outcomes of the convergence of these two algorithms is that the penalisation method will be shown to converge to a S-stationary point, while we get C-stationarity from the Scholtes relaxation method.

3.1 Penalisation

We start by recalling that the penalisation method works by moving the problematic product term $G(\boldsymbol{z})^\top H(\boldsymbol{z})$ from the constraints of problem (MPCC) to its objective function. This leads to the penalised problem with $\pi \geq 0$ being the penalty parameter:

$$\begin{aligned} \underset{\boldsymbol{z}\in\mathbb{R}^n}{\text{minimise}} \quad & f(\boldsymbol{z}) \ + \pi \ G(\boldsymbol{z})^\top H(\boldsymbol{z}) \\ \text{subject to} \quad & g_i(\boldsymbol{z}) \geq 0 && \text{for } i = 1, ..., p, \\ & h_i(\boldsymbol{z}) = 0 && \text{for } i = 1, ..., q, \\ & G_i(\boldsymbol{z}) \geq 0, \quad H_i(\boldsymbol{z}) \geq 0 && \text{for } i = 1, ..., r. \end{aligned} \tag{$\mathcal{P}_\pi$}$$

Interestingly, $(\mathcal{P}_\pi)$ is more likely to satisfy standard constraint qualifications. In fact, if MPCC satisfies MPCC-MFCQ-R at a point $\boldsymbol{z}^*$, then $(\mathcal{P}_\pi)$ will satisfy the MFCQ at $\boldsymbol{z}^*$ [16, Theorem 5.1]. This means that we can employ standard NLP solvers such as the interior-point method (IPM) or sequential quadratic programming (SQP) for $(\mathcal{P}_\pi)$. However, when employing such solvers, we should not expect them to find a global minimum but instead terminate within a certain tolerance $\epsilon \geq 0$ of a KKT point. With this in mind, we define a point $\boldsymbol{z} \in \mathbb{R}^n$ to be an ϵ-KKT point for $(\mathcal{P}_\pi)$ if there exists a Lagrange multiplier vector $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi}) \in \mathbb{R}^{(p+q+2r)}$ satisfying the following system:

$$\begin{aligned} \Big\| \nabla f(\boldsymbol{z}) + \pi H(\boldsymbol{z}) \left[\nabla G(\boldsymbol{z})\right] + \pi G(\boldsymbol{z}) \left[\nabla H(\boldsymbol{z})\right] - \sum_{i=1}^{p} \lambda_i \nabla g_i(\boldsymbol{z}) - \sum_{i=1}^{q} \mu_i \nabla h_i(\boldsymbol{z}) \\ - \sum_{i=1}^{r} \nu_i \nabla G_i(\boldsymbol{z}) - \sum_{i=1}^{r} \xi_i \nabla H_i(\boldsymbol{z}) \Big\| \leq \epsilon, \end{aligned} \tag{3.1a}$$

$$\begin{aligned} & g_i(\boldsymbol{z}) \geq 0, \quad \lambda_i \geq 0, \quad g_i(\boldsymbol{z})\lambda_i = 0, && \text{for } i = 1, \dots, p, \\ & h_i(\boldsymbol{z}) = 0, \quad \mu_i \text{ free}, && \text{for } i = 1, \dots, q, \\ & G_i(\boldsymbol{z}) \geq 0, \quad \nu_i \geq 0, \quad G_i(\boldsymbol{z})\nu_i = 0, && \text{for } i = 1, \dots, r, \\ & H_i(\boldsymbol{z}) \geq 0, \quad \xi_i \geq 0, \quad H_i(\boldsymbol{z})\xi_i = 0, && \text{for } i = 1, \dots, r. \end{aligned} \tag{3.1b}$$

This concept is widely used in the literature; see, e.g. [49, Definition 3.1]. For $\epsilon = 0$ the system (3.1a-b) is the classical KKT conditions for $(\mathcal{P}_\pi)$. The idea is then for a large enough penalty parameter π

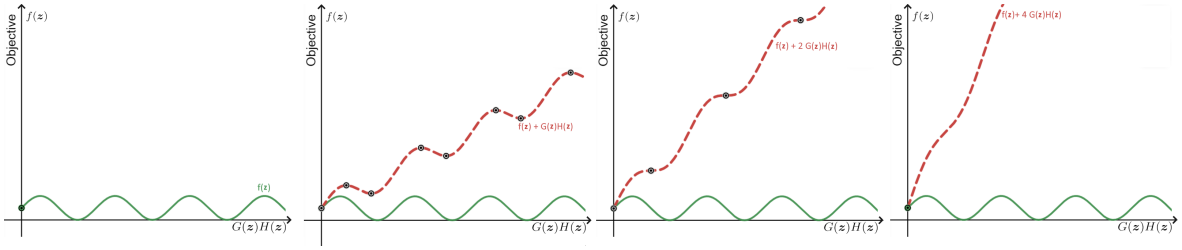

Fig. 1 This figure visualises applying the penalisation sequence to a hypothetical MPCC with a single complementarity constraint. The vertical axis is the objective value, while the horizontal axis is the product of the two complementarity constraint functions $G(z)H(z)$. The green line shows the objective value $f(z)$, and the red curve shows the penalised objective $f(z) + \pi G(z)H(z)$ for increasing penalty parameters π. The grey dots show infeasible stationary points that are slowly ironed out.

and small enough tolerance ϵ, finding a solution to (3.1a-b) that will correspond to a stationary point of problem MPCC. If we know such values for π and ϵ, we only need to solve $(\mathcal{P}_\pi)$ once. This is called *exact penalisation*. However, in practice, we may not have this information, or in the case where π is too large, $(\mathcal{P}_\pi)$ may become ill-conditioned [18, Example 3]. Therefore, a more practical approach is to introduce an increasing sequence of non-negative penalty parameters $(\pi^t)_{t\in\mathbb{N}}$ and a decreasing sequence of tolerances $(\epsilon^t)_{t\in\mathbb{N}}$ and solve problem $(\mathcal{P}_{\pi^t})$ repeatedly using the previous solution as the starting point for the next iteration until our stopping criteria are met. We call this *sequential penalisation*. A precise description of the method is provided in Algorithm 1 and its theoretical convergence is established in the result below. A graphical illustration, for increasing penalty parameter values, is given in Figure 1.

Algorithm 1 Inexact sequential partial penalisation

Input: Access to the MPCC's functions f, g, h, G, H and their first order derivatives; Starting point $\boldsymbol{z}^0 \in \mathbb{R}^n$; Initial penalty parameter $\pi^1 > 0$; Initial KKT tolerance $\epsilon^1 > 0$; Exit tolerance $\epsilon_{\text{exit}} > 0$;

Procedure:

1: **repeat** for $t = 1, 2, 3, \ldots$:
2: Find an ϵ^t-KKT point $\boldsymbol{z}^t$ of $(\mathcal{P}_{\pi^t})$ with starting point $\boldsymbol{z}^{t-1}$;
3: Increase the penalty parameter $\pi^{t+1} > \pi^t$;
4: Decrease the KKT tolerance $\epsilon^{t+1} < \epsilon^t$, ensuring $\epsilon^t \to 0$;
5: **until** Stopping condition: $\max_i \left(\min(G_i(\boldsymbol{z}^t),\ H_i(\boldsymbol{z}^t))\right) \leq \epsilon_{\text{exit}}$.
6: Output $\boldsymbol{z}^t$.

Theorem 3.1 *Consider problem MPCC and a sequence of iterates $\left(\boldsymbol{z}^t\right)_{t\in\mathbb{N}}$, generated by Algorithm 1, which is assumed to converge to the point $\boldsymbol{z}^*$ where $G(\boldsymbol{z}^*)^\top H(\boldsymbol{z}^*) = 0$, then*

(i) If MPCC-MFCQ-T holds at $\boldsymbol{z}^*$, then $\boldsymbol{z}^*$ is a C-stationary point.

(ii) If MPCC-MFCQ-R holds at $\boldsymbol{z}^*$ but additionally $\pi^t G_i(\boldsymbol{z}^t) \to 0$, $\pi^t H_i(\boldsymbol{z}^t) \to 0$, for the biactive indices $i \in I^{GH}(\boldsymbol{z}^*)$ then $\boldsymbol{z}^*$ is a S-stationary point.

Proof For each iterate $\boldsymbol{z}^t$, since it is an ϵ^t-KKT point of $(\mathcal{P}_{\pi^t})$, let $\left(\boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\nu}^t, \boldsymbol{\xi}^t\right) \in \mathbb{R}^{(p+q+2r)}$ be a Lagrange multiplier vector satisfying (3.1a-b). We now define augmented multipliers:

$$\hat{\nu}_i^t \coloneqq \nu_i^t - \pi^t H_i(\boldsymbol{z}^t) \quad \text{and} \quad \hat{\xi}_i^t \coloneqq \xi_i^t - \pi^t G_i(\boldsymbol{z}^t). \tag{3.2}$$

In this way we can rewrite (3.1a) as

$$\left\| \nabla f(\boldsymbol{z}^t) - \sum_{i=1}^{p} \lambda_i^t \nabla g_i(\boldsymbol{z}^t) - \sum_{i=1}^{q} \mu_i^t \nabla h_i(\boldsymbol{z}^t) - \sum_{i=1}^{r} \hat{\nu}_i^t \nabla G_i(\boldsymbol{z}^t) - \sum_{i=1}^{r} \hat{\xi}_i^t \nabla H_i(\boldsymbol{z}^t) \right\| \leq \epsilon^t. \tag{3.3}$$

We wish to prove that the sequence $(\boldsymbol{a}_t)_{t\in\mathbb{N}}$, where $\boldsymbol{a}_t := \left(\boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \hat{\boldsymbol{\nu}}^t, \hat{\boldsymbol{\xi}}^t\right)$, has a convergent subsequence. Suppose towards a contradiction that it does not. Then $\|\boldsymbol{a}_t\| \to \infty$; for otherwise some subsequence of $(\boldsymbol{a}_t)_{t\in\mathbb{N}}$ would be bounded and would, by the Bolzano–Weierstrass theorem, admit a convergent sub-subsequence. Now, dividing equation (3.3) through by $\|\boldsymbol{a}_t\|$ leads to

$$\left\| \frac{\nabla f(\boldsymbol{z}^t)}{\|\boldsymbol{a}_t\|} - \sum_{i=1}^{p} \frac{\lambda_i^t}{\|\boldsymbol{a}_t\|} \nabla g_i(\boldsymbol{z}^t) - \sum_{i=1}^{q} \frac{\mu_i^t}{\|\boldsymbol{a}_t\|} \nabla h_i(\boldsymbol{z}^t) - \sum_{i=1}^{r} \frac{\hat{\nu}_i^t}{\|\boldsymbol{a}_t\|} \nabla G_i(\boldsymbol{z}^t) - \sum_{i=1}^{r} \frac{\hat{\xi}_i^t}{\|\boldsymbol{a}_t\|} \nabla H_i(\boldsymbol{z}^t) \right\| \leq \frac{\epsilon^t}{\|\boldsymbol{a}_t\|}. \quad (3.4)$$

We consider the normalised sequence of Lagrange multipliers $\boldsymbol{a}_t / \|\boldsymbol{a}_t\|$. As it is bounded, by the Bolzano–Weierstrass theorem, there exists some infinite subsequence that converges. Let $(\boldsymbol{\lambda}', \boldsymbol{\mu}', \hat{\boldsymbol{\nu}}', \hat{\boldsymbol{\xi}}')$ be the limit of that subsequence. Since every term of the normalised sequence has unit norm, so does its limit, by continuity of the norm; in particular $(\boldsymbol{\lambda}', \boldsymbol{\mu}', \hat{\boldsymbol{\nu}}', \hat{\boldsymbol{\xi}}') \neq 0$. We next identify the components that vanish in the limit. For $i \in \{1, \dots, p\} \setminus I^g$ we have $g_i(\boldsymbol{z}^*) > 0$, hence $g_i(\boldsymbol{z}^t) > 0$ for all large t, and the complementarity slackness in (3.1b) gives $\lambda_i^t = 0$; therefore $\lambda_i' = 0$. Now consider $i \in I^H$, so that $G_i(\boldsymbol{z}^*) > 0$ and $H_i(\boldsymbol{z}^*) = 0$. Then $G_i(\boldsymbol{z}^t) > 0$ for all large t, so complementarity slackness gives $\nu_i^t = 0$ and hence $\hat{\nu}_i^t = -\pi^t H_i(\boldsymbol{z}^t)$ by (3.2). If $H_i(\boldsymbol{z}^t) = 0$, then $\hat{\nu}_i^t = 0$. Otherwise, complementarity slackness gives $\xi_i^t = 0$, so that $\hat{\xi}_i^t = -\pi^t G_i(\boldsymbol{z}^t) \neq 0$, and since $|\hat{\xi}_i^t| \leq \|\boldsymbol{a}_t\|$,

$$\frac{|\hat{\nu}_i^t|}{\|\boldsymbol{a}_t\|} \leq \frac{|\hat{\nu}_i^t|}{|\hat{\xi}_i^t|} = \frac{\pi^t H_i(\boldsymbol{z}^t)}{\pi^t G_i(\boldsymbol{z}^t)} = \frac{H_i(\boldsymbol{z}^t)}{G_i(\boldsymbol{z}^t)} \longrightarrow \frac{0}{G_i(\boldsymbol{z}^*)} = 0.$$

In either case $\hat{\nu}_i' = 0$ for $i \in I^H$; exchanging the roles of G and H gives $\hat{\xi}_i' = 0$ for $i \in I^G$. Using this indexing, we can say that it must be the $\left(\lambda'_{I^g}, \boldsymbol{\mu}', \hat{\nu}'_{I^{G}\cup I^{GH}}, \hat{\xi}'_{I^{H}\cup I^{GH}}\right)$ components of the sequence that are non-zero. Now, returning to (3.4), as $t \to \infty$, the terms $\frac{\nabla f(\boldsymbol{z}^t)}{\|\boldsymbol{a}_t\|}$ and $\frac{\epsilon^t}{\|\boldsymbol{a}_t\|}$ go to zero, leaving the following linear combination equal to zero

$$\sum_{i\in I^g} \lambda_i' \nabla g_i(\boldsymbol{z}^*) + \sum_{i\in I^h} \mu_i' \nabla h_i(\boldsymbol{z}^*) + \sum_{i\in I^G\cup I^{GH}} \hat{\nu}_i' \nabla G_i(\boldsymbol{z}^*) + \sum_{i\in I^H\cup I^{GH}} \hat{\xi}_i' \nabla H_i(\boldsymbol{z}^*) = 0. \quad (3.5)$$

Under the assumptions of part (i), since $\lambda_i' \geq 0$, the sets of gradients given in (2.5) are positive-linearly dependent, contradicting MPCC-MFCQ-T. Under the assumptions of part (ii), i.e. $\pi^t G_i(\boldsymbol{z}^t) \to 0$ and $\pi^t H_i(\boldsymbol{z}^t) \to 0$ for $i \in I^{GH}$, we have that $\hat{\nu}_i' \geq 0$ and $\hat{\xi}_i' \geq 0$ for $i \in I^{GH}$, so that the sets of gradients given in (2.6) are positive-linearly dependent, contradicting MPCC-MFCQ-R. By these contradictions, we conclude in both cases (i) and (ii) that $(\boldsymbol{a}_t)_{t\in\mathbb{N}}$ has a convergent subsequence. Let $\left(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \hat{\boldsymbol{\nu}}^*, \hat{\boldsymbol{\xi}}^*\right)$ be the limit of that convergent subsequence. By the continuous differentiability of f, g, h, G, H, and since the sequence $\left(\boldsymbol{z}^t\right)_{t\in\mathbb{N}}$ converges to $\boldsymbol{z}^*$ in their domain, these functions and their derivatives must converge in range. Applying this to inequality (3.3), and recalling that $\epsilon^t \to 0$, we get

$$\nabla f(\boldsymbol{z}^*) - \sum_{i=1}^{p} \lambda_i^* \nabla g_i(\boldsymbol{z}^*) - \sum_{i=1}^{q} \mu_i^* \nabla h_i(\boldsymbol{z}^*) - \sum_{i=1}^{r} \hat{\nu}_i^* \nabla G_i(\boldsymbol{z}^*) - \sum_{i=1}^{r} \hat{\xi}_i^* \nabla H_i(\boldsymbol{z}^*) = 0. \quad (3.6)$$

satisfying the stationarity condition (2.8a) of Definition 2.8. To establish W-stationarity, it remains to be shown that condition (2.8b) is satisfied, i.e., $\hat{\nu}_i^* = 0$ for each $i \in I^H$ and $\hat{\xi}_i^* = 0$ for each $i \in I^G$. Considering an index $i \in I^G$, by definition, we have $G_i(\boldsymbol{z}^*) = 0$ and $H_i(\boldsymbol{z}^*) > 0$. Since $H_i(\boldsymbol{z}^t) \to H_i(\boldsymbol{z}^*) > 0$, complementarity slackness gives $\xi_i^t = 0$ for all large t, and hence $\xi_i^* = 0$. To deduce $\hat{\xi}_i^* = 0$ from (3.2), the penalty term $\pi^t G_i(\boldsymbol{z}^t)$ must vanish in the limit. For every t with $G_i(\boldsymbol{z}^t) > 0$, complementarity slackness forces $\nu_i^t = 0$, so that $\hat{\nu}_i^t = -\pi^t H_i(\boldsymbol{z}^t)$; since $\hat{\nu}_i^t$ converges to the finite limit $\hat{\nu}_i^*$ while $H_i(\boldsymbol{z}^t)$ is bounded away from zero, the penalty parameter π^t remains bounded for all such t. Consequently $\pi^t G_i(\boldsymbol{z}^t) \to 0$, because for each t either $G_i(\boldsymbol{z}^t) = 0$ or π^t is bounded while $G_i(\boldsymbol{z}^t) \to 0$; therefore $\hat{\xi}_i^* = 0$ for all $i \in I^G$. Exchanging the roles of G and H, the same argument gives $\hat{\nu}_i^* = 0$ for all $i \in I^H$.

To establish C-stationarity (2.8C) under the assumptions of Theorem 3.1 (i), consider a biactive index $i \in I^{GH}$ and the product $\hat{\nu}_i^t \hat{\xi}_i^t$. Expanding it and cancelling the two cross terms by the complementarity slackness $\nu_i^t G_i(\boldsymbol{z}^t) = 0$ and $\xi_i^t H_i(\boldsymbol{z}^t) = 0$ of (3.1b), we obtain, for every $t \in \mathbb{N}$,

$$\hat{\nu}_i^t \, \hat{\xi}_i^t = \left(\nu_i^t - \pi^t H_i(\boldsymbol{z}^t)\right)\left(\xi_i^t - \pi^t G_i(\boldsymbol{z}^t)\right) = \nu_i^t \, \xi_i^t + \left(\pi^t\right)^2 G_i(\boldsymbol{z}^t) \, H_i(\boldsymbol{z}^t) \;\geq\; 0.$$

To establish S-stationarity (2.8S) under the assumptions of Theorem 3.1 (ii), consider a biactive index $i \in I^{GH}$. Under the assumption that $\pi^t G_i(\boldsymbol{z}^t) \to 0$ and $\pi^t H_i(\boldsymbol{z}^t) \to 0$, we establish that $\liminf_{t\to\infty} \hat{\nu}_i^t \geq 0$ and $\liminf_{t\to\infty} \hat{\xi}_i^t \geq 0$ and thus $\hat{\nu}_i^* \geq 0$ and $\hat{\xi}_i^* \geq 0$, respectively. □

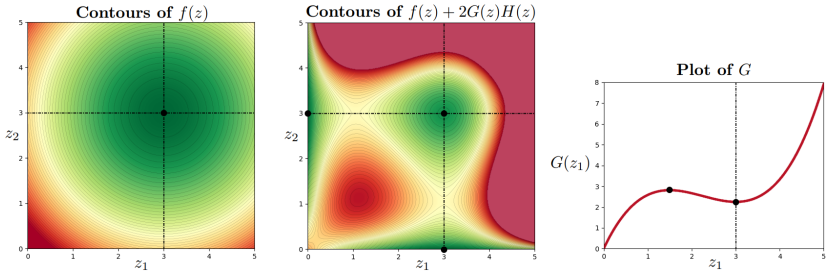

Fig. 2 A graphical representation of Example 3.3. On the left are the contours of the objective function $f(\boldsymbol{z})$. When the complementarity constraints $0 \leq G(\boldsymbol{z}) \perp H(\boldsymbol{z}) \geq 0$ are added, the feasible set becomes the positive vertical and horizontal axes. In the centre are the contours of the penalised objective function $f(\boldsymbol{z}) + \pi G(\boldsymbol{z})H(\boldsymbol{z})$. The two dark green wells at (0,3) and (3,0) are the two global minima. The light green well at (3,3) is a local minimum. On the right is the constraint $G(\boldsymbol{z})$ plotted against z_1.

Recall that Ralph and Wright [16, Theorem 5.2] provide a nice proof for penalisation with a fixed parameter π. However, in practice, a user would not know a good value for π ahead of time. If the chosen π is too small, the property $G(\boldsymbol{z}^*)^\top H(\boldsymbol{z}^*) = 0$ may not hold. On the other hand, if it is too large, the program may become ill-conditioned [18, Example 3] as is common with penalisation-based algorithms [50, 51]. This provides a motivation for the sequential approach. Also, observe that the paper by Leyffer et al. [18, Theorem 3.4] already presents a sequential penalisation theory in the case where the subproblem $(\mathcal{P}_\pi)$ is specifically solved with the interior point algorithm. Theorem 3.1 provides two useful generalisations over their work. Firstly, it gives flexibility to the user in the selection of the algorithm for the penalised subproblem $(\mathcal{P}_\pi)$. Secondly, we only assume MPCC-MFCQ-R, which is a far weaker CQ compared to the MPCC-LICQ, required in [18]. It is also important to note all the aforementioned convergence results for penalisation, including ours, require the fulfilment of firstly $G(\boldsymbol{z}^*)^\top H(\boldsymbol{z}^*) = 0$, and secondly $\pi^t G_i(\boldsymbol{z}^t) \to 0$ and $\pi^t H_i(\boldsymbol{z}^t) \to 0$ for biactive indices $i \in I^{GH}$. If an exact penalty parameter π exists, e.g. as in [16], then the latter assumption is automatically satisfied. In this case we have shown you can weaken the MPCC-MFCQ-T assumption to MPCC-MFCQ-R. However, it is unclear how to ensure that $G(\boldsymbol{z}^*)^\top H(\boldsymbol{z}^*) = 0$ in practice. In the remainder of this section, we provide Theorem 3.2 and Example 3.3 to establish when this assumption may and may not be reasonable.

Theorem 3.2 *Let $(\boldsymbol{z}^t)_{t\in\mathbb{N}}$ be a sequence of iterates generated by Algorithm 1 such that $\boldsymbol{z}^t$ is a global optimal solution of problem $(\mathcal{P}_{\pi^t})$ for $t \in \mathbb{N}$. If $\lim_{t\to\infty} \boldsymbol{z}^t = \boldsymbol{z}^*$, then $G(\boldsymbol{z}^*)^\top H(\boldsymbol{z}^*) = 0$.*

The proof is given in supplementary material Subsection SM3.1.

Next, we provide an example, which shows when the assumption that $G(\boldsymbol{z}^*)^\top H(\boldsymbol{z}^*) = 0$ fails. In this example, the objective and constraint functions $f, G, H : \mathbb{R}^2 \to \mathbb{R}$ are all continuously differentiable, and the MPCC-LICQ is satisfied. Nevertheless, its corresponding penalisation formulation has a sequence of local minima that converge to an infeasible point for the corresponding MPCC.

Example 3.3 Consider the following example of MPCC:

$$\begin{aligned} \underset{z_1,z_2}{\text{minimise}} \quad & f(\boldsymbol{z}) \coloneqq (z_1-3)^2 + (z_2-3)^2 \\ \text{subject to} \quad & G(\boldsymbol{z}) \coloneqq \left(\frac{1}{3}z_1^3 - \frac{9}{4}z_1^2 + \frac{9}{2}z_1\right) \geq 0, \\ & H(\boldsymbol{z}) \coloneqq \left(\frac{1}{3}z_2^3 - \frac{9}{4}z_2^2 + \frac{9}{2}z_2\right) \geq 0, \\ & G(\boldsymbol{z})H(\boldsymbol{z}) = 0. \end{aligned} \tag{3.7}$$

It is helpful to note that G or H are only equal to zero for $z_1 = 0$ or $z_2 = 0$, respectively. The feasible set is, therefore, the union of the two non-negative axes; see Figure 2. We begin by writing the derivatives of all the functions involved in the above problem:

$$\nabla f(\boldsymbol{z}) = \begin{bmatrix} 2z_1 - 6 \\ 2z_2 - 6 \end{bmatrix}, \qquad \nabla G(\boldsymbol{z}) = \begin{bmatrix} (z_1 - \frac{3}{2})(z_1 - 3) \\ 0 \end{bmatrix}, \qquad \nabla H(\boldsymbol{z}) = \begin{bmatrix} 0 \\ (z_2 - \frac{3}{2})(z_2 - 3) \end{bmatrix}.$$

Program (3.7) has two local minima at $[3, 0]^\top$ and $[0, 3]^\top$. Furthermore, it satisfies MPCC-LICQ at all feasible points. This can be verified by checking the two strictly complementary cases of $(G(\boldsymbol{z}) = 0, H(\boldsymbol{z}) > 0)$ and $(G(\boldsymbol{z}) > 0, H(\boldsymbol{z}) = 0)$ and the biactive case $(G(\boldsymbol{z}) = 0, H(\boldsymbol{z}) = 0)$. For each of these, in order, the set of gradients of the active constraints are $\left\{ \left[\frac{9}{2}, 0\right]^\top \right\}$ and $\left\{ \left[0, \frac{9}{2}\right]^\top \right\}$ and $\left\{ \left[\frac{9}{2}, 0\right]^\top, \left[0, \frac{9}{2}\right]^\top \right\}$ which are each linearly independent sets.

Now consider the penalisation formulation $(\mathcal{P}_\pi)$ of this example whose feasible set is the whole non-negative quadrant. Note (3.7) has only complementarity constraint functions G, H and no inequality g nor equality h constraint functions. We introduce multipliers ν and ξ corresponding to G and H respectively and write the KKT conditions of the corresponding problem $(\mathcal{P}_\pi)$:

$$\begin{aligned} &0 = \nabla \left[f(\boldsymbol{z}) + \pi^t G(\boldsymbol{z}) H(\boldsymbol{z}) \right] - \nu \nabla G(\boldsymbol{z}) - \xi \nabla H(\boldsymbol{z}), \\ &G(\boldsymbol{z}) \geq 0, \quad H(\boldsymbol{z}) \geq 0, \quad \nu \geq 0, \quad \xi \geq 0, \quad G(\boldsymbol{z})\nu = 0, \quad H(\boldsymbol{z})\xi = 0. \end{aligned} \tag{3.8}$$

Consider the point $(z_1^*, z_2^*) = (3, 3)$, feasible for the corresponding version of problem $(\mathcal{P}_\pi)$ for our example but not for problem (3.7) itself. At this point $G(\boldsymbol{z}^*) = H(\boldsymbol{z}^*) = 2.25$ and all the derivatives are zero. $\nabla f(\boldsymbol{z}^*) = \nabla G(\boldsymbol{z}^*) = \nabla H(\boldsymbol{z}^*) = 0$. Substituting these values into the KKT conditions for the penalisation problem (3.8) we can find that $(z_1^*, z_2^*, \nu, \xi) = (3, 3, 0, 0)$ is a KKT point of the penalisation problem $(\mathcal{P}_{\pi^t})$ for all parameters $\pi^t \geq 0$. Algorithm 1 could reasonably find the sequence of iterates $\boldsymbol{z}^t = (3, 3)$ for each $t \in \mathbb{N}$. However, this converges to a point that is infeasible for the original MPCC. This highlights a key weakness of the penalisation method.

3.2 Relaxation

Scholtes [11] proposed the first regularisation scheme where problem MPCC is approximated by a sequence of parametrised nonlinear programs. Since then, many alternative relaxation schemes have been proposed (e.g., [19, 22]), and a good survey on the subject is given by Hoheisel et al. [20]. Typically, in relaxation schemes for MPCCs, the constraint involving the problematic product term $G(\boldsymbol{z})^\top H(\boldsymbol{z})$ of the complementarity system is relaxed. This enlarges the feasible set and can make the resulting problem more tractable. In the specific context of the Scholtes relaxation, the following subproblem is considered:

$$\begin{aligned} \underset{\boldsymbol{z}}{\text{minimise}} \quad & f(\boldsymbol{z}) \\ \text{subject to} \quad & g_i(\boldsymbol{z}) \geq 0 && \text{for } i = 1, \ldots, p, \\ & h_i(\boldsymbol{z}) = 0 && \text{for } i = 1, \ldots, q, \\ & G_i(\boldsymbol{z}) \geq 0,\ H_i(\boldsymbol{z}) \geq 0,\ G_i(\boldsymbol{z}) H_i(\boldsymbol{z}) \leq \tau && \text{for } i = 1, \ldots, r. \end{aligned} \tag{$\mathcal{R}_\tau$}$$

Note that for τ close to zero, the feasible set of subproblem $(\mathcal{R}_\tau)$ closely approximates that of MPCC. This is illustrated in Figure SM2. This forms the basis of the relaxation method. The program $(\mathcal{R}_{\tau^t})$ is solved, inexactly, multiple times with a sequence of decreasing parameters τ^t. For this, any classical NLP solver may be used. The algorithm and proof of convergence are as follows.

Algorithm 2 Inexact sequential Scholtes relaxation

Input: Access to the MPCC's functions f, g, h, G, H and their first order derivatives; Starting point $\boldsymbol{z}^0 \in \mathbb{R}^n$; Initial relaxation parameter $\tau^1 > 0$; Initial KKT tolerance $\epsilon^1 > 0$; Exit tolerance $\epsilon_{\text{exit}} > 0$;

Procedure:

1: **repeat** for $t = 1, 2, 3, \ldots$:
2: Find an ϵ^t-KKT point $\boldsymbol{z}^t$ of $(\mathcal{R}_{\tau^t})$ with starting point $\boldsymbol{z}^{t-1}$;
3: Decrease the relaxation parameter $\tau^{t+1} < \tau^t$;
4: Decrease the KKT tolerance $\epsilon^{t+1} < \epsilon^t$;
5: **until** Stopping condition: $\max_i (\min(G_i(\boldsymbol{z}^t),\ H_i(\boldsymbol{z}^t))) \leq \epsilon_{\text{exit}}$.
6: Output $\boldsymbol{z}^t$.

Theorem 3.4 *Consider problem MPCC. Let $\left(\boldsymbol{z}^t\right)_{t\in\mathbb{N}}$ be a sequence of ϵ^t-KKT points of $(\mathcal{R}_{\tau^t})$ parametrised by a strictly decreasing sequence of positive relaxation parameters $\left(\tau^t\right)_{t\in\mathbb{N}}$ and tolerances $\left(\epsilon^t\right)_{t\in\mathbb{N}}$ that both tend to zero. Let $\boldsymbol{z}^* := \lim_{t\to\infty}\left(\boldsymbol{z}^t\right)$ exist. Let MPCC-MFCQ-R hold at $\boldsymbol{z}^*$. Then, $\boldsymbol{z}^*$ is a C-stationary point for problem MPCC.*

The structure of the proof is similar to that of Theorem 3.1 and left to the supplementary materials, Subsection SM3.3, for brevity. A similar version of Theorem 3.4 was initially proven by Scholtes [11, p. 922, Section 3] but required the stronger MPCC-LICQ. An updated proof is given by Schwartz [9, p. 112, Theorem 7.3], which weakened the requirement to MPCC-MFCQ-T. We present a similar proof but weakening the CQ further to MPCC-MFCQ-R. This is a significant weakening of assumptions (cf. Theorem 2.5 and Example 2.6). Another unique feature of our result (Theorem 3.4) is that we only require an inexact KKT point to be computed at each iteration, unlike in the previous papers, where exact KKT points were used for the penalised problem. This allows the theorem to be applied to a much broader class of problems and for the numerical implementation to reflect the theory more closely.

4 Application

In this section, we construct a MPCC model to optimally select the hyperparameters of a Support Vector Machine (SVM). We then show, amongst other theoretical properties, that the model satisfies MPCC-MFCQ-R for any feasible point with positive hyperparameters but that it systematically fails MPCC-LICQ and MPCC-MFCQ-T in general.

4.1 Modeling hyperparameter tuning

A major aspect of machine learning is judging if a model is overfitted to its training data [52, p. 28], [53, Chapter 7]. For classifiers, this could mean unreasonably contorting the decision boundary to include outliers. By doing so, the classifier achieves near-zero empirical training error but misses the overarching pattern. SVMs' power comes from their ability to accept some level of empirical error in favour of a simpler boundary that generalises well to unseen data. These two objectives are balanced by the hyperparameters. This leads to the question of how to evaluate the accuracy of a model or the choice of hyperparameters if training error does not provide the whole picture.

The cross-validation procedure (see literature [53, Section 7.10.1], [52, p. 77], [54]) partitions the data indices into K folds. It then runs K experiment(s) such that at the k-th experiment ($k = 1, \ldots, K$), new model parameters are learned using all the examples but withholding those from fold k. Then, the remaining data points from fold k can be used as unseen validation data to evaluate the model's ability to generalise. This typically results in a less biased evaluation of the model.

To denote the cross-validation procedure applied to a classification dataset, let K be the total number of folds. For each experiment $k = 1, \ldots, K$, the feature vectors $\boldsymbol{x}_i^k \in \mathbb{R}^d$ and labels $y_i^k \in$

$\{-1,+1\}$ relating to that fold will be identified with the superscript k. More precisely, we use the following notation for experiment k:

	TRAINING	VALIDATION
Cardinality:	n^k	$\bar{n}^k$
Data/examples:	$\left(\boldsymbol{x}_i^k, y_i^k\right)_{i=1,\dots,n^k}$	$\left(\bar{\boldsymbol{x}}_i^k, \bar{y}_i^k\right)_{i=1,\dots,\bar{n}^k}$
Kernel matrix:	$Q(\gamma)^k$	$\bar{Q}(\gamma)^k$

To represent a non-linear relationship we use the RBF kernel though our theory holds for any positive semi-definite kernel. Note that in training, the kernel $Q(\gamma)_{ij}^k := y_i^k y_j^k \exp\left(-\gamma\|\boldsymbol{x}_i^k - \boldsymbol{x}_j^k\|^2\right)$ is calculated between two training examples $(\boldsymbol{x}_i^k, y_i^k)$ and $(\boldsymbol{x}_j^k, y_j^k)$ but, in validation, the kernel $\bar{Q}(\gamma)_{ij}^k := \bar{y}_i^k y_j^k \exp\left(-\gamma\|\bar{\boldsymbol{x}}_i^k - \boldsymbol{x}_j^k\|^2\right)$ is between one validation $(\bar{\boldsymbol{x}}_i^k, \bar{y}_i^k)$ and one training example $(\boldsymbol{x}_j^k, y_j^k)$. The scalar $\gamma > 0$ is a hyperparameter. With this notation in mind we proceed to define the SVM hyperparameter tuning optimisation problem.

$$
\begin{aligned}
\underset{C,\gamma,\zeta,\boldsymbol{\alpha},\underline{\boldsymbol{v}},\bar{\boldsymbol{v}},u}{\text{minimise}} \quad & \frac{1}{K}\sum_{k=1}^{K}\frac{1}{\bar{n}^k}\sum_{i=1}^{\bar{n}^k}\zeta_i^k \\
\text{subject to} \quad & C \geq 0, \\
& \gamma \geq 0, \\
& \left.\begin{aligned} & \zeta_i^k \geq 0 \\ & \zeta_i^k \geq 1 - \sum_{j=1}^{n^k}\alpha_j^k \bar{Q}(\gamma)_{ij}^k - \bar{y}_i^k u^k \end{aligned}\right\} \begin{array}{l}\texttt{(Validation)} \\ \text{for } k = 1,\dots,K, \\ \quad \text{for } i = 1,\dots,\bar{n}^k,\end{array} \\
& \left.\begin{aligned} & \sum_{j=1}^{n^k}\alpha_j^k Q(\gamma)_{ij}^k - 1 - \underline{v}_i^k + \bar{v}_i^k + u^k y_i^k = 0 \\ & 0 \leq \alpha_i^k \perp \underline{v}_i^k \geq 0 \\ & 0 \leq (C - \alpha_i^k) \perp \bar{v}_i^k \geq 0 \\ & \sum_{j=1}^{n^k}\alpha_j^k y_j^k = 0 \end{aligned}\right\} \begin{array}{l}\texttt{(Training)} \\ \text{for } k = 1,\dots,K, \\ \quad \text{for } i = 1,\dots,n^k.\end{array}
\end{aligned}
\tag{$\mathcal{M}$}
$$

The set of constraints tagged `(Training)` are exactly the KKT conditions of the (Dual-SVM) for hyperparameters C and γ [55]. The decision variable α_i^k is non-zero if training data point i from experiment k is a support vector and $\underline{v}_i^k$, $\bar{v}_i^k$ and u^k are the Lagrange multipliers corresponding to its upper bound, lower bound and bias respectively. Since the (Dual-SVM)is strictly convex with affine constraints these constraints are only satisfied for an optimal choice of support vectors. For a full exposé on this system see supplementary material Subsection SM4.3 and [55, p. 1891].

The variable ζ_i^k and the two constraints tagged `(Validation)` calculate the validation loss of each data point i from fold k. The objective function then seeks to minimise the average validation loss. In the language of bilevel optimisation this is equivalent to the KKT-reformulation of a bilevel program with lower-level programs corresponding to training K dual-SVMs over training datasets $k = 1,\dots,K$ and upper-level program is minimising validation loss over the choice of hyperparameters. For full details on the link to bilevel programming see supplementary material Section SM2.

Problem ($\mathcal{M}$) is explicit but not concise. It uses a two-dimensional indexing scheme: First, it indexes the fold $k = 1,\dots,K$, then the individual examples i within that fold. We now wish to flatten the indices into a one-dimensional sequence. Hence, we introduce the index sets

$$
I := \left\{(k,i) : k = 1,\dots,K, i = 1,\dots,n^k\right\} \text{ and } \bar{I} := \left\{(k,i) : k = 1,\dots,K, i = 1,\dots,\bar{n}^k\right\}. \tag{4.1}
$$

This allows us to stack all the constraint functions into vectors via the notations

$$Z_i^k(\boldsymbol{z}) \coloneqq \zeta_i^k - 1 + \sum_{j=1}^{n^k} \alpha_j^k \bar{Q}(\gamma)_{ij}^k + \bar{y}_i^k u^k \qquad \text{for} \quad (k,i) \in \bar{I}, \tag{4.2a}$$

$$\theta_i^k(\boldsymbol{z}) \coloneqq \sum_{j=1}^{n^k} \alpha_j^k Q(\gamma)_{ij}^k - 1 - \underline{v}_i^k + \bar{v}_i^k + u^k y_i^k \qquad \text{for} \quad (k,i) \in I, \tag{4.2b}$$

$$\varphi^k(\boldsymbol{z}) \coloneqq \sum_{j=1}^{n^k} \alpha_j^k y_j^k \qquad \text{for} \quad k = 1, \ldots, K. \tag{4.2c}$$

where we have also introduced $\boldsymbol{z} \coloneqq [C, \gamma, \zeta, \boldsymbol{\alpha}, \underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u]^\top \in \mathbb{R}^N$ to collect all variables, which has the dimension $N \coloneqq 3(K-1)n + n + K + 2$. This leads to the representation $f(\boldsymbol{z}) \coloneqq \frac{1}{K} \sum_{k=1}^K \frac{1}{\bar{n}^k} \sum_{i=1}^{\bar{n}^k} \zeta_i^k$ for the upper-level objective function of $(\mathcal{M})$, as well as its constraint functions

$$g(\boldsymbol{z}) \coloneqq \begin{bmatrix} C \\ \gamma \\ \zeta_1^1 \\ \vdots \\ \zeta_{\bar{n}^K}^K \\ Z_1^1(\boldsymbol{z}) \\ \vdots \\ Z_{\bar{n}^K}^K(\boldsymbol{z}) \end{bmatrix}, \quad h(\boldsymbol{z}) \coloneqq \begin{bmatrix} \theta_1^1(\boldsymbol{z}) \\ \vdots \\ \theta_{n^K}^K(\boldsymbol{z}) \\ \varphi^1(\boldsymbol{z}) \\ \vdots \\ \varphi^K(\boldsymbol{z}) \end{bmatrix}, \quad G(\boldsymbol{z}) \coloneqq \begin{bmatrix} \alpha_1^1 \\ \vdots \\ \alpha_{n^K}^K \\ (C - \alpha_1^1) \\ \vdots \\ (C - \alpha_{n^K}^K) \end{bmatrix}, \quad H(\boldsymbol{z}) \coloneqq \begin{bmatrix} \underline{v}_1^1 \\ \vdots \\ \underline{v}_{n^K}^K \\ \bar{v}_1^1 \\ \vdots \\ \bar{v}_{n^K}^K \end{bmatrix}, \tag{4.3}$$

where we obviously have $g(\boldsymbol{z}) \in \mathbb{R}^{2+2n}$, $h(\boldsymbol{z}) \in \mathbb{R}^{(K-1)n+K}$ and $G(\boldsymbol{z})$, $H(\boldsymbol{z}) \in \mathbb{R}^{2(K-1)n}$. Subsequently, using the above constructions for the functions f, g, h, G and H, we can re-write problem $(\mathcal{M})$ more compactly in the general form MPCC.

4.2 Theoretical properties of the model

We begin by making an important assumption that for each fold, there exists at least one dual variable in the interior of its bounds. This is equivalent to assuming at least one data point lies on the margin.

$$\forall k = 1, \ldots, K, \quad \exists i \in \{1, \ldots, n^k\} \quad \text{such that } 0 < \alpha_i^k < C. \tag{4.4}$$

We now give the derivatives of the constraint functions of $(\mathcal{M})$. This will be useful later in studying which constraint qualifications problem $(\mathcal{M})$ satisfies.

Proposition 4.1 *The transposed Jacobian matrix of the constraint functions of* $(\mathcal{M})$ *is*

Constraint functions

Derivative w.r.t.	$g(\boldsymbol{z})$				$h(\boldsymbol{z})$		$G(\boldsymbol{z})$		$H(\boldsymbol{z})$	
	C	γ	ζ	$Z(\boldsymbol{z})$	$\theta(\boldsymbol{z})$	$\varphi(\boldsymbol{z})$	$\boldsymbol{\alpha}$	$(C - \boldsymbol{\alpha})$	$\underline{\boldsymbol{v}}$	$\bar{\boldsymbol{v}}$
$\nabla_C \bullet$	1	0	0	$\nabla_C Z(\boldsymbol{z})$	0	0	0	1	0	0
$\nabla_\gamma \bullet$	0	1	0	$\nabla_\gamma Z(\boldsymbol{z})$	$\nabla_\gamma \theta(\boldsymbol{z})$	0	0	0	0	0
$\nabla_\zeta \bullet$	0	0	$\mathcal{I}$	$\mathcal{I}$	0	0	0	0	0	0
$\nabla_{\boldsymbol{\alpha}} \bullet$	0	0	0	$\nabla_\alpha Z(\boldsymbol{z})$	$\mathcal{Q}(\gamma)$	$\mathcal{Y}$	$\mathcal{I}$	$-\mathcal{I}$	0	0
$\nabla_{\underline{\boldsymbol{v}}} \bullet$	0	0	0	0	$-\mathcal{I}$	0	0	0	$\mathcal{I}$	0
$\nabla_{\bar{\boldsymbol{v}}} \bullet$	0	0	0	0	$\mathcal{I}$	0	0	0	0	$\mathcal{I}$
$\nabla_u \bullet$	0	0	0	$\bar{\mathcal{Y}}^\top$	$\mathcal{Y}^\top$	0	0	0	0	0

(4.5)

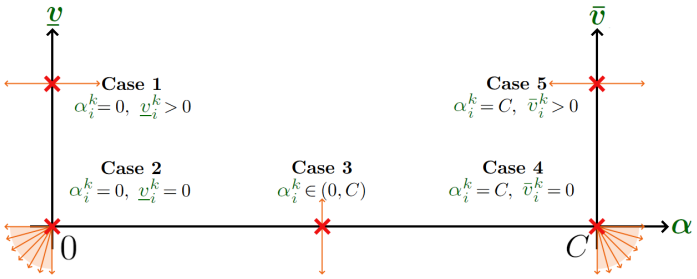

Fig. 3 This figure illustrates the geometry of our problem by showing example feasible points for each of the five index sets C_1, C_2, C_3, C_4 and C_5. The horizontal axis represents values of α_i^k. The vertical axis represents non-negative values of $\underline{v}_i^k$ and $\bar{v}_i^k$ when $\alpha_i^k = 0$ and $\alpha_i^k = C$, respectively. The orange arrows show directions in the normal cone of the feasible set of the problem.

with the column headers denoting which constraint function (refer to (4.3)*) the column corresponds to; the row headers* $\nabla_v \bullet$ *denote that their row corresponds to the derivative with respect to decision variable* v*;* $\mathcal{I}$ *is the identity matrix with inferred dimension;* $\mathcal{Q} \in \mathbb{R}^{|I|\times|I|}$ *and* $\mathcal{Y} \in \{-1, 0, 1\}^{|I|\times K}$ *are block diagonal matrices formed by*

$$\mathcal{Q}(\gamma) \coloneqq \begin{bmatrix} Q(\gamma)^1 & & 0 \\ & \ddots & \\ 0 & & Q(\gamma)^K \end{bmatrix}, \qquad \mathcal{Y} \coloneqq \begin{bmatrix} \boldsymbol{y}^1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \boldsymbol{y}^K \end{bmatrix} \text{ and } \qquad \boldsymbol{y}^k \coloneqq \begin{bmatrix} y_1^k \\ \vdots \\ y_{n^k}^k \end{bmatrix} \text{ for } k = 1, \dots K.$$

Using the standard MPCC index sets I^g, I^G, I^H, I^{GH} that were defined in (2.1) becomes laborious and hides the structure of our problem. Recalling the indexing scheme of (4.1), we define the problem-specific index sets

$$\bar{I}^\zeta \coloneqq \left\{(k,i) \in \bar{I} : \zeta_i^k = 0\right\} \quad \text{and} \quad \bar{I}^Z \coloneqq \left\{(k,i) \in \bar{I} : Z_i^k(\boldsymbol{z}) = 0\right\}.$$

We then partition the index set I into five cases that better describe $(\mathcal{M})$:

$$\begin{aligned}
C_1 &\coloneqq \left\{(k,i) \in I : \quad \alpha_i^k = 0, \quad \underline{v}_i^k > 0, \quad (C - \alpha_i^k) = C, \quad \bar{v}_i^k = 0\right\}; \\
C_2 &\coloneqq \left\{(k,i) \in I : \quad \alpha_i^k = 0, \quad \underline{v}_i^k = 0, \quad (C - \alpha_i^k) = C, \quad \bar{v}_i^k = 0\right\}; \\
C_3 &\coloneqq \left\{(k,i) \in I : \quad \alpha_i^k \in (0,C), \ \underline{v}_i^k = 0, \quad (C - \alpha_i^k) \in (0,C), \ \bar{v}_i^k = 0\right\}; \\
C_4 &\coloneqq \left\{(k,i) \in I : \quad \alpha_i^k = C, \quad \underline{v}_i^k = 0, \quad (C - \alpha_i^k) = 0, \quad \bar{v}_i^k = 0\right\}; \\
C_5 &\coloneqq \left\{(k,i) \in I : \quad \alpha_i^k = C, \quad \underline{v}_i^k = 0, \quad (C - \alpha_i^k) = 0, \quad \bar{v}_i^k > 0\right\}.
\end{aligned}$$

We denote the union of multiple cases in the subscript. For example, $C_{123} \coloneqq C_1 \cup C_2 \cup C_3$. It is easy to see that these sets are disjoint and cover all possible cases. That is, $C_{12345} = I$. Figure 3 illustrates the geometry of these index sets. Next, we show that $(\mathcal{M})$ satisfies MPCC-MFCQ-R at every feasible point with positive hyperparameters.

Proposition 4.2 *Let* $\boldsymbol{z} \coloneqq [C, \gamma, \zeta, \boldsymbol{\alpha}, \underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u]^\top$ *be any feasible point of problem* $(\mathcal{M})$ *such that* $C > 0$, $\gamma > 0$ *and* (4.4) *holds. Then MPCC-MFCQ-R holds at this point.*

Proof Recall from Lemma 2.4 that MPCC satisfies MPCC-MFCQ-R if and only if the family of gradients (A, B) defined in (2.6) is positive-linearly independent. We write these gradients explicitly for $(\mathcal{M})$ using the problem specific variables, index sets $\bar{I}^\zeta$, $\bar{I}^Z$, and partition C_l for $l = 1, \dots, 5$:

$$A \coloneqq \left\{\nabla_{\boldsymbol{z}} \alpha_i^k, \nabla_{\boldsymbol{z}} \underline{v}_i^k : (k,i) \in C_2\right\} \cup \left\{\nabla_{\boldsymbol{z}}(C - \alpha_i^k), \nabla_{\boldsymbol{z}} \bar{v}_i^k : (k,i) \in C_4\right\}$$

$$B \coloneqq \left\{\nabla_{\boldsymbol{z}} \alpha_i^k : (k,i) \in C_1\right\} \cup \left\{\nabla_{\boldsymbol{z}} \underline{v}_i^k : (k,i) \in C_{345}\right\} \cup \left\{\nabla_{\boldsymbol{z}} \zeta_i^k : (k,i) \in \bar{I}^{\zeta}\right\} \cup \left\{\nabla_{\boldsymbol{z}} Z_i^k(\boldsymbol{z}) : (k,i) \in \bar{I}^{Z}\right\}; \quad (4.6a)$$
$$\cup \left\{\nabla_{\boldsymbol{z}} (C - \alpha_i^k) : (k,i) \in C_5\right\}$$
$$\cup \left\{\nabla_{\boldsymbol{z}} \bar{v}_i^k : (k,i) \in C_{123}\right\} \cup \left\{\nabla_{\boldsymbol{z}} \theta_i^k(\boldsymbol{z}) : (k,i) \in I\right\} \cup \left\{\nabla_{\boldsymbol{z}} \varphi^k(\boldsymbol{z}) : k = 1, ..., K\right\}. \quad (4.6b)$$

Suppose there exist non-negative multipliers $\lambda^1, \lambda^2, \lambda^3, \lambda^4, \lambda^5, \lambda^6 \geq 0$ corresponding to the vectors in set A and free multipliers $\mu^1, \mu^2, \mu^3, \mu^4, \mu^5, \mu^6$ corresponding to the vectors in set B such that together they give rise to a linear combination whose sum is zero. More precisely,

$$\begin{aligned}
&\sum_{(k,i)\in C_2} \left(\lambda_{ki}^1 \nabla_{\boldsymbol{z}} \alpha_i^k + \lambda_{ki}^2 \nabla_{\boldsymbol{z}} \underline{v}_i^k\right) + \sum_{(k,i)\in C_4} \left(\lambda_{ki}^3 \nabla_{\boldsymbol{z}} (C - \alpha_i^k) + \lambda_{ki}^4 \nabla_{\boldsymbol{z}} \bar{v}_i^k\right) \\
&\quad + \sum_{(k,i)\in \bar{I}^{\zeta}}^{K} \lambda_{ki}^5 \nabla_{\boldsymbol{z}} \zeta_i^k + \sum_{(k,i)\in \bar{I}^{Z}}^{K} \lambda_{ki}^6 \nabla_{\boldsymbol{z}} Z_i^k(\boldsymbol{z}) + \sum_{(k,i)\in C_1} \mu_{ki}^1 \nabla_{\boldsymbol{z}} \alpha_i^k + \sum_{(k,i)\in C_{345}} \mu_{ki}^2 \nabla_{\boldsymbol{z}} \underline{v}_i^k \\
&\quad + \sum_{(k,i)\in C_5} \mu_{ki}^3 \nabla_{\boldsymbol{z}} (C - \alpha_i^k) + \sum_{(k,i)\in C_{123}} \mu_{ki}^4 \nabla_{\boldsymbol{z}} \bar{v}_i^k + \sum_{(k,i)\in I} \mu_{ki}^5 \nabla_{\boldsymbol{z}} \theta_i^k(\boldsymbol{z}) + \sum_{k=1}^{K} \mu_{ki}^6 \nabla_{\boldsymbol{z}} \varphi^k(\boldsymbol{z}) = \mathbf{0}. \quad (4.7)
\end{aligned}$$

where we have written $\nabla_{\boldsymbol{z}} \bullet$ to denote the derivative with respect to the full decision variable $\boldsymbol{z} \coloneqq [C, \gamma, \zeta, \boldsymbol{\alpha}, \underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u]^\top$. Moving forward, we will ignore the derivatives with respect to C and γ and write the remaining components of the derivative as vectors of five blocks corresponding to the derivatives with respect to $[\zeta, \boldsymbol{\alpha}, \underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u]$. For example $\nabla_{[\zeta, \boldsymbol{\alpha}, \underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u]}(\alpha_i^k) = [0, e_i^k, 0, 0, 0]^\top$ where e_i^k is the standard basis vector of zeros with a 1 at index (k,i). Thus we take (4.7) and rewrite explicitly

$$\begin{aligned}
&\sum_{(k,i)\in C_2} \left(\lambda_{ki}^1 \begin{bmatrix} 0 \\ e_i^k \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_{ki}^2 \begin{bmatrix} 0 \\ 0 \\ e_i^k \\ 0 \\ 0 \end{bmatrix}\right) + \sum_{(k,i)\in C_4} \left(\lambda_{ki}^3 \begin{bmatrix} 0 \\ -e_i^k \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_{ki}^4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ e_i^k \\ 0 \end{bmatrix}\right) \\
&\quad + \sum_{(k,i)\in \bar{I}^{\zeta}}^{K} \lambda_{ki}^5 \begin{bmatrix} e_i^k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \sum_{(k,i)\in \bar{I}^{Z}}^{K} \lambda_{ki}^6 \begin{bmatrix} e_i^k \\ \nabla_{\alpha} Z_i^k(\boldsymbol{z}) \\ 0 \\ 0 \\ e^k \bar{y}_i^k \end{bmatrix} + \sum_{(k,i)\in C_1} \mu_{ki}^1 \begin{bmatrix} 0 \\ e_i^k \\ 0 \\ 0 \\ 0 \end{bmatrix} + \sum_{(k,i)\in C_{345}} \mu_{ki}^2 \begin{bmatrix} 0 \\ 0 \\ e_i^k \\ 0 \\ 0 \end{bmatrix} \\
&\quad + \sum_{(k,i)\in C_5} \mu_{ki}^3 \begin{bmatrix} 0 \\ -e_i^k \\ 0 \\ 0 \\ 0 \end{bmatrix} + \sum_{(k,i)\in C_{123}} \mu_{ki}^4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ e_i^k \\ 0 \end{bmatrix} + \sum_{k=1}^{K} \mu_{ki}^5 \begin{bmatrix} 0 \\ Y^k \\ 0 \\ 0 \\ 0 \end{bmatrix} + \sum_{(k,i)\in I} \mu_{ki}^6 \begin{bmatrix} 0 \\ Q(\gamma)_i^k \\ -e_i^k \\ e_i^k \\ Y \end{bmatrix} = \mathbf{0}.
\end{aligned} \quad (4.8)$$

Inspecting the first row of the vector derivatives, which represents the derivative with respect to ζ, we have that $\lambda^5 + \lambda^6 = \mathbf{0}$. By their non-negativity we conclude $\lambda^5 = \mathbf{0}$ and $\lambda^6 = \mathbf{0}$.

We continue by inspecting subvectors of the multipliers based on C_{12345}. For the vector $\mu \in \mathbb{R}^{|I|}$ and matrix $\mathcal{Q} \in \mathbb{R}^{|I| \times |I|}$, denote the subvector and submatrix according to index set $C_l \subseteq I$ as

$$\mu_{C_l} \coloneqq \mu_{[(k,i)\in C_l]} \in \mathbb{R}^{|C_l|} \text{ and } \mathcal{Q}_{C_l, C_{l'}} \coloneqq \mathcal{Q}(\gamma)_{[(k,i)\in C_l, (k',i')\in C_{l'}]} \in \mathbb{R}^{|C_l| \times |C_{l'}|},$$

respectively, for $l = 1, \ldots, 5$. Note that $\mathcal{Q}_{C_l, C_l}$ is a principal submatrix of a positive definite matrix (Lemma SM2.3); therefore, it is positive definite itself. It is still a function of gamma, but we drop the argument for brevity.

Consider the derivative with respect to $\underline{v}_i^k$ for $(k,i) \in C_2$; i.e., the third row in our block notation in (4.8). We have that $\lambda^2 - \mu_{C_2}^6 = \mathbf{0}$, which implies that $\mu_{C_2}^6$ is non-negative. Consider the derivative with respect to α_i^k for $(k,i) \in C_2$. i.e. the second row block in (4.8). We have $\lambda^1 + \mathcal{Q}_{C_2, C_2} \mu_{C_2}^6 = \mathbf{0}$. Taking the inner product with $\mu_{C_2}^6$ gives

$$\left(\mu_{C_2}^6\right)^\top \left(\lambda^1\right) + \left(\mu_{C_2}^6\right)^\top \left(\mathcal{Q}_{C_2, C_2}\right) \left(\mu_{C_2}^6\right) = 0.$$

By the positive definiteness of $\mathcal{Q}_{C_2, C_2}$ and since $\lambda^1, \lambda^2, \mu_{C_2}^6 \geq 0$ we must have $\lambda^1 = \lambda^2 = \mu_{C_2}^6 = 0$.

In a similar manner, we now consider the derivative with respect to $\bar{v}_i^k$ for $(k,i) \in C_4$; i.e., the fourth row block in (4.8). We have that $\lambda^4 + \mu_{C_4}^6 = \mathbf{0}$ which implies that $\mu_{C_4}^6$ is non-positive. Considering the derivative with respect to α_i^k for $(k,i) \in C_4$, we have $-\lambda^3 + \mathcal{Q}_{C_4, C_4} \mu_{C_4}^6 = \mathbf{0}$. Taking the inner product with $\mu_{C_4}^6$ leads to

$$-\left(\mu_{C_4}^6\right)^\top \left(\lambda^3\right) + \left(\mu_{C_4}^6\right)^\top \left(\mathcal{Q}_{C_4, C_4}\right) \left(\mu_{C_4}^6\right) = 0.$$

By the positive definiteness of the principal submatrix $\mathcal{Q}_{C_4,C_4}$ of $\mathcal{Q}$, and given that λ^3, $\lambda^4 \geq 0$ and $\mu^6_{C_4} \leq 0$, we must have the equalities $\lambda^3 = \lambda^4 = \mu^6_{C_4} = \mathbf{0}$.

At this point, we have shown that λ^1, λ^2, λ^3, λ^4, λ^5, λ^6, $\mu^6_{C_2}$ and $\mu^6_{C_4}$ are all zero. Now, consider the remaining vectors involved in (4.8) whose multipliers we have not yet concluded are zero. These make up the set B (4.6b). By stacking the vectors as columns we get the following matrix which we shall refer to as $(\tilde{M})$

	1 $(\theta^k_i(\boldsymbol{z}))$ $(k,i)\in C_1$	2 $(\theta^k_i(\boldsymbol{z}))$ $(k,i)\in C_3$	3 $(\theta^k_i(\boldsymbol{z}))$ $(k,i)\in C_5$	4 $(\varphi^k(\boldsymbol{z}))$ $k=1,\ldots,K$	5 (α^k_i) $(k,i)\in C_1$	6 $(C-\alpha^k_i)$ $(k,i)\in C_5$	7 $(\underline{v}^k_i)$ $(k,i)\in C_{345}$	8 $(\bar{v}^k_i)$ $(k,i)\in C_{123}$
1 $(\nabla_{\alpha^k_i}\bullet)_{(k,i)\in C_1}$	$\mathcal{Q}_{C_1,C_1}$	$\mathcal{Q}_{C_3,C_1}$	$\mathcal{Q}_{C_5,C_1}$	$\mathcal{Y}_{C_1}$	$\mathcal{I}_{C_1}$	0	0	0
2 $(\nabla_{\alpha^k_i}\bullet)_{(k,i)\in C_3}$	$\mathcal{Q}_{C_1,C_3}$	$\mathcal{Q}_{C_3,C_3}$	$\mathcal{Q}_{C_5,C_3}$	$\mathcal{Y}_{C_3}$	0	0	0	0
3 $(\nabla_{\alpha^k_i}\bullet)_{(k,i)\in C_5}$	$\mathcal{Q}_{C_1,C_5}$	$\mathcal{Q}_{C_3,C_5}$	$\mathcal{Q}_{C_5,C_5}$	$\mathcal{Y}_{C_5}$	0	$-\mathcal{I}_{C_5}$	0	0
4 $(\nabla_{\underline{v}^k_i}\bullet)_{(k,i)\in C_1}$	$\mathcal{I}_{C_1}$	0	0	0	0	0	0	0
5 $(\nabla_{\underline{v}^k_i}\bullet)_{(k,i)\in C_{345}}$	0	$[-\mathcal{I}_{C_3},0,0]^\top$	$[0,0,-\mathcal{I}_{C_5}]^\top$	0	0	0	$\mathcal{I}_{C_{345}}$	0
6 $(\nabla_{\bar{v}^k_i}\bullet)_{(k,i)\in C_{123}}$	$[\mathcal{I}_{C_1},0,0]^\top$	$[0,0,\mathcal{I}_{C_3}]^\top$	0	0	0	0	0	$\mathcal{I}_{C_{123}}$
7 $(\nabla_{\bar{v}^k_i}\bullet)_{(k,i)\in C_5}$	0	0	$\mathcal{I}_{C_5}$	0	0	0	0	0
8 $(\nabla_{u^k}\bullet)_{k=1,\ldots,K}$	$\mathcal{Y}^\top_{C_1}$	$\mathcal{Y}^\top_{C_3}$	$\mathcal{Y}^\top_{C_5}$	0	0	0	0	0

The matrix $(\tilde{M})$ has full rank. This can be shown through a sequence of elementary row operations; e.g. as detailed in the supplementary materials Subsection SM2.3. Therefore, equation (4.7) is satisfied for non-negative λ only when $\lambda = \mathbf{0}, \mu = \mathbf{0}$, giving us that (A, B) is positive-linearly independent. □

Remark 4.3 The above proof uses the dual form of MPCC-MFCQ-R. It would be equivalent to show for the sets defined in (4.6a,b) that B is linearly independent and there exists a direction d such that $Bd = \mathbf{0}$ and $Ad > \mathbf{0}$. This direction d is illustrated geometrically by the orange arrows in Figure 3. Also, as a key consequence of Proposition 4.2, the assumptions of Theorem 3.1 and Theorem 3.4 hold accordingly.

We now show that problem $(\mathcal{M})$ in general fails the stronger constraint qualification MPCC-MFCQ-T, and therefore displaying that the second implication in Theorem 2.5 is strict for the application problem under consideration in this section.

Proposition 4.4 *The MPCC-MFCQ-T fails at all feasible points $\boldsymbol{z} := (C, \gamma, \zeta, \boldsymbol{\alpha}, \underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u)$ of the program $(\mathcal{M})$ with more than two biactive indices; i.e., with $|I^{GH}(\boldsymbol{z})| > 2$.*

Proof MPCC-MFCQ-T requires that the following family of vectors be linearly independent:

$$\Big\{\nabla h_i(\boldsymbol{z}) : i \in I^h\Big\} \cup \Big\{\nabla G_i(\boldsymbol{z}) : i \in I^G(\boldsymbol{z}) \cup I^{GH}(\boldsymbol{z})\Big\} \cup \Big\{\nabla H_i(\boldsymbol{z}) : i \in I^H(\boldsymbol{z}) \cup I^{GH}(\boldsymbol{z})\Big\}. \tag{4.9}$$

The constraint functions h, G and H are in the variable $(\boldsymbol{\alpha}, \underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u, C, \gamma)$, which has a dimension of $3(K-1)n + K + 2$, where n is the number of training examples and K is the number of folds in the cross-validation. Observe that all the vectors in (4.9) are constant with respect to ζ, so their derivatives are zero with respect to ζ. There are a total of $3(K-1)n + K + |I^{GH}|$ vectors in this set (4.9). Therefore, if $|I^{GH}| > 2$, then there are more vectors than non-zero dimensions, and so they cannot be linearly independent. □

By Theorem 2.5, we conclude that MPCC-LICQ also fails for program $(\mathcal{M})$. Many of these propositions are conditional on the regularisation hyperparameter C being strictly positive. This is a very sensible restriction, since on any dataset for which $C = 0$ or $\gamma = 0$ is optimal, the best classifier ignores the data entirely.

5 Numerical experiments

In this section, we present our numerical experiments on hyperparameter tuning for support vector machines where we solve large instances of the highly non-convex MPCC problem $(\mathcal{M})$ using the solution methods presented in Section 3 under various settings.

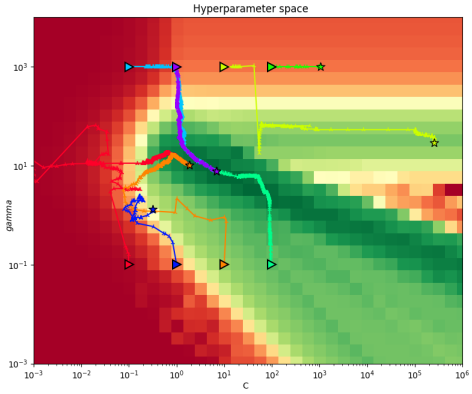

Fig. 4 This heat map shows the hyperparameter space of the Moons dataset. Green regions correspond to (C, γ) combinations, which allow the SVM to achieve high validation accuracy. The eight coloured lines represent the path of iterates produced by the Relaxation method from eight different starting points (triangles) to various stationary points (stars).

5.1 Implementation details

Each program and system of equations we wish to solve, such as (Dual-SVM), $(\mathcal{M})$, $(\mathcal{R}_\tau)$, $(\mathcal{P}_\pi)$ and (SM3.10), are written in the algebraic modelling language AMPL [56]. This allows us to robustly compute the derivatives and interface with the solvers we wish to use. The algorithms from Section 3 are implemented in Python [57] with use of the library NumPy [58] for efficient linear algebra computation. For solving the subproblems $(\mathcal{R}_\tau)$ and $(\mathcal{P}_\pi)$, we trialled two different solvers: Interior Point Optimizer (IPOPT) [59], an open source solver that implements a primal-dual interior point method with line search, and Artelys' Knitro [60], a commercial solver that implements an interior point trust region method. Both converged reliably in our context though our results are stated for Knitro as it shows faster runtimes.

Regarding cross-fold validation, it is common in the literature to set $K = 3$ (see [52, p. 91]) and $K = 5$ (see [53, Section 7.10.1]) as the number of folds. We set $K = 3$ because it gives a more stable validation accuracy for small datasets. The stopping tolerance for the subprograms is gradually decreased to 10^{-6}. The overarching MPCC loop in Algorithms 1 and 2 is terminated when one of each pair of complementarity constraint functions is at most 10^{-6} i.e.

$$\max_i \left(\min(|G_i(z)|,\ |H_i(z)|)\right) \leq 10^{-6}. \tag{5.1}$$

For the penalisation method, we select the initial penalty parameter to be 100 and sequentially increase it by multiples of 10. For the relaxation method, we select the initial relaxation parameter to be 0.01 and sequentially decrease it by multiples of 10^{-1}. Many publications have suggested using the so-called *exact penalisation* (e.g. [16]) and *exact relaxation* (e.g. [20]) approaches. Here, the parameter π (resp. τ) is fixed at some chosen value. If chosen correctly, only a single instance of the subproblem $(\mathcal{P}_\pi)$ (resp. $(\mathcal{R}_\tau)$) is solved resulting in an MPCC feasible solution. In our experiments, we implement this approach but note that our designation of fixed parameter comes from an intimate knowledge of the problem and is, in general, hard to find.

We use 18 datasets, 6 synthetically generated and 12 from real-world observations. For full details see the supplementary material Subsection SM5.2. When $(\mathcal{M})$ is parametrised with these data sets, it is large in comparison to other non-convex MPCCs that have been solved in the literature so far. The median problem size is 1773 variables, whereas for example, see Leyffer's famous MacMPEC collection [43], which has a median problem size of 49 variables.

5.2 Numerical behaviour of penalisation and relaxation methods

Our first observation is that due to the highly non-convex nature of $(\mathcal{M})$, the stationary point to which the algorithms converge is quite variable depending on the initialisation of the decision variables;

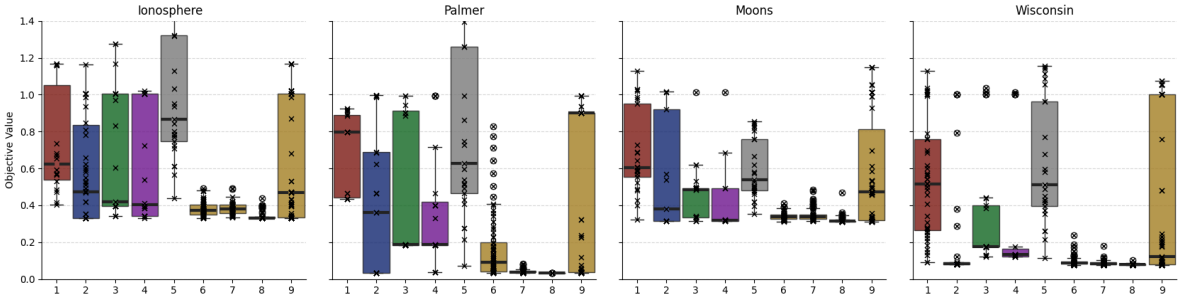

Fig. 5 These box plots show the distribution of objective values achieved by initialising from a range of different starting points. The four axes represent the datasets (left to right): Ionosphere, Palmer, Moons and Wisconsin. Within each axis, the box plots represent the solution method (left to right): **1. Penalisation (Exact), 2. Penalisation (Sequential), 3. Relaxation (Exact), 4. Relaxation (sequential), 5. Semi-Smooth Newton, 6. Grid Search, 7. Random Search, 8. Bayesian Optimisation, 9. Pattern Search.**

see Figure 4. Throughout the experiments a multi-start strategy is used which is fully described in supplementary material Section SM5.1.

Secondly, we observe that for the sequential relaxation method, parameter τ almost always decreases to 10^{-6}. That is, at each iteration t with $\tau^t = 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$, the solutions found by Algorithm 2 for the subproblem $(\mathcal{R}_{\tau^t})$ are not MPCC feasible (i.e. (5.1) does not hold) and the algorithm only terminates when the relaxation parameter reaches 10^{-6} and the complementarity constraints are forced within the exit tolerance. On the other hand, the penalisation method terminates with a range of different penalty parameters π^t between 10^3 and 10^6 producing MPCC feasible solutions. For more details see Table SM2. Further research should investigate the structure of $(\mathcal{M})$ to quantify why the penalisation approach is able to find an MPCC feasible solution before the penalty parameter π^t potentially gets too large and causes $(\mathcal{P}_{\pi^t})$ to become ill-conditioned (e.g. [18, Example 3]), as it is usually observed for penalty-based algorithms [50, 51].

Next we note that the complementarity constraint functions of $(\mathcal{M})$ are all linear. Also, observe that for all indices $k = 1, \ldots, K, i = 1, \ldots, n^k$, the equality constraint (4.2b) could be rearranged and used to substitute out either $\underline{v}_i^k$ or $\bar{v}_i^k$ from the complementarity constraints as is done by Coniglio et al. [42]. This would reduce the number of variables and constraints by $(K-1)n$ but introduce a nonlinear term in the complementarity constraints. In line with the advice of Fletcher and Leyffer [61, 62], we find that keeping the complementarity constraint linear is more beneficial than reducing the number of variables via substitution.

Finally we remark that the MPCC optimisation approach for hyperparameter optimisation, studied in this paper, sometimes overfits to the validation data. To monitor this, on top of the training and validation data that parametrises problem $(\mathcal{M})$, we require further data that is entirely unseen by the model; this will be called *test* data and is formed by putting aside 10% of examples and parametrising with the remaining 90%. Refer back to Figure SM1. The overfitting can then be observed when the classifiers, represented by the solutions of problem $(\mathcal{M})$, achieve higher validation accuracy than test accuracy. Evidence of this is presented in our later experiments, in particular, see the Ionosphere and Moons datasets in Table SM3.

5.3 Performance evaluation in comparison to derivative-free optimisation

Across industry and academia, Derivative-Free Optimisation (DFO) remains by far the most used tool for hyperparameter tuning [63]. It is simple and effective. The defining characteristic of DFO is that it treats training as a black box, meaning it does not use any information from the training process other than the final evaluation of the model in optimising the hyperparameters. We compare the MPCC optimisation to four such methods: *grid search*, *random search*, *Bayesian optimisation*, and *pattern search*. For the exact derivative free parametrisations and procedures used see Subsection SM5.3.

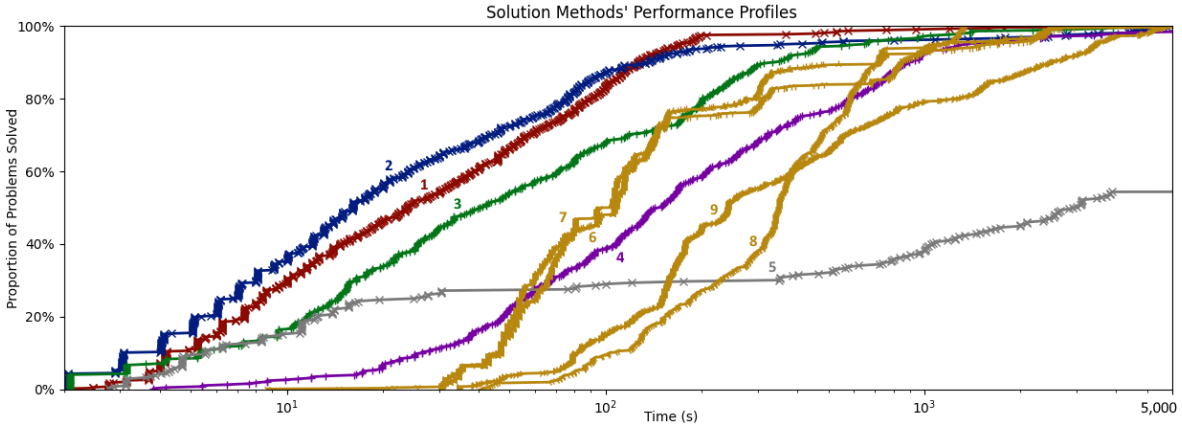

Fig. 6 This performance profile graph plots the proportion of instances that were solved to the target test accuracy (vertical axis) within the given time (horizontal axis). The lines are coloured and enumerated for clarity as follows: **1. Penalisation (Exact), 2. Penalisation (Sequential), 3. Relaxation (Exact), 4. Relaxation (sequential), 5. Semi-Smooth Newton, 6. Grid Search, 7. Random Search, 8. Bayesian Optimisation, 9. Pattern Search.**

Note that (Dual-SVM) has n variables and $2n$ constraints. On the other hand, the MPCC model ($\mathcal{M}$) has $7n + 5$ variables, $2n$ inequality constraints, $2n + 3$ equality constraints and $4n$ complementarity constraints. Therefore, the comparison must be made as to whether better performance can be achieved by minimising the much smaller problem (Dual-SVM) many times employing a DFO method or by minimising the much larger ($\mathcal{M}$) with the solution methods from Section 3.

We also add the semi-smooth Newton method to the experiment suite. This is an entirely different approach to solving MPCCs and converges to an M-stationary point, making it an interesting comparison, given that the sequential penalisation and Scholtes relaxation algorithms that we primarily study in this paper have been shown to instead converge to S- and C-stationarity, respectively. For the full algorithm see supplementary material Subsection SM3.4.

For each of the eighteen datasets detailed in supplementary material Subsection SM5.2, each solution method is run 100 times from different initialisations. We document the range of objective values achieved; see Table SM3 and box-plots in Figure 5. In general, the MPCC methods run much faster, but, on the other hand, converge to a large variety of stationary points, many of which have poor objective function values. To gain an understanding of their overall performance, we attribute a target test accuracy to each of the datasets. We then study the average runtime to achieve the target test accuracy. These results are plotted as a performance profile in Figure 6.

Overall, the obtained results make a convincing argument that our model ($\mathcal{M}$), being solved with the sequential penalisation method, can consistently outperform derivative-free methods in the tuning of SVM hyperparameters. Additionally, the sequential approach, when compared to the exact approach, often achieves a smaller range of objective function values (see Figure 5). The exact relaxation outperforms the derivative-free methods on a majority of instances. The semi-smooth Newton method performs well on small problems but fails to converge for larger problems - it seems to get stuck at the non-differentiable corners of the stationarity function. Between the derivative-free methods, grid and random search perform equally well on all but the very largest problems for which Bayesian optimisation is preferable. The penalisation methods clearly come out on top overall. Sequential penalisation is the fastest method to achieve the target test accuracy in 88% of instances. For the other 12%, exact penalisation is the fastest. However, it must be noted that, in practice, a user may prefer to always use sequential penalisation as it does not require the selection of a penalty parameter a priori.

6 Conclusion

We began by highlighting the distinction between the definitions of MPCC-MFCQ-T and MPCC-MFCQ-R, which is often overlooked in the literature. From there, we presented two solution methods

for the MPCC. For the penalisation and relaxation methods, we provided new convergence proofs that an inexact sequence of KKT points converges, under the assumption of MPCC-MFCQ-R, to S-stationary and C-stationary points, respectively. This provides a generalisation over the existing literature and allows the methods to be applied to a broader range of problems.

In the paper's second half, we presented our model ($\mathcal{M}$) for the hyperparameter tuning of support vector machines. We then proved that this satisfies the MPCC-MFCQ-R, thus satisfying the assumptions of the theorems proved earlier in the paper. Finally, we ran an extensive set of experiments on real-world data sets. In this context, we showed the practical use of our methods in beating the derivative-free search that is most commonly used. We conclude that the sequential penalisation method both converges faster and is more likely to converge to a higher quality objective value than the other methods.

For future work, we will develop first-order gradient descent methods to solve ($\mathcal{M}$) such as those being studied by Bengio [64] and Chapelle et al. [65]. These have a distinct advantage in machine learning applications as they can be efficiently parallelised on modern graphics processing units.

Acknowledgements

All graphics are produced using Matplotlib Pyplot [66] (version 3.8.2) and seaborn [67] (version 0.13.2). Thank you to Qingna Li from the Beijing Institute of Technology for her correction regarding assumption (4.4).

Statements and Declarations

The work of the first author is jointly funded by DAS Ltd and the School of Mathematical Sciences at the University of Southampton through a PhD studentship grant. The second author's research is partly funded by an EPSRC grant with reference EP/X040909/1. The authors have no competing interests to declare that are relevant to the content of this article. All datasets analysed during this study are publicly available; the individual sources are cited in supplementary material Subsection SM5.2. The AMPL models and Python implementations of the algorithms reported in this paper are available from the corresponding author on reasonable request. All authors contributed to theory established in this paper. All authors read and approved the final manuscript. This article is accompanied by supplementary material containing further technical details, proofs and numerical results; it is available at [44].

Supplementary Material: Mathematical programs with complementarity constraints and application to hyperparameter tuning

Contents

SM1 Table of notation

		Parameters
γ, δ	$:=$	Hyperparameters that determine the kernel.
C	$:=$	Regularisation hyperparameters for (Soft-Margin-SVM).
π^t	$:=$	Penalty parameter at iteration t.
τ^t	$:=$	Relaxation parameter at iteration t.
		Training and Validation Data
x_i	$:=$	Data point i.
x_i^k	$:=$	Training data point i in cross fold experiment k.
$\bar{x}_i^k$	$:=$	Validation data point i in cross fold experiment k.
y_i	$:=$	Label for data point i.
y_i^k	$:=$	Label for training data point i in cross fold experiment k.
$\bar{y}_i^k$	$:=$	Label for validation data point i in cross fold experiment k.
n	$:=$	Total number of data points.
n^k	$:=$	Number of training data points in cross fold experiment k.
$\bar{n}^k$	$:=$	Number of validation data points in cross fold experiment k.
$Q(\gamma)_{ij}$	$:=$	$y_i y_j \exp\left(-\gamma \Vert x_i - x_j\Vert^2\right)$ the full RBF kernel matrix.
$Q(\gamma)_{ij}^k$	$:=$	$y_i^k y_j^k \exp\left(-\gamma \Vert x_i^k - x_j^k\Vert^2\right)$ the k^{th} training RBF kernel matrix.
$\bar{Q}(\gamma)_{ij}^k$	$:=$	$\bar{y}_i^k y_j^k \exp\left(-\gamma \Vert \bar{x}_i^k - x_j^k\Vert^2\right)$ the k^{th} validation RBF kernel matrix.
		Decision Variables
w, b	$:=$	Weights and bias for the primal program (Soft-Margin-SVM).
$\boldsymbol{\alpha}, \boldsymbol{\eta}$	$:=$	Dual variables for the dual program (Dual-SVM).
$\underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u$	$:=$	Multipliers in the KKT conditions of the dual SVM (KKT-SVM).
$\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi}$	$:=$	Lagrange multipliers for the MPCC; see (2.7).
		Functions
f	$:=$	Objective function.
g	$:=$	Inequality constraint function i.e. $g(\boldsymbol{z}) \geq 0$.
h	$:=$	Equality constraint function i.e. $h(\boldsymbol{z}) = 0$.
G, H	$:=$	Complementarity constraint functions i.e. $0 \leq G(\boldsymbol{z}) \perp H(\boldsymbol{z}) \geq 0$.
Z, θ, φ	$:=$	Specific constraint functions of model $(\mathcal{M})$ defined in (4.2).
F	$:=$	System of nonlinear equations corresponding to M-stationary points (SM3.10).
		Index Sets
I^g, I^h	$:=$	Index sets of the active constraints; see Section 2.1.
I^G, I^H, I^{GH}	$:=$	Index sets of the complementarity constraints; see equation (2.1).
$I, \bar{I}$	$:=$	Sets of indices (k, i) for fold k, example i in training and validation respectively; see equation (4.1).
C_{12345}	$:=$	The five index cases used for the proof of Proposition 4.2.

SM2 Relationship to bilevel optimisation

As our support vector machine (SVM) application can be viewed as a bilevel optimisation problem, it is important to introduce the definition of a bilevel optimisation program and present its relationship to the MPCC.

First, recall that bilevel optimisation dates back to Stackelberg's economic leader-follower game theory [68], where the leader first gets to select their decision variable, that we denote here by $\boldsymbol{z_1} \in \mathbb{R}^n$, to optimise their upper-level objective function $f_{\text{up}}(\boldsymbol{z}_1, \boldsymbol{z}_2)$ subject to some upper-level constraints $g_{\text{up}}(\boldsymbol{z}_1) \geq 0$ and $h_{\text{up}}(\boldsymbol{z}_1) = 0$. Whatever selection the leader makes for $\boldsymbol{z}_1$, the follower will then get to choose their own variable, represented here by $\boldsymbol{z_2} \in \mathbb{R}^m$, to optimise their lower-level objective function $f_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2)$ subject to some lower-level constraints $g_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2) \geq 0$ and $h_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2) = 0$. However, as the leader's choices depend on the follower's variable $\boldsymbol{z}_2$, the upper-level player has to carefully choose their strategy. More precisely, given dimensions $n, m, p, q, r, s \in \mathbb{N}$ and differentiable functions f_{up}, f_{low}, g_{up}, h_{up}, g_{low}, and h_{low} from domain $\mathbb{R}^{n+m}$ to range $\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}^p$, $\mathbb{R}^q$, $\mathbb{R}^r$, and $\mathbb{R}^s$, respectively, the described *Bilevel Optimisation Program* can be written as

$$\begin{aligned} \underset{\boldsymbol{z}_1, \boldsymbol{z}_2}{\text{minimise}} \quad & f_{\text{up}}(\boldsymbol{z}_1, \boldsymbol{z}_2) \\ \text{subject to} \quad & g_{\text{up}}(\boldsymbol{z}_1) \geq 0, \\ & h_{\text{up}}(\boldsymbol{z}_1) = 0, \\ & \boldsymbol{z}_2 \in S(\boldsymbol{z}_1) \coloneqq \underset{\boldsymbol{z}_2'}{\operatorname{argmin}} \left\{ f_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2') : \quad g_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2') \geq 0, \;\; h_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2') = 0 \right\}, \end{aligned} \tag{BP}$$

where the lower-level problem corresponds to the following program parametrised by $\boldsymbol{z}_1$ is

$$\underset{\boldsymbol{z}_2'}{\text{minimise}} \;\; f_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2') \quad \text{subject to} \;\; g_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2') \geq 0, \; h_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2') = 0.$$

Problem (BP) has a wide range of applications (see, e.g., [69] for more details) and our SVM hyperparameter tuning problem in Section 4 is one its practical applications.

General NLP assumptions and CQs can be applied to the lower-level program. For instance, we say that problem (BP) has a convex lower-level program if and only if for all $\boldsymbol{z}_1 \in \mathbb{R}^n$, the functions $f(\boldsymbol{z}_1, .)$ and $g_i(\boldsymbol{z}_1, .)$, for $i = 1, \ldots, q$, are convex and $h(\boldsymbol{z}_1, .)$ is affine linear. Additionally, we say that the *Lower-Level Linear Independence Constraint Qualification* (LLICQ) and respectively the *Lower-Level Mangasarian–Fromovitz Constraint Qualification* (LMFCQ) are satisfied at a point $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ if and only if (LICQ) and respectively (MFCQ) are satisfied at the point $\boldsymbol{z}_2$ for the lower-level problem parametrised by $\boldsymbol{z}_1$.

A common technique towards solving problem (BP) is to replace the lower-level problem with its Karush–Kuhn–Tucker (KKT) conditions, leading to the problem

$$\begin{aligned} \underset{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{\lambda}, \boldsymbol{\mu}}{\text{minimise}} \quad & f_{\text{up}}(\boldsymbol{z}_1, \boldsymbol{z}_2) \\ \text{subject to} \quad & g_{\text{up}}(\boldsymbol{z}_1) \geq 0, \\ & h_{\text{up}}(\boldsymbol{z}_1) = 0, \\ & \begin{cases} \nabla_{\boldsymbol{z}_2} f_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2) - \boldsymbol{\lambda}^\top \nabla_{\boldsymbol{z}_2} g_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2) - \boldsymbol{\mu}^\top \nabla_{\boldsymbol{z}_2} h_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2) = 0, \\ 0 \leq \boldsymbol{\lambda} \perp g_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2) \geq 0, \\ h_{\text{low}}(\boldsymbol{z}_1, \boldsymbol{z}_2) = 0, \end{cases} \end{aligned} \tag{KKTR}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^r$ and $\boldsymbol{\mu} \in \mathbb{R}^s$ correspond to the Lagrange multiplier associated with the lower-level constraint functions $g(\boldsymbol{z}_1, \boldsymbol{z}_2)$ and $h(\boldsymbol{z}_1, \boldsymbol{z}_2)$ at the point $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ such that $\boldsymbol{z}_1$ is the upper-level selection and $\boldsymbol{z}_2$ is the corresponding lower-level optimal solution.

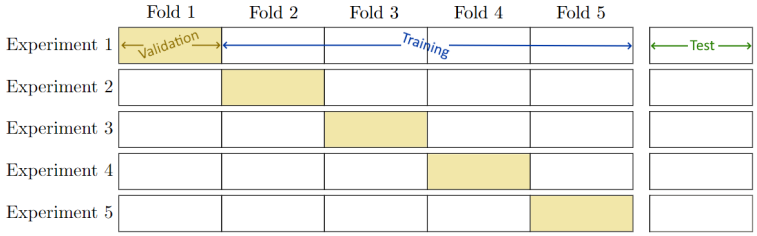

Fig. SM1 A visualisation of 5-fold cross-validation with dedicated test data kept aside.

Clearly, problem (KKTR) is a special class of problem (MPCC). However, problem (KKTR) is not necessarily equivalent to the original problem (BP), as shown in [70]. Nevertheless, under suitable assumptions, the following global and local relationships are well-known.

Theorem SM2.1 ([70]) *Let the lower-level problem in* (BP) *be convex.*

(i) Let $(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*)$ *be a global (resp. local) optimal solution of problem* (BP) *and assume that the LMFCQ holds at* $(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*)$*. Then, for each choice of Lagrange multipliers*

$$(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \text{ such that } \begin{cases} \nabla_{\boldsymbol{z}_2^*} f_{low}(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*) - \boldsymbol{\lambda}^{*\top} \nabla_{\boldsymbol{z}_2^*} g_{low}(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*) - \boldsymbol{\mu}^{*\top} \nabla_{\boldsymbol{z}_2^*} h_{low}(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*) = 0, \\ 0 \leq \boldsymbol{\lambda}^* \perp g_{low}(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*) \geq 0, \\ h_{low}(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*) = 0, \end{cases}$$

the point $(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ *is a global (resp. local) optimal solution of problem* (KKTR)*.*

(ii) Let $(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ *be a global (resp. local) optimal solution of problem* (KKTR) *and assume the LLICQ holds at* $(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*)$ *(resp. LLICQ holds at* $(\boldsymbol{z}_1, \boldsymbol{z}_2)$*, for all* $\boldsymbol{z}_2 \in S(\boldsymbol{z}_1)$*, for all* $\boldsymbol{z}_1 : G(\boldsymbol{z}_1) \geq 0, H(\boldsymbol{z}_1) = 0$*). Then,* $(\boldsymbol{z}_1^*, \boldsymbol{z}_2^*)$ *is a global (resp. local) optimal solution of problem* (BP)*.*

This theorem establishes global and local relationships between problem (BP) and its KKT/MPCC reformulation (KKTR). This will be useful in Section 4 when we construct a model for hyperparameter selection in support vector machines.

SM2.1 SVM hyperparameter tuning as a bilevel program

With the above theory in mind, we can now model our SVM hyperparameter tuning problem, as a bilevel program. We begin in the primal space using the notation of Subsection SM4.1. The leader is to select hyperparameters C and a feature mapping ϕ that minimise the average K-fold cross-validation error measured by the hinge loss. The follower wishes to select optimal model parameters w^k and b^k according to (Soft-Margin-SVM), for each fold k, that minimise the training error.

$$\begin{aligned} \underset{C,\phi,w^1,\ldots,w^K,b^1,\ldots,b^K}{\text{minimise}} \quad & \sum_{k=1}^{K} \frac{1}{\bar{n}^k} \sum_{i=1}^{\bar{n}^k} \max\left(0, 1 - \bar{y}_i^k \left(w^{k\top} \phi\left(\bar{\boldsymbol{x}}_i^k\right) + b^k\right)\right) \\ \text{subject to} \quad & C \geq 0, \\ & \text{For each } k = 1, \ldots, K, \text{ lower-level variables } w^k, b^k \text{ optimise} \\ & \text{(Soft-Margin-SVM) parametrised by } (x_i^k, y_i^k)_{i=1,\ldots,n^k}. \end{aligned} \tag{BP-primal}$$

For the problem to be well-posed, as before, we will convert the upper-level variables into dual ones, by means of the kernel trick referring to the notation of Subsection SM4.2. This can be done by substituting $w^k = \sum_j \alpha_j^k y_j^k \phi\left(\boldsymbol{x}_j^k\right)$ into the objective function and applying the kernel trick $\bar{y}_i^k y_j^k \phi\left(\bar{\boldsymbol{x}}_i^k\right) \phi\left(\boldsymbol{x}_j^k\right) = \bar{Q}(\gamma)_{ij}^k$. A few other technical arrangements with the bias b^k will lead to the dual

formulation.

$$\begin{aligned} \underset{C,\gamma,\boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^K}{\text{minimise}} \quad & \sum_{k=1}^{K} \frac{1}{\bar{n}^k} \sum_{i=1}^{\bar{n}^k} \max\left(0, 1 - \sum_{j=1}^{n^k} \alpha_j^k \bar{Q}(\gamma)_{ij}^k - \bar{y}_i^k b^k\left(\boldsymbol{\alpha}^k, \gamma, C\right)\right). \\ \text{subject to} \quad & C \geq 0, \gamma \geq 0, \\ & \text{For each } k = 1, ..., K, \text{ lower-level variables } \boldsymbol{\alpha}^k \text{ optimise} \\ & \text{(Dual-SVM) parametrised by } (x_i^k, y_i^k)_{i=1,\ldots,n^k}. \end{aligned} \tag{BP-dual}$$

To obtain a smooth upper-level objective function, we remove the max operator by introducing slack variables ζ_i and pairs of constraints $\zeta_i \geq 0$ and $\zeta_i \geq 1 - \sum \alpha_j^k \bar{Q}(\gamma)_{ij}^k - \bar{y}_i^k b^k\left(\boldsymbol{\alpha}^k, \gamma, C\right)$. This must be done for each fold $k = 1, ..., K$ and for each validation example in those folds $i = 1, \ldots, \bar{n}^k$. To remove the embedded lower-level optimisation problem, we perform a KKT reformulation. This results in the MPCC model $(\mathcal{M})$ given in Subsection 4.1.

We now show that the lower-level problem (Dual-SVM) satisfies the assumptions of Theorem SM2.1, and so, in the provided framework, problems $(\mathcal{M})$ and (BP-dual) will be locally equivalent.

Proposition SM2.2 *Problem* (BP-dual) *has a convex lower level program, and satisfies the LLICQ at every point* $\boldsymbol{z} \coloneqq [C, \gamma, \zeta, \boldsymbol{\alpha}, \underline{v}, \overline{v}, u]^\top$ *where* $C > 0$, $\gamma > 0$ *and* (4.4) *holds.*

Proof By Lemma SM2.3, the objective function of (Dual-SVM) is a positive quadratic function. Each of the constraints is linear in $\boldsymbol{\alpha}$. Therefore, we have convexity. For each index i, only one of the constraints $0 \leq \alpha_i$ or $\alpha_i \leq C$ can be active. Therefore, each of our active inequality constraints is in a unique variable. Furthermore, by assumption (4.4), there exists some index i such that the variable α_i is involved in the equality constraint $\sum \alpha_i y_i = 0$ but none of the active inequality constraints. Thus, their derivatives are linearly independent, and problem (BP-dual) fulfils the LLICQ. □

SM2.2 Kernel matrix Q is positive definite

Lemma SM2.3 *Given distinct data points* $x_1^k, \ldots, x_{n^k}^k \in \mathbb{R}^d$ *and labels* $y_1^k, \ldots, y_{n^k}^k \in \{-1, 1\}$ *for each* $k = 1, \ldots, K$ *and* $\gamma > 0$, *we have:*

(i) Each RBF kernel matrix $Q(\gamma)^k \in \mathbb{R}^{n \times n}$, *defined by* $Q(\gamma)_{ij}^k \coloneqq y_i^k y_j^k \exp\left(-\gamma \left\|x_i^k - x_j^k\right\|^2\right)$ *for* $i = 1, \ldots, n$ *and* $j = 1, \ldots, n$, *is positive definite.*

(ii) The following block diagonal matrix $\mathcal{Q}(\gamma)$ *formed from blocks* $Q(\gamma)^1, \ldots, Q(\gamma)^K$ *is also positive definite:*

$$\mathcal{Q}(\gamma) \coloneqq \begin{bmatrix} Q(\gamma)^1 & & 0 \\ & \ddots & \\ 0 & & Q(\gamma)^K \end{bmatrix}. \tag{SM2.1}$$

Proof The Gaussian $x \mapsto \exp(-\gamma\|x\|^2)$ is a positive definite function on $\mathbb{R}^d$ [71, Theorem 6.10]. Therefore, $P_{ij}^k \coloneqq \exp\left(-\gamma \left\|x_i^k - x_j^k\right\|^2\right)$ is a positive definite matrix [71, Definition 6.1]. Hence, for any non-zero vector $v \in \mathbb{R}^n$, by the positive definiteness of P^k, it holds that

$$v^\top Q(\gamma)^k v = \sum_{i=1}^{n} \sum_{j=1}^{n} v_i v_j y_i^k y_j^k \exp\left(-\gamma \left\|x_i^k - x_j^k\right\|^2\right) = (vy)^\top P^k (vy) > 0,$$

where $vy \in \mathbb{R}^n$ is the vector $(vy)_i = v_i y_i^k$ for $i = 1, \ldots, n$. Thus, $Q(\gamma)^k$ is positive definite and so has only positive eigenvalues. Applying either [72, Definition 1.59, Property a] or Schur determinant formula [73] to the characteristic polynomial of $\mathcal{Q}(\gamma)$ we get:

$$\det\left(\mathcal{Q}(\gamma) - \lambda \mathcal{I}_n\right) = \prod_{k=1}^{K} \det\left(Q(\gamma)^k - \lambda \mathcal{I}_{n^k}\right).$$

Hence, $\mathcal{Q}(\gamma)$ has all positive eigenvalues and is therefore positive definite. □

SM2.3 Constraint matrix M has full rank

Recall the matrix $(\tilde{M})$ presented in the proof of Proposition 4.2. In this subsection, we walk through a sequence of elementary row operations to demonstrate that $(\tilde{M})$ is full rank. These are described in five steps. First, the row-blocks are enumerated $1, 2, 3, 4, 5, 6, 7, 8$ and are reordered to $1, 3, 5, 6, 2, 4, 7, 8$. Secondly, the column-blocks are enumerated $1, 2, 3, 4, 5, 6, 7, 8$ and are reordered to $5, 6, 7, 8, 2, 1, 3, 4$. Thirdly, we multiply rows in row-block 3 by -1 so that its leading entries are positive.

Observe that $\mathcal{Q}_{C_3,C_3}$ has full rank, so there exist elementary row operations that map $\mathcal{Q}_{C_3,C_3}$ into row echelon form. Denote with R the product of matrices that perform those elementary row operations so that $R\mathcal{Q}_{C_3,C_3}$ is in row echelon form. The fourth step is to premultiply row-block 2 by R.

Finally, take the rows of 8 and subtract scaled copies of rows of 2, 4 and 7 according to Gaussian elimination until the rows of 8 are zero in all columns of 2, 1, and 3. This leaves some non-zero matrix which we will denote S in row-block 8, column-block 4. Since the individual rows $k = 1, \ldots, K$ of row-block 8 correspond to the same folds $k = 1, \ldots, K$ as the columns $k = 1, \ldots, K$ of column-block 4, the matrix S is diagonal. Furthermore, assumption (4.4) gives us that C_3 is non-empty for each fold $k = 1, \ldots, K$; therefore, the diagonal elements of S are non-zero. After these five steps of row operations, the resulting matrix can be written in the following way:

	5 (α_i^k) $(k,i)\in C_1$	6 $(C-\alpha_i^k)$ $(k,i)\in C_5$	7 $(\underline{v}_i^k)$ $(k,i)\in C_{345}$	8 $(\bar{v}_i^k)$ $(k,i)\in C_{123}$	2 $(\theta_i^k(\boldsymbol{z}))$ $(k,i)\in C_3$	1 $(\theta_i^k(\boldsymbol{z}))$ $(k,i)\in C_1$	3 $(\theta_i^k(\boldsymbol{z}))$ $(k,i)\in C_5$	4 $(\varphi^k(\boldsymbol{z}))$ $k=1,\ldots,K$
1 $(\nabla_{\alpha_i^k}\bullet)_{(k,i)\in C_1}$	$\mathcal{I}_{C_1}$	0	0	0	$\mathcal{Q}_{C_3,C_1}$	$\mathcal{Q}_{C_1,C_1}$	$\mathcal{Q}_{C_5,C_1}$	$\mathcal{Y}_{C_1}$
3 $-(\nabla_{\alpha_i^k}\bullet)_{(k,i)\in C_5}$	0	$\mathcal{I}_{C_5}$	0	0	$-\mathcal{Q}_{C_3,C_5}$	$-\mathcal{Q}_{C_1,C_5}$	$-\mathcal{Q}_{C_5,C_5}$	$-\mathcal{Y}_{C_5}$
5 $(\nabla_{\underline{v}_i^k}\bullet)_{(k,i)\in C_{345}}$	0	0	$\mathcal{I}_{C_{345}}$	0	$[-\mathcal{I}_{C_3}, 0, 0]^\top$	0	$[0, 0, -\mathcal{I}_{C_5}]^\top$	0
6 $(\nabla_{\bar{v}_i^k}\bullet)_{(k,i)\in C_{123}}$	0	0	0	$\mathcal{I}_{C_{123}}$	$[0, 0, \mathcal{I}_{C_3}]^\top$	$[\mathcal{I}_{C_1}, 0, 0]^\top$	0	0
2 $R(\nabla_{\alpha_i^k}\bullet)_{(k,i)\in C_3}$	0	0	0	0	$R\mathcal{Q}_{C_3,C_3}$	$R\mathcal{Q}_{C_1,C_3}$	$R\mathcal{Q}_{C_5,C_3}$	$R\mathcal{Y}_{C_3}$
4 $(\nabla_{\underline{v}_i^k}\bullet)_{(k,i)\in C_1}$	0	0	0	0	0	$\mathcal{I}_{C_1}$	0	0
7 $(\nabla_{\bar{v}_i^k}\bullet)_{(k,i)\in C_5}$	0	0	0	0	0	0	$\mathcal{I}_{C_5}$	0
8 †	0	0	0	0	0	0	0	S

From this sequence of row reductions, we see that each column has a unique row index with its leading non-zero entry. Therefore, the matrix $(\tilde{M})$ has full rank.

SM3 Further details of the solution methods

Here we provide the equations, algorithms and proofs that were excluded from Section 3 of the main article for brevity.

SM3.1 Proof of Theorem 3.2

Proof We first want to show that $\left(G(\boldsymbol{z}^t)^\top H(\boldsymbol{z}^t)\right)_{t\in\mathbb{N}}$ is a non-increasing sequence. We will apply the classical trick of taking a difference of terms at two indices. Let $\bar{t} > t$. Notice both $\boldsymbol{z}^{\bar{t}}$ and $\boldsymbol{z}^t$ are feasible for both $(\mathcal{P}_{\pi^{\bar{t}}})$ and $(\mathcal{P}_{\pi^t})$. Since $\boldsymbol{z}^t$ minimises $f(\boldsymbol{z}) + \pi^t G(\boldsymbol{z})H(\boldsymbol{z})$ and $\boldsymbol{z}^{\bar{t}}$ minimises $f(\boldsymbol{z}) + \pi^{\bar{t}} G(\boldsymbol{z})H(\boldsymbol{z})$ we can conclude that

$$f(\boldsymbol{z}^{\bar{t}}) + \pi^t G(\boldsymbol{z}^{\bar{t}})H(\boldsymbol{z}^{\bar{t}}) \geq f(\boldsymbol{z}^t) + \pi^t G(\boldsymbol{z}^t)H(\boldsymbol{z}^t), \tag{SM3.1a}$$

$$f(\boldsymbol{z}^t) + \pi^{\bar{t}} G(\boldsymbol{z}^t)H(\boldsymbol{z}^t) \geq f(\boldsymbol{z}^{\bar{t}}) + \pi^{\bar{t}} G(\boldsymbol{z}^{\bar{t}})H(\boldsymbol{z}^{\bar{t}}). \tag{SM3.1b}$$

Adding the two inequalities (SM3.1a) and (SM3.1b) gives

$$(\pi^{\bar{t}} - \pi^t)\left(G(\boldsymbol{z}^t)H(\boldsymbol{z}^t) - G(\boldsymbol{z}^{\bar{t}})H(\boldsymbol{z}^{\bar{t}})\right) \geq 0.$$

Since $(\pi^{\bar{t}} - \pi^t) > 0$, it follows that $G(\boldsymbol{z}^t)H(\boldsymbol{z}^t) \geq G(\boldsymbol{z}^{\bar{t}})H(\boldsymbol{z}^{\bar{t}})$.

Next, we show that $\lim_{t\to\infty}\left(G(\boldsymbol{z}^t)^\top H(\boldsymbol{z}^t)\right) = 0$. Let $\bar{\boldsymbol{z}}$ be any feasible point of MPCC. Let $\boldsymbol{z}^1$ be the minimiser of the penalised problem for $\pi^1 = 1$. Towards a contradiction, suppose that there exists an $\epsilon > 0$ such that for all $T \in \mathbb{N}$ we have $G(\boldsymbol{z}^t)^\top H(\boldsymbol{z}^t) > \epsilon$ for some $t > T$. Under that premise, we can choose an arbitrarily large t such that $\pi^t \geq \frac{1}{\epsilon}|f(\bar{\boldsymbol{z}}) - f(\boldsymbol{z}^1)| + 2$ and $G(\boldsymbol{z}^t)H(\boldsymbol{z}^t) > \epsilon$. Consider the infimum over the objective values of problem MPCC:

$$\begin{aligned}
&\inf\{f(\boldsymbol{z}) :\ g(\boldsymbol{z}) \geq 0,\ h(\boldsymbol{z}) = 0,\ 0 \leq G(\boldsymbol{z}) \perp H(\boldsymbol{z}) \geq 0\} && \text{(SM3.2a)}\\
&\geq \inf\left\{f(\boldsymbol{z}) + \pi^t G(\boldsymbol{z})H(\boldsymbol{z}) :\ g(\boldsymbol{z}) \geq 0,\ h(\boldsymbol{z}) = 0,\ G(\boldsymbol{z}) \geq 0,\ H(\boldsymbol{z}) \geq 0\right\} && \text{(SM3.2b)}\\
&= f(\boldsymbol{z}^t) + \pi^t G(\boldsymbol{z}^t)H(\boldsymbol{z}^t)\\
&\geq f(\boldsymbol{z}^1) + \pi^t G(\boldsymbol{z}^t)H(\boldsymbol{z}^t) && \text{(SM3.2c)}\\
&\geq f(\boldsymbol{z}^1) + \pi^t \epsilon\\
&\geq f(\boldsymbol{z}^1) + |f(\bar{\boldsymbol{z}}) - f(\boldsymbol{z}^1)| + 2\epsilon\\
&> f(\bar{\boldsymbol{z}}). && \text{(SM3.2d)}
\end{aligned}$$

It is helpful to remember that t here is chosen according to ϵ. The first inequality (SM3.2b) comes from the fact that any global minimum of MPCC is also feasible for $(\mathcal{P}_\pi)$. The second inequality (SM3.2c) comes from the fact that $\boldsymbol{z}^t$ is a global minimum of $f(\boldsymbol{z}) + \pi^t G(\boldsymbol{z})H(\boldsymbol{z})$.

The conclusion of (SM3.2d) implies that $\bar{\boldsymbol{z}}$ achieves a smaller objective value than the infimum over objectives of the MPCC feasible points in problem (SM3.2a). But this contradicts the MPCC feasibility of $\bar{\boldsymbol{z}}$. Therefore we conclude that $\lim_{t\to\infty}\left(G(\boldsymbol{z}^t)^\top H(\boldsymbol{z}^t)\right) = 0$. By the continuity of the functions G and H, we have $G(\boldsymbol{z}^*)^\top H(\boldsymbol{z}^*) = 0$. □

SM3.2 ϵ-KKT system of MPCC relaxation

In a similar fashion to our construction for penalisation, we shall not expect the NLP solvers to find the exact global minimum but instead terminate within a certain tolerance $\epsilon \geq 0$ of a KKT point. We say a point $\boldsymbol{z} \in \mathbb{R}^n$ is an ϵ-KKT point for $(\mathcal{R}_\tau)$ if there exists a Lagrange multiplier vector $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi}, \boldsymbol{\omega}) \in \mathbb{R}^{(p+q+3r)}$ such that

$$\begin{aligned}
\Big\|\nabla f(\boldsymbol{z}) - \sum_{i=1}^{p}\lambda_i\nabla g_i(\boldsymbol{z}) - \sum_{i=1}^{q}\mu_i\nabla h_i(\boldsymbol{z}) - \sum_{i=1}^{r}\nu_i\nabla G_i(\boldsymbol{z}) - \sum_{i=1}^{r}\xi_i\nabla H_i(\boldsymbol{z})\\
+ \sum_{i=1}^{r}\omega_i\left[G_i(\boldsymbol{z})\nabla H_i(\boldsymbol{z}) + H_i(\boldsymbol{z})\nabla G_i(\boldsymbol{z})\right]\Big\| \leq \epsilon,
\end{aligned} \tag{SM3.3a}$$

$$\begin{aligned}
g_i(\boldsymbol{z}) \geq 0, \quad \lambda_i \geq 0, \quad & g_i(\boldsymbol{z})\lambda_i = 0, \quad && \text{for } i = 1,\dots,p,\\
h_i(\boldsymbol{z}) = 0, \quad \mu_i \text{ free}, \quad & && \text{for } i = 1,\dots,q,\\
G_i(\boldsymbol{z}) \geq 0, \quad \nu_i \geq 0, \quad & G_i(\boldsymbol{z})\nu_i = 0, \quad && \text{for } i = 1,\dots,r,\\
H_i(\boldsymbol{z}) \geq 0, \quad \xi_i \geq 0, \quad & H_i(\boldsymbol{z})\xi_i = 0, \quad && \text{for } i = 1,\dots,r,\\
[\tau - G_i(\boldsymbol{z})H_i(\boldsymbol{z})] \geq 0, \quad \omega_i \geq 0, \quad & [\tau - G_i(\boldsymbol{z})H_i(\boldsymbol{z})]\,\omega_i = 0, \quad && \text{for } i = 1,\dots,r.
\end{aligned} \tag{SM3.3b}$$

In the following subsection we prove that a sequence of such ϵ-KKT points generated by Algorithm 2, under certain assumptions, converges to a C-stationary point of the original MPCC problem.

SM3.3 Proof of theorem 3.4

Proof For each iterate $\boldsymbol{z}^t$, since it is an ϵ-KKT point of $(\mathcal{R}_{\tau^t})$, there exists a vector of Lagrange multipliers $\left(\boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \boldsymbol{\nu}^t, \boldsymbol{\xi}^t, \boldsymbol{\omega}^t\right) \in \mathbb{R}^{(p+q+3r)}$ satisfying (SM3.3a-e). Consider the following constructions:

$$\hat{\nu}_i^t \coloneqq \begin{cases} \nu_i^t & \text{if } i \in I^G(\boldsymbol{z}^t) \cup I^{GH}(\boldsymbol{z}^t), \\ -\omega_i^t H_i(\boldsymbol{z}^t) & \text{if } i \in \left\{i : G_i(\boldsymbol{z}^t)H_i(\boldsymbol{z}^t) = \tau^t\right\}, \\ 0 & \text{otherwise,} \end{cases}$$

$$\hat{\xi}_i^t \coloneqq \begin{cases} \xi_i^t & \text{if } i \in I^H(\boldsymbol{z}^t) \cup I^{GH}(\boldsymbol{z}^t), \\ -\omega_i^t G_i(\boldsymbol{z}^t) & \text{if } i \in \left\{i : G_i(\boldsymbol{z}^t)H_i(\boldsymbol{z}^t) = \tau^t\right\}, \\ 0 & \text{otherwise.} \end{cases}$$

This allows us to rewrite equation (SM3.3a) as

$$\left\| \nabla f(\boldsymbol{z}^t) - \sum_{i=1}^{p} \lambda_i^t \nabla g_i(\boldsymbol{z}^t) - \sum_{i=1}^{q} \mu_i^t \nabla h_i(\boldsymbol{z}^t) - \sum_{i=1}^{r} \hat{\nu}_i^t \nabla G_i(\boldsymbol{z}^t) - \sum_{i=1}^{r} \hat{\xi}_i^t \nabla H_i(\boldsymbol{z}^t) \right\| \leq \epsilon^t. \tag{SM3.4}$$

We also note that for each $i \in \left\{i : G_i(\boldsymbol{z}^t)H_i(\boldsymbol{z}^t) = \tau^t\right\}$, both $\hat{\nu}_i^t$ and $\hat{\xi}_i^t$ are non-positive, while for each $i \in I^G(\boldsymbol{z}^t) \cup I^{GH}(\boldsymbol{z}^t)$, both $\hat{\nu}_i^t$ and $\hat{\xi}_i^t$ are non-negative. For all other indices, they are zero. Therefore, we can conclude that

$$\hat{\nu}_i^t \hat{\xi}_i^t \geq 0 \quad \forall i \in \{1, ..., r\}. \tag{SM3.5}$$

We now wish to prove that the sequence $\boldsymbol{a}_t \coloneqq \left(\boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \hat{\boldsymbol{\nu}}^t, \hat{\boldsymbol{\xi}}^t\right)_{t\in\mathbb{N}}$ has a convergent subsequence. Suppose towards a contradiction that it does not. It must be that $\boldsymbol{a}_t$ is unbounded and $\|\boldsymbol{a}_t\| \to \infty$. Dividing equation (SM3.4) through by $\|\boldsymbol{a}_t\|$ leads to

$$\left\| \frac{\nabla f(\boldsymbol{z}^t)}{\|\boldsymbol{a}_t\|} - \sum_{i=1}^{p} \frac{\lambda_i^t}{\|\boldsymbol{a}_t\|} \nabla g_i(\boldsymbol{z}^t) - \sum_{i=1}^{q} \frac{\mu_i^t}{\|\boldsymbol{a}_t\|} \nabla h_i(\boldsymbol{z}^t) \right.$$
$$\left. - \sum_{i=1}^{r} \frac{\hat{\nu}_i^t}{\|\boldsymbol{a}_t\|} \nabla G_i(\boldsymbol{z}^t) - \sum_{i=1}^{r} \frac{\hat{\xi}_i^t}{\|\boldsymbol{a}_t\|} \nabla H_i(\boldsymbol{z}^t) \right\| \leq \frac{\epsilon^t}{\|\boldsymbol{a}_t\|}. \tag{SM3.6}$$

Now consider the normed sequence $\frac{(\boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \hat{\boldsymbol{\nu}}^t, \hat{\boldsymbol{\xi}}^t,)}{\|(\boldsymbol{\lambda}^t, \boldsymbol{\mu}^t, \hat{\boldsymbol{\nu}}^t, \hat{\boldsymbol{\xi}}^t)\|}$. Clearly, it is bounded. It follows from the Bolzano–Weierstrass theorem that there exists some subsequence that converges. Let the point $\left(\boldsymbol{\lambda}', \boldsymbol{\mu}', \hat{\boldsymbol{\nu}}', \hat{\boldsymbol{\xi}}'\right)$ be the limit of that subsequence. We know this must be non-zero; otherwise, the original sequence $\boldsymbol{a}_t$ would also have a subsequence converging to zero. From the complementarity slackness conditions in (SM3.3b) we conclude that: λ_i' is non-zero only for $i \in I^g$ and $\hat{\nu}_i'$ is non-zero only for $i \in I^G \cup I^{GH}$ and $\hat{\xi}_i'$ is non-zero only for $i \in I^H \cup I^{GH}$. Using this indexing we can say that it must be the $\left(\lambda'_{I^g}, \boldsymbol{\mu}', \nu'_{I^{G}\cup I^{GH}}, \xi'_{I^{H}\cup I^{GH}}\right)$ components of the sequence that are non-zero.

Returning to (SM3.6), as $t \to \infty$, the terms $\frac{\nabla f(\boldsymbol{z}^t)}{\|\boldsymbol{a}_t\|}$ and $\frac{\epsilon^t}{\|\boldsymbol{a}_t\|}$ go to zero, and we are left with

$$\sum_{i\in I^g} \lambda_i' \nabla g_i(\boldsymbol{z}^*) + \sum_{i\in I^h} \mu_i' \nabla h_i(\boldsymbol{z}^*) + \sum_{i\in I^G\cup I^{GH}} \hat{\nu}_i' \nabla G_i(\boldsymbol{z}^*) + \sum_{i\in I^H\cup I^{GH}} \hat{\xi}_i' \nabla H_i(\boldsymbol{z}^*) = \boldsymbol{0}. \tag{SM3.7}$$

From the assumptions of the theorem, we have that $\boldsymbol{z}^*$ satisfies MPCC-MFCQ-R. By Lemma 2.4, the set of gradient vectors in (2.6) should be positively linearly independent. But that is contradicted by (SM3.7). Therefore, we conclude $\boldsymbol{a}_t$ must have a convergent subsequence. Let $\left(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \hat{\boldsymbol{\nu}}^*, \hat{\boldsymbol{\xi}}^*\right)$ be the limit of that convergent subsequence. By the continuous differentiability of f, g, h, G, H, and since the sequence $\left(\boldsymbol{z}^t\right)_{t\in\mathbb{N}}$ converges in their domain, these functions and their derivatives must converge in range. We can apply all this to (SM3.3a) to get the following:

$$\nabla f(\boldsymbol{z}^*) - \sum_{i=1}^{p} \lambda_i^* \nabla g_i(\boldsymbol{z}^*) - \sum_{i=1}^{q} \mu_i^* \nabla h_i(\boldsymbol{z}^*) - \sum_{i=1}^{r} \hat{\nu}_i^* \nabla G_i(\boldsymbol{z}^*) - \sum_{i=1}^{r} \hat{\xi}_i^* \nabla H_i(\boldsymbol{z}^*) = 0. \tag{SM3.8}$$

From equations (SM3.8), (SM3.3b) and (SM3.5), we can conclude that (2.8a), (2.8b) and (2.8C) hold, respectively. These are exactly the C-stationary conditions. □

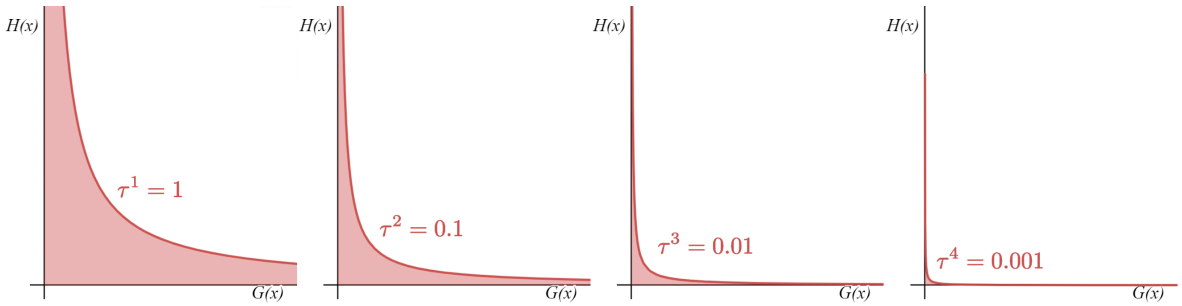

Fig. SM2 The feasible set for the relaxation subproblems with (left to right) decreasing relaxation parameters.

SM3.4 Semismooth Newton method

During our experiments, we compared to the Semi-smooth Newton method. This method writes the M-stationary conditions of problem MPCC as a system of semismooth equations, which can be solved with a Newton-style method. This framework is proposed in [74]. It requires, in the first instance, the use of an NCP function that, given inputs a, b returns zero if and only if $a, b \geq 0$ and $ab = 0$. Two examples of such functions are the minimum and Fischer-Burmeister functions defined by

$$\min(a,b) := \begin{cases} a \text{ if } a \leq b, \\ b \text{ otherwise;} \end{cases} \qquad \text{fb}(a,b) := \sqrt{a^2+b^2} - a - b.$$

The second step of building the framework consists of introducing the M-stationarity function $\text{msf} : \mathbb{R}^4 \to \mathbb{R}^2$, which has the property that its zero level set $\{(G_i, H_i, \nu_i, \xi_i) : \text{msf}(G_i, H_i, \nu_i, \xi_i) = 0\}$ is equal to the set of points that satisfy the M-stationary conditions; i.e.,

$$\text{msf}(G_i, H_i, \nu_i, \xi_i) = 0 \quad \Longleftrightarrow \quad \begin{cases} 0 \leq G_i \perp H_i \geq 0, \\ G_i \nu_i = 0, \\ H_i \xi_i = 0, \\ (\nu_i, \xi_i \geq 0 \text{ or } \nu_i \xi_i = 0)\,. \end{cases} \tag{SM3.9}$$

For an example of such a function, see [74, p. 1470, Section 3.2]. The full system of equations to be solved to compute M-stationary points for problem MPCC becomes

$$F(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi}) := \begin{bmatrix} \nabla_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\xi}) \\ [\text{fb}(g_i(\boldsymbol{z}), \lambda_i)]_{i=1,\ldots,p} \\ [h_i(\boldsymbol{z})]_{i=1,\ldots,q} \\ [\text{msf}(G_i(\boldsymbol{z}), H_i(\boldsymbol{z}), \nu_i, \xi_i)]_{i=1,\ldots,r} \end{bmatrix} = 0 \tag{SM3.10}$$

with the function F being semismooth (see [74] and references therein for the definition), while $\mathcal{L}$ is the Lagrangian function as defined in (2.7). Note that here, the fb function may be replaced by min or any other NCP function.

It can easily be shown that $(\boldsymbol{z}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*, \boldsymbol{\xi}^*)$ solves equation (SM3.10) if and only if $\boldsymbol{z}^*$ is an M-stationary point of the MPCC with Lagrange multipliers $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*, \boldsymbol{\xi}^*)$. To solve the system, we use a semismooth Newton Method with an Armijo line search and the Newton derivative to address the involved non-smoothness, according to [74, Algorithm 5.1].

The system of equations, when applied to our SVM hyperparameter tuning model ($\mathcal{M}$), is

$$F_{\mathcal{M}}(\boldsymbol{z}, \boldsymbol{\Lambda}) = \begin{bmatrix} \nabla_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) \\ \text{fb}(C, \Lambda^C) \\ \text{fb}(\gamma, \Lambda^\gamma) \\ \left[\text{fb}\left(\zeta_i^k, \Lambda^\zeta_{(k,i)}\right)\right]_{(k,i)\in\bar{I}} \\ \left[\text{fb}\left(Z(\boldsymbol{z})_i^k, \Lambda^Z_{(k,i)}\right)\right]_{(k,i)\in\bar{I}} \\ \left[\theta(\boldsymbol{z})_i^k\right]_{(k,i)\in I} \\ \left[\varphi(\boldsymbol{z})^k\right]_{k=1,\ldots,K} \\ \left[\text{msf}\left(\alpha_i^k, \underline{v}_i^k, \Lambda^{G_1}_{(k,i)}, \Lambda^{H_1}_{(k,i)}\right)\right]_{(k,i)\in I} \\ \left[\text{msf}\left(C - \alpha_i^k, \bar{v}_i^k, \Lambda^{G_2}_{(k,i)}, \Lambda^{H_2}_{(k,i)}\right)\right]_{(k,i)\in I} \end{bmatrix} \tag{SM3.11}$$

where fb : $\mathbb{R}^2 \to \mathbb{R}$ is the Fischer Burmeister function and msf : $\mathbb{R}^4 \to \mathbb{R}^2$ is the M-stationarity function, the derivatives of the Lagrangian are written explicitly in Subsection SM3.5 and the functions Z, θ, φ are written in (4.2a,b,c).

SM3.5 Explicit formula of the Lagrangian and its derivatives

Early in the paper, we dealt with a compact form of the MPCC problem ($\mathcal{M}$). However, when it comes to implementing solution methods, we need accurate formulas for the Lagrangian and its first and second-order derivatives. We use the notation defined in equations (4.1) and (4.2). In this supplementary material, we present those explicit formulas in full, beginning with the Lagrangian

$$\begin{aligned} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) = &\sum_{(k,i)\in\bar{I}} \zeta_i^k - \Lambda^C C - \Lambda^\gamma \gamma - \sum_{(k,i)\in\bar{I}} \Lambda^\zeta_{(k,i)} \zeta_i^k - \sum_{(k,i)\in\bar{I}} \Lambda^Z_{(k,i)} Z(\boldsymbol{z})_i^k - \sum_{(k,i)\in I} \Lambda^\theta_{(k,i)} \theta(\boldsymbol{z})_i^k \\ &- \sum_{(k,i)\in I} \Lambda^{G_1}_{(k,i)} \alpha_i^k - \sum_{(k,i)\in I} \Lambda^{H_1}_{(k,i)} \underline{v}_i^k - \sum_{(k,i)\in I} \Lambda^{G_2}_{(k,i)} (C - \alpha_i^k) - \sum_{(k,i)\in I} \Lambda^{H_2}_{(k,i)} \bar{v}_i^k - \sum_{k=1,\ldots,K} \Lambda^\varphi_k \varphi(\boldsymbol{z})^k; \end{aligned} \tag{SM3.12}$$

where the decision variables and Lagrange multipliers are

$$\begin{aligned} \boldsymbol{z} &= [C, \gamma, \zeta, \boldsymbol{\alpha}, \underline{\boldsymbol{v}}, \bar{\boldsymbol{v}}, u] \in \mathbb{R}^{3(K-1)n+n+K+2}; \\ \boldsymbol{\Lambda} &= \left[\Lambda^C, \Lambda^\gamma, \boldsymbol{\Lambda}^\zeta, \boldsymbol{\Lambda}^Z, \boldsymbol{\Lambda}^\theta, \boldsymbol{\Lambda}^{G_1}, \boldsymbol{\Lambda}^{H_1}, \boldsymbol{\Lambda}^{G_2}, \boldsymbol{\Lambda}^{H_2}, \boldsymbol{\Lambda}^\varphi, \right] \in \mathbb{R}^{5(K-1)n+2n+K+2}; \end{aligned}$$

with dimensions in terms of the total number of data points n and the number of folds K.

The first order derivative with respect to all input variables is a vector to vector function $\nabla\mathcal{L} : \mathbb{R}^L \to \mathbb{R}^L$ where $L = 8(K-1)n + 3n + 2K + 4$. We define it in terms of each partial derivative.

$$
\begin{aligned}
\nabla_C \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\Lambda^C - \sum_{k=1}^{K}\sum_{i=1}^{\bar{n}^k} \Lambda^Z_{(k,i)} \bar{y}_i^k \nabla_C b - \sum_{k=1}^{K}\sum_{i=1}^{n^k} \Lambda^{G_2}_{(k,i)};\\
\nabla_\gamma \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\Lambda^\gamma + \sum_{k=1}^{K}\sum_{i=1}^{\bar{n}^k} \Lambda^Z_{(k,i)} \left(\sum_{j=1}^{n^k} \alpha_j^k \left\| \bar{x}_i^k - x_j^k \right\|^2 \bar{Q}(\gamma)_{ij}^k - \bar{y}_i^k \nabla_\gamma b \right)\\
&+ \sum_{k=1}^{K}\sum_{i=1}^{n^k} \Lambda^\theta_{(k,i)} \left(\sum_{j=1}^{n} \alpha_j^k \left\| x_i^k - x_j^k \right\|^2 Q(\gamma)_{ij}^k \right);\\
\nabla_{\zeta_i^k} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& 1 - \Lambda^\zeta_{(k,i)} - \Lambda^Z_{(k,i)};\\
\nabla_{\alpha_j^k} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\sum_{i=1}^{\bar{n}^k} \Lambda^Z_{(k,i)} \left(\bar{Q}(\gamma)_{ij}^k + \bar{y}_i^k \nabla_{\alpha_j^k} b \right) - \sum_{i=1}^{n^k} \Lambda^\theta_{(k,i)} Q(\gamma)_{ij}^k - \Lambda^{G_1}_{(k,j)} + \Lambda^{G_2}_{(k,j)}, -\Lambda^\varphi_k y_j^k;\\
\nabla_{\underline{v}_i^k} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& +\Lambda^\theta_{(k,i)} - \Lambda^{H_1}_{(k,i)};\\
\nabla_{\bar{v}_i^k} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\Lambda^\theta_{(k,i)} - \Lambda^{H_2}_{(k,i)};\\
\nabla_{u^k} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\sum_{i=1}^{n^k} \Lambda^\theta_{(k,i)} y_i^k.
\end{aligned}
$$

The Hessian of the Lagrangian is:

$$
\nabla^2_{zz} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) = \begin{bmatrix}
\nabla^2_{CC}\mathcal{L} & \nabla^2_{C\gamma}\mathcal{L} & 0 & \nabla^2_{C\boldsymbol{\alpha}}\mathcal{L} & 0 & 0 & 0 \\
\nabla^2_{\gamma C}\mathcal{L} & \nabla^2_{\gamma\gamma}\mathcal{L} & 0 & \nabla^2_{\gamma\boldsymbol{\alpha}}\mathcal{L} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\nabla^2_{\boldsymbol{\alpha} C}\mathcal{L} & \nabla^2_{\boldsymbol{\alpha}\gamma}\mathcal{L} & 0 & \nabla^2_{\boldsymbol{\alpha}\boldsymbol{\alpha}}\mathcal{L} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

The partial second-order derivatives can be written explicitly as:

$$
\begin{aligned}
\nabla^2_{CC} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\sum_{k=1}^{K}\sum_{i=1}^{\bar{n}^k} \Lambda^Z_{(k,i)} \bar{y}_i^k \nabla^2_{CC} b\left(\boldsymbol{\alpha}^k, \gamma, C\right);\\
\nabla^2_{\gamma\gamma} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\sum_{k=1}^{K}\sum_{i=1}^{\bar{n}^k} \Lambda^Z_{k,i} \left(\sum_{j=1}^{n^k} \alpha_j^k \left\| \bar{x}_i^k - x_j^k \right\|^4 \bar{Q}(\gamma)_{ij}^k + \bar{y}_i^k \nabla^2_{\gamma\gamma} b \right) - \sum_{k=1}^{K}\sum_{i=1}^{n^k} \Lambda^\theta_{k,i} \sum_{j=1}^{n^k} \alpha_j^k \left\| x_i^k - x_j^k \right\|^4 Q(\gamma)_{ij}^k;\\
\nabla^2_{\alpha_j^k \alpha_l^k} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& \sum_{i=1}^{\bar{n}^k} \Lambda^Z_{(k,i)} \bar{y}_i^k \nabla^2_{\alpha_j^k \alpha_l^k} b\left(\boldsymbol{\alpha}^k, \gamma, C\right);\\
\nabla^2_{C\gamma} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\sum_{k=1}^{K}\sum_{i=1}^{\bar{n}^k} \Lambda^Z_{(k,i)} \bar{y}_i^k \nabla^2_{C\gamma} b\left(\boldsymbol{\alpha}^k, \gamma, C\right);\\
\nabla^2_{C\alpha_l^k} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\Lambda}) =& -\sum_{i=1}^{\bar{n}^k} \Lambda^Z_{(k,i)} \bar{y}_i^k \nabla^2_{C\alpha_l^k} b\left(\boldsymbol{\alpha}^k, \gamma, C\right);
\end{aligned}
$$

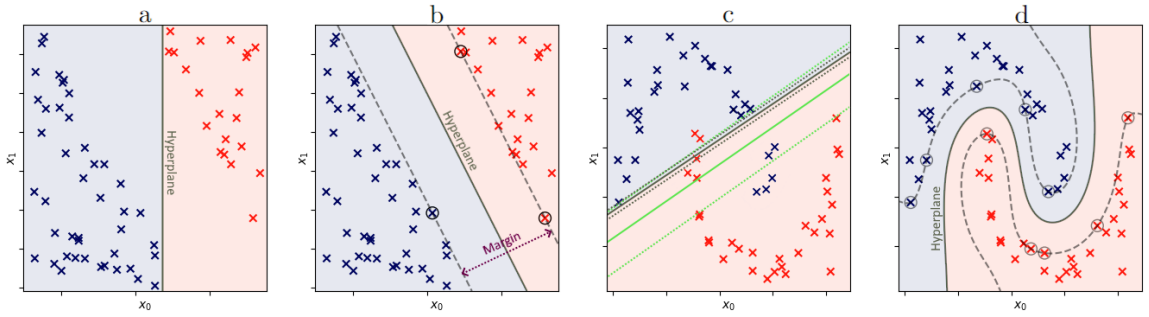

Fig. SM3 Each of the four plots show blue and red data points in a two-dimensional space. Plots a and b (left) are the same linearly separable data set. Plots c and d (right) are sampled from the Moons data set. The first plot shows a naive separating hyperplane. Plot b shows a more intelligent hyperplane with a maximum margin found using the (Hard-Margin-SVM). Plot c shows two possible hyperplanes in black and green found by the (Soft-Margin-SVM) with different choices of hyperparameter C. Plot d shows a nonlinear boundary with a maximum margin. We found this using the (Dual-SVM) attached with a RBF kernel.

$$\nabla^2_{\gamma\alpha^k_j}\mathcal{L}(\boldsymbol{z},\boldsymbol{\Lambda}) = \sum_{i=1}^{\bar{n}^k} \Lambda^{Z}_{(k,i)} \left(\left\| \bar{x}^k_i - x^k_j \right\|^2 \bar{Q}(\gamma)^k_{ij} - \bar{y}^k_i \nabla^2_{\alpha^k_j\gamma} b \right) + \sum_{i=1}^{n} \Lambda^{\theta}_{(k,i)} \left\| x^k_i - x^k_j \right\|^2 Q(\gamma)^k_{ij}.$$

SM4 Preliminaries on the Support Vector Machine (SVM)

The concept of support vector machine (SVM) was introduced by Cortes and Vapnik [23, 24] and has since been a prolific research subject; see, e.g., [75]. Here, we provide an introduction to the topic as well as a setup of notation that is used throughout the final half of the paper.

SM4.1 Primal SVM

The SVM is defined by a set of n training examples, denoted by $(\boldsymbol{x}_i, y_i)$ for $i = 1, \ldots, n$, such that each example i is a vector of features collected by $\boldsymbol{x}_i \in \mathbb{R}^d$ and a class label $y_i \in \{-1, +1\}$, where the aim is to find weights $\boldsymbol{w} \in \mathbb{R}^d$ and a bias $b \in \mathbb{R}$ describing a hyperplane $\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\top \boldsymbol{x} + b = 0\}$ and a margin around it, $\{\boldsymbol{x} \in \mathbb{R}^d : -1 \leq \boldsymbol{w}^\top \boldsymbol{x} + b \leq 1\}$. The weights should be such that the positive label points lie above the margin ($\boldsymbol{w}^\top \boldsymbol{x}_i + b \geq 1 \quad \forall i : y_i = +1$) and the negative label points lie below the margin ($\boldsymbol{w}^\top \boldsymbol{x}_i + b \leq -1 \quad \forall i : y_i = -1$). This is equivalent to the constraint $1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \leq 0$ for any i. Under these constraints, our goal is to maximise the width of the margin given by $\frac{2}{\|\boldsymbol{w}\|}$. Intuitively, this means new unseen data points have the most leeway to deviate beyond the range of the training data without crossing to the wrong side of the hyperplane; see Figure SM3a-b. Once optimal weights $\boldsymbol{w}^*$ and bias b^* are found, we can predict the label of new data using the classifier $\boldsymbol{x} \mapsto \text{sign}\left(\boldsymbol{w}^{*\top} \boldsymbol{x} + b^*\right)$ where the sign function returns -1 for negative inputs and +1 otherwise. This problem can be formulated as

$$\begin{aligned} &\underset{\boldsymbol{w}\in\mathbb{R}^d, b\in\mathbb{R}}{\text{minimise}} && \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} \\ &\text{subject to} && 1 - y_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) \leq 0 \quad \text{for } i = 1, \ldots, n. \end{aligned} \tag{Hard-Margin-SVM}$$

The weakness of the (Hard-Margin-SVM) becomes apparent when data is not linearly separable; the above problem is then infeasible. Thus, the (Hard-Margin-SVM) is rarely used in practice. A far more common approach is to use the hinge loss function

$$\text{HingeLoss}(\boldsymbol{x}_i, y_i) := \max\left(0, 1 - y_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right)\right),$$

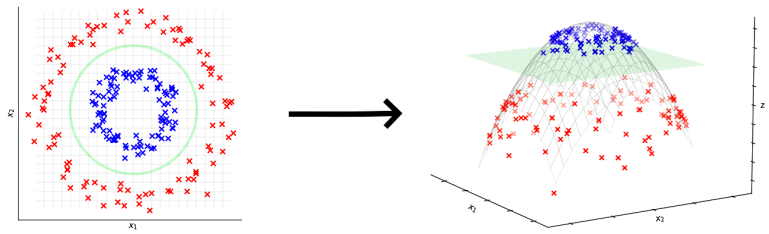

Fig. SM4 On the left, data in two dimensions that is not linearly separable is shown. On the right, the same data is lifted into a third dimension, where a separating hyperplane exists.

which measures how much example $(\boldsymbol{x}_i, y_i)$ violates the margin constraint for a given weight and bias. It takes the value zero when the constraint is satisfied. By replacing each constraint with a hinge loss penalty in the objective function, we can form a model that discourages but allows some points to violate the constraint in favour of ensuring a large margin. A regularisation parameter C can be introduced to instruct the balance between these two objectives, with a large C ensuring that the margin constraint is less violated, whereas a smaller C is more tolerant to such violations where it can find a larger margin. The new objective is now to minimise the function

$$\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{n} \max\left(0, 1 - y_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right)\right).$$

The non-smooth max operator in this new objective function can be eliminated by introducing a variable ξ_i for each example $(\boldsymbol{x}_i, y_i)$. This leads to the soft margin formulation of the SVM problem

$$\begin{array}{lll} \underset{\boldsymbol{w}\in\mathbb{R}^d, \boldsymbol{\xi}\in\mathbb{R}^n, b\in\mathbb{R}}{\text{minimise}} & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{n} \xi_i & \\ \text{subject to} & \xi_i \geq 1 - y_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) & \text{for } i = 1, \ldots, n, \\ & \xi_i \geq 0 & \text{for } i = 1, \ldots, n. \end{array} \qquad \text{(Soft-Margin-SVM)}$$

Now, we wish to extend this model so that it can find nonlinear boundaries to separate data. To achieve this, we consider a mapping $\boldsymbol{x}_i \mapsto \phi(\boldsymbol{x}_i)$ that lifts the features of each data point $\boldsymbol{x}_i$, for $i = 1, \ldots, n$, to a higher dimension. We can then find a hyperplane in this higher dimension, which corresponds to a nonlinear boundary in the original feature space; see Figure SM4 for an illustration. However, it is often computationally hard to compute $\phi(\boldsymbol{x}_i)$. The solution is the so-called kernel trick, which allows us to find a separating hyperplane in the higher dimension without ever explicitly evaluating $\phi(\boldsymbol{x}_i)$. In our numerical experiments, we use the Radial Basis Function (RBF), which is defined by

$$K_\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2) \coloneqq \exp\left(-\gamma \left\|\boldsymbol{x}_1 - \boldsymbol{x}_2\right\|^2\right) \qquad \text{(SM4.1)}$$

(with hyperparameter γ) and maps features to an infinite-dimensional space, making the kernel trick necessary. Other kernel functions commonly used in the literature [75, 76] include the linear, polynomial, and Sigmoid ones, respectively given as follows (for hyperparameters γ, δ):

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) \coloneqq \boldsymbol{x}_1^\top \boldsymbol{x}_2, \quad K_{\gamma,\delta}(\boldsymbol{x}_1, \boldsymbol{x}_2) \coloneqq (\gamma \boldsymbol{x}_1^\top \boldsymbol{x}_2 + \delta)^2, \quad \text{and} \quad K_{\gamma,\delta}(\boldsymbol{x}_1, \boldsymbol{x}_2) \coloneqq \tanh\left(\gamma \boldsymbol{x}_1^\top \boldsymbol{x}_2 + \delta\right).$$

To apply the kernel trick, we need to work with the dual of the soft margin SVM optimisation problem. This was first discovered by Guyon, Boser and Vapnik in 1992 [76, 77] and is now standard. Nevertheless, for the paper to be self-contained, in the following Subsection SM4.2, we provide the details of the conversion from the primal problem (Soft-Margin-SVM) to the dual (Dual-SVM).

SM4.2 Dual SVM

The first step in dualising the SVM is to write the Lagrangian function $\mathcal{L} : \mathbb{R}^{d+3n+1} \to \mathbb{R}$ of the primal program (Soft-Margin-SVM). This is the sum of the original objective function plus each constraint scaled by a corresponding Lagrange multipliers $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $\boldsymbol{\eta} \in \mathbb{R}^n$. For the textbook theory on Lagrange functions, see Boyd and Vandenberghe [78, Section 5.1.2].

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\eta}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i \Big(1 - \xi_i - y_i(\boldsymbol{w}\phi(\boldsymbol{x}_i) + b)\Big) - \sum_{i=1}^{n} \eta_i \xi_i \tag{SM4.2}$$

We note three properties of the primal problem (Soft-Margin-SVM). First, the objective function is convex (in particular, positive quadratic). Secondly, the constraints are each linear in $\boldsymbol{w}, \boldsymbol{\xi}, b$, and so the feasible set is an (unbounded) convex polyhedron. Thirdly the feasible set has a non-empty interior since we can pick $\xi_i = \max\big(0, 1 - y_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right)\big) + 10^{-6}$ which strictly satisfies each inequality. Therefore the problem satisfies Slater's condition and strong duality. Any feasible point $\boldsymbol{w}, \boldsymbol{\xi}, b$ is a local minimiser of (Soft-Margin-SVM) (equivalent to a global minimiser by convexity) only if there exist Lagrange multipliers $\boldsymbol{\alpha}, \boldsymbol{\eta}$ that satisfy the following first-order stationarity conditions

$$\nabla_w \mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\eta}) = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) = \mathbf{0}, \tag{SM4.3a}$$

$$\nabla_b \mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\eta}) = \sum_{i=1}^{n} \alpha_i y_i = 0, \tag{SM4.3b}$$

$$\nabla_{\xi_i} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\eta}) = C\mathbf{1} - \boldsymbol{\alpha} - \boldsymbol{\eta} = \mathbf{0}. \tag{SM4.3c}$$

We now manipulate the Lagrangian (SM4.2) by substituting in the stationary conditions (SM4.3a, SM4.3b, SM4.3c). This will give us a function of the Lagrangian in terms of $\boldsymbol{\alpha}, \boldsymbol{\eta}$ assuming optimal choices for $\boldsymbol{w}, \boldsymbol{\xi}, b$. The term $\sum_{i=1}^{n}(C - \alpha_i - \eta_i)\xi_i$ vanishes due to (SM4.3c). The term $\sum_{i=1}^{n} \alpha_i y_i b$ vanishes due to (SM4.3b). And we eliminate $\boldsymbol{w}$ by substituting in (SM4.3a). The conversion of the problem from primal space variables ($\boldsymbol{w}$, $\boldsymbol{\xi}$ and b) to dual space variables ($\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$) like this allows us to apply the kernel trick $\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) = K_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Below, we work through the manipulation line by line.

$$\begin{aligned}
& \mathcal{L}^*(\boldsymbol{\alpha}, \boldsymbol{\eta}) \\
= \;& \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i \Big(1 - \xi_i - y_i(\boldsymbol{w}\phi(\boldsymbol{x}_i) + b)\Big) - \sum_{i=1}^{n} \eta_i \xi_i \\
= \;& \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} \quad + \sum_{i=1}^{n}(C - \alpha_i - \eta_i)\xi_i \quad + \sum_{i=1}^{n} \alpha_i \quad - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{w}\phi(\boldsymbol{x}_i) \quad - \sum_{i=1}^{n} \alpha_i y_i b \\
= \;& \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} \quad + 0 \qquad\qquad\quad + \sum_{i=1}^{n} \alpha_i \quad - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{w}\phi(\boldsymbol{x}_i) \quad - 0 \\
= \;& \frac{1}{2}\Big(\sum_{i=1}^{n} \alpha_i y_i \phi(\boldsymbol{x}_i)\Big)^\top \Big(\sum_{j=1}^{n} \alpha_j y_j \phi(\boldsymbol{x}_j)\Big) \quad + \sum_{i=1}^{n} \alpha_i \quad - \sum_{i=1}^{n} \alpha_i y_i \Big(\sum_{j=1}^{n} \alpha_j y_j \phi(\boldsymbol{x}_j)\Big)\phi(\boldsymbol{x}_i) \\
= \;& \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) \qquad + \sum_{i=1}^{n} \alpha_i \quad - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) \\
= \;& -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j) \qquad + \sum_{i=1}^{n} \alpha_i
\end{aligned}$$

By considering the infimum of the $\{\mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\eta})\}$ (SM4.2) we now write the dual of (Soft-Margin-SVM). For the textbook duality theory of convex programs, see Boyd and Vandenberghe [78, Section 5.2].

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^n, \boldsymbol{\eta}\in\mathbb{R}^n}{\text{maximise}} \quad & \inf_{\boldsymbol{w},\boldsymbol{\xi}\in\mathbb{R}^n, b\in\mathbb{R}} \Big\{\mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\eta})\Big\} \\ \text{subject to} \quad & 0 \leq \alpha_i \quad \text{for } i = 1, \ldots, n, \\ & 0 \leq \boldsymbol{\eta} \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

The above is impractical for us. We replace the infimum of the Lagrangian ($\inf\{\mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \boldsymbol{\eta})\}$) with the expression $\mathcal{L}^*(\boldsymbol{\alpha}, \boldsymbol{\eta})$ derived from the stationary conditions above. Next, we multiply by -1 to convert from maximisation to a minimisation problem. Finally, the constraints $\boldsymbol{\alpha} \geq 0, \quad \boldsymbol{\eta} \geq 0$ can be rearranged to $0 \leq \boldsymbol{\alpha} \leq C$ by substituting out $\eta_i = C - \alpha_i$ according to (SM4.3c). Thus, the dual problem can be written as follows

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{minimise}} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j K_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n}\alpha_i \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \ldots, n, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0, \end{aligned} \tag{Dual-SVM}$$

which will be the main focus of our attention for the remainder of this section.

The (Dual-SVM) is highly effective at learning to separate real-world data. Many machine-learning libraries have become proficient at solving this; one of the most widely used is the open-source library LIBSVM [79]. At this point, it is worth discussing the link between the primal variable $(\boldsymbol{w}, b)$ and dual variable $\boldsymbol{\alpha}$. For each index i, if $\alpha_i = 0$, the solution represents a boundary that ensures data point i lies on the correct side of the margin. If $\alpha_i > 0$, then the data point i lies within the margin or on the wrong side. Details of how to reconstruct the primal variables and further explanations are provided in Subsection SM4.4.

SM4.3 KKT system of the dual SVM

We conclude this subsection by stating the Karush-Kuhn-Tucker (KKT) conditions for problem (Dual-SVM), which will be useful later. For the theory on KKT conditions, we refer again to Boyd and Vandenberghe [78, Section 5.5.3]. Since the program is convex and the constraints are linear, a point $\boldsymbol{\alpha}$ is optimal if and only if there exist Lagrange multipliers $\underline{\boldsymbol{v}}, \bar{\boldsymbol{v}} \in \mathbb{R}^n, u \in \mathbb{R}$ such that

$$\begin{aligned} & \sum_{j=1}^{n} \alpha_j Q(\gamma)_{ij} - \underline{v}_i + \bar{v}_i + u y_i = 1 \quad && \text{for } i = 1, \ldots, n, \\ & 0 \leq \alpha_i \perp \underline{v}_i \geq 0 && \text{for } i = 1, \ldots, n, \\ & 0 \leq (C - \alpha_i) \perp \bar{v}_i \geq 0 && \text{for } i = 1, \ldots, n, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0, \end{aligned} \tag{KKT-SVM}$$

where $Q(\gamma)_{ij} \coloneqq y_i y_j K_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i = 1, \ldots, n$, $j = 1, \ldots, n$, denotes the so-called *kernel* matrix which we write as a function of γ as this will become a decision variable of our MPCC model.

SM4.4 Reconstruction of primal variables from dual solutions

The weights w and bias b are primal variables. In order to use the kernel trick, we will be solving the dual optimisation problem (Dual-SVM) in terms of dual variables $\boldsymbol{\alpha}$. This makes it less obvious what our final classifier should predict as we cannot explicitly calculate $x \mapsto \text{sign}\left(w^T\varphi(x) + b\right)$. From the stationary conditions (SM4.3a), we have

$$w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i). \tag{SM4.4}$$

Using this formula (SM4.4), we can substitute out w from the classifier. Our classifier should predict

$$x \mapsto \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \varphi(x_i)\varphi(x) + b\right).$$

We must work harder to find an expression for the bias b. Below, we write the complementarity slackness conditions for (Soft-Margin-SVM)

$$(1 - \xi_i - y_i(w^\top \phi(x_i) + b)\alpha_i = 0, \tag{SM4.5}$$
$$\xi_i \eta_i = 0. \tag{SM4.6}$$

Suppose $\alpha_i > 0$. Then by complementarity slackness (SM4.5) we must have

$$(1 - \xi_i - y_i(w^\top \phi(x_i) + b) = 0 \qquad \text{for } i \in \{1, ..., n\} : \alpha_i > 0. \tag{SM4.7}$$

Suppose $\eta_i > 0$. Notice, this is equivalent to $\alpha_i < C$ by the stationary condition (SM4.3c). Again, by complementarity slackness (SM4.6) we must have

$$\xi_i = 0 \qquad \text{for } i \in \{1, ..., n\} : \alpha_i < C. \tag{SM4.8}$$

Putting these two (SM4.7, SM4.8) together we get

$$1 - y_i\left(w^\top \phi(x_i) + b\right) = 0 \qquad \text{for } i \in \{1, ..., n\} : 0 < \alpha_i < C.$$

This gives us an important insight. If α_i takes a value between 0 and C then we can conclude $y_i(w^\top \varphi(x_i) + b) = 1$. Geometrically, this means that example i sits exactly on the edge of the margin. We call these "support vectors", and they are our classifier's most crucial data points. Now recall that $y_i \in \{-1, +1\}$ so we can divide through by $\frac{1}{y_i} \equiv y_i$. This leads to

$$b = y_i - w^\top \phi(x_i) \qquad \text{for } i \in \{1, ..., n\} : 0 < \alpha_i < C.$$

We can now substitute out w use (SM4.3a) giving:

$$b = y_i - \sum_{j=1}^{m} \alpha_j y_j \phi(x_j)^\top \phi(x_i) \qquad \text{for } i \in \{1, ..., n\} : 0 < \alpha_i < C. \tag{SM4.9}$$

Theoretically, we can reconstruct the bias according to (SM4.9) from any support vector i on the margin's edge - identified by $0 < \alpha_i < C$. However, most libraries make the bias reconstruction calculation (SM4.9) for each such support vector and then take the average. With this in mind, let

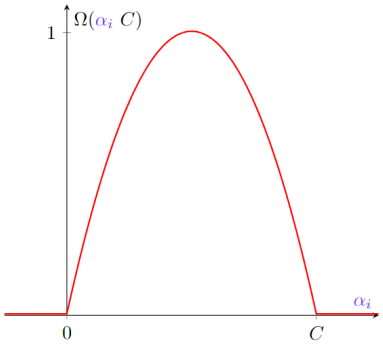

Fig. SM5 A graph of the function $\Omega(\alpha_i, C)$

$\Omega(\alpha_i, C)$ be the indicator function that equals 1 if $0 < \alpha_i < C$ and 0 otherwise. Whenever we write the bias as a function of dual variables $b(\boldsymbol{\alpha}, C)$ we mean the following reconstruction

$$b = \frac{1}{\sum_{i=1}^{n} \Omega(\alpha_i, C)} \sum_{i=1}^{n} \Omega(\alpha_i, C) \left(y_i - \sum_{j=1}^{n} \alpha_j y_j K\left(x_i, x_j\right) \right). \tag{SM4.10}$$

The algorithms we propose require a further adjustment. Firstly, because we want to avoid the combinatorial nature of a 0,1 indicator function. Secondly, it is hard to distinguish between α_i very close to zero and α_j that should be zero but carries some small numerical error. We, therefore, use an alternative indicator function as proposed by Anthony Dunn [42].

$$\Omega(\alpha_i, C) = \begin{cases} 1 - \left(\frac{2\alpha_i}{C} - 1\right)^2 + \tau & \text{if } 0 \leq \alpha_i \leq C, \\ 0 & \text{otherwise.} \end{cases} \tag{SM4.11}$$

Similar to the original indicator, this takes the value zero for $\alpha_i \notin (0, C)$ and a positive value for $\alpha_i \in (0, C)$. Importantly, it assigns a much smaller value to α_i close to 0 or C. With this new indicator, the reconstruction (SM4.10) still averages across the examples where $0 < \alpha_i < C$ and so is still theoretically correct but far more numerically stable.

SM5 Further details on the numerical experiments

To allow reproducibility of our experiments, this section of supplementary material provides full details on the starting point selection SM5.1, datasets SM5.2, and derivative free methods SM5.3 we employed.

Our experiments run on Southampton University's Iridis Compute Cluster [80]. Each experiment is scheduled on a single Intel Xeon Gold 6138 CPU. This CPU is comparable with most modern home PCs.

SM5.1 Starting point selection

In this subsection, we discuss multi-starts and initialisation strategies for the algorithms in Section 3, since each requires the user to provide a starting point $z^0 = [C^0, \gamma^0, \zeta^0, \boldsymbol{\alpha}^0, \underline{\boldsymbol{v}}^0, \bar{\boldsymbol{v}}^0, u^0]^\top$ in the context of our problem $(\mathcal{M})$. The methods are deterministic but perform very differently depending on the choice of z^0. Recall also that the penalisation and relaxation methods converge to S and C-stationary points, respectively. Such points may not be global or even local minima. Therefore, initialising the methods from a range of different starting points can increase the chance of finding one. However, it is well-known that under a MPCC-specific second-order sufficient condition [16, Definition 2.8], S-stationary points correspond to strict local minima [16, Theorem 3.1]. We remind the reader here of

	`LL-Zero`	`LL-Centre`	`LL-Feasible`
Relative computational cost	low	low	high
Feasible z^0 for $(\mathcal{M})$?	no	no	yes
Feasible z^0 for $(\mathcal{P}_{\pi^0})/(\mathcal{R}_{\tau^0})$?	no	yes	yes

Table SM1 Summary of the lower-level initialisation strategies.

the theoretical advantage that the penalisation method has by converging to S-stationary points and their potential to be local minima.

We propose a multi-start, where the upper-level variable (C^0, γ^0) shall be initiated according to a grid spread uniformly across the log space such as $\{10^n : n = -5, \dots, 4\} \times \{10^n : n = -3, \dots, 6\}$, for example. This encourages the solution paths to explore as much of the hyperparameter space as possible. Note that this grid is far coarser than the derivative-free grid search we compare against later. Figure 4 shows a two-by-four grid of starting points and the resulting path of iterates that the sequential relaxation algorithm produced for the Moons dataset. Of these, three converge to the global minimum, while the remaining five converge to other stationary points.

Considering that the variable ζ is only involved in two constraints, given an initialisation of $\boldsymbol{\alpha}^0$, we can simply choose

$$\zeta_i^{0k} = \max\left(0, 1 - \sum_{j=1}^{n^k} \alpha_j^{0k} \bar{Q}(\gamma)_{ij}^k - \bar{y}_i^k b^k\left(\boldsymbol{\alpha}^{0k}, \gamma^0, C^0\right)\right) \quad \text{for} \quad k = 1 \dots K,\ i = 1, \dots \bar{n}^k,$$

which ensures initial feasibility of the constraints $\zeta_i^{0k} \geq 0$ and $Z_i^k(z^0) \geq 0$; see (4.2a).

It remains to be discussed how to initialise the lower-level variables $\boldsymbol{\alpha}^0$, $\underline{\boldsymbol{v}}^0$, $\bar{\boldsymbol{v}}^0$ and u^0. We propose three strategies. The first and simplest strategy shall be referred to as `LL-Zero`. This strategy initialises all of the lower-level variables to zero. Notice this satisfies the complementarity constraints $0 \leq \alpha_i^k \perp \underline{v}_i^k \geq 0$ and $0 \leq (C - \alpha_i^k) \perp \bar{v}_i^k \geq 0$ but violates the stationarity constraint $\theta_i^k(z) = 0$ for each $(k, i) \in I$. The strategy has no computational cost and can perform well but must be treated with caution as it is infeasible for all the programs $(\mathcal{M})$, $(\mathcal{P}_\pi)$ and $(\mathcal{R}_\tau)$.

We call the second strategy `LL-Centre`, which, for a given choice of C^0 and γ^0, selects values for the lower-level variables as follows:

$$\alpha_i^{0k} = \frac{C^0}{2n^k} \sum_{j=1}^{n^k} (1 + y_i^k y_j^k), \qquad \underline{v}_i^{0k} = \max\left(0, \sum_{j=1}^{n^k} \alpha_j^{0k} Q(\gamma)_{ij}^k - 1 + y_i\right), \tag{SM5.1}$$

$$u^{0k} = 1, \qquad \bar{v}_i^{0k} = \max\left(0, -\sum_{j=1}^{n^k} \alpha_j^{0k} Q(\gamma)_{ij}^k + 1 - y_i\right) \tag{SM5.2}$$

for $k = 1, \dots, K$, $i = 1, \dots, n^k$. By construction, $\theta_i^k(z^0) = 0$ for $k = 1, \dots, K$, $i = 1, \dots, n^k$, and $\varphi^k(z^0) = 0$ for $k = 1, \dots, K$; see (4.2). For a balanced dataset, where there is a similar number of positive and negative labels y_i^k in each fold, the variables α_i^k are initialised close to $\frac{C}{2}$, the midpoint of its feasible interval $0 \leq \alpha_i^k \leq C$, for $k = 1, \dots, K$, $i = 1, \dots, n^k$. Since one of $\bar{\boldsymbol{v}}$ or $\underline{\boldsymbol{v}}$ is also positive, the complementarity constraints are violated. This means that the initialisation is not MPCC feasible for problem $(\mathcal{M})$ but is feasible for the corresponding penalisation $(\mathcal{P}_{\pi^0})$ and relaxation $(\mathcal{R}_{\tau^0})$ formulations for suitable choices of π^0 and τ^0.

Finally, we call the third strategy `LL-Feasible`, which corresponds, for a given choice of (C^0, γ^0), to find $\boldsymbol{\alpha}^0$, $\underline{\boldsymbol{v}}^0$, $\bar{\boldsymbol{v}}^0$ and u^0 that satisfy the KKT conditions (KKT-SVM), characterising the lower-level problem. This is essentially the task of solving a mixed system of equations with complementarity terms, as studied by Michael Ferris and Steven Dirkse [81]. In our experiments we will call the PATH solver [82–84] K times, where K is the number of folds, to solve each of the K lower-level KKT

	Penalisation (Sequential)	Relaxation (Sequential)
Parameter π/τ Q_1	10^4	10^{-6}
Parameter π/τ Q_2	10^6	10^{-6}
Parameter π/τ Q_3	10^6	10^{-6}
MPCC Feasibility Q_1	$3.94e-07$	$1.00e-06$
MPCC Feasibility Q_2	$1.93e-08$	$9.98e-07$
MPCC Feasibility Q_3	$4.88e-09$	$2.01e-07$

Table SM2 This table presents the observed distributions of final parameter π and τ as well as the MPCC feasibility at termination. Q_1, Q_2, Q_3 denote the first quartile, median and third quartile.

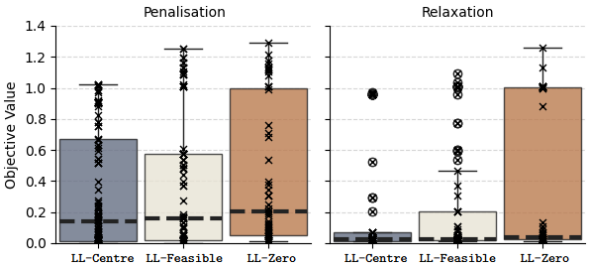

Fig. SM6 These box plots summarise the distribution of objective values achieved when using each of the three different lower-level initialisation strategies for the Circles dataset.

systems. It is the most computationally expensive initialisation but ensures z^0 is feasible for all the programs $(\mathcal{M})$, $(\mathcal{P}_{\pi^0})$ and $(\mathcal{R}_{\tau^0})$. The three strategies can be summarised in Table SM1.

When a feasible initialisation is made (i.e. `LL-Feasible`), at each index i, the complementarity choice (for example, choosing $G_i(z^0) = 0$ or $H_i(z^0) = 0$) is pre-decided. We observe that, for many indices, this is unlikely to flip throughout the iterations of the solution methods. However, when an infeasible initialisation (i.e. `LL-Centre`, `LL-Zero`) is chosen that violates the complementarity constraint (like having $G_i(z^0) = 1$ and $H_i(z^0) = 1$, for example), we observe that the same algorithms are more free to choose between $G_i(z^0) = 0$ and $H_i(z^0) = 0$ in the convergence process and arrive at better solutions. This phenomenon is demonstrated in the following experiment: For each dataset and solution method, we run 75 starting points, with 25 according to each of the strategies `LL-Zero`, `LL-Feasible` and `LL-Centre`. We then assess the quality of the solution by the distribution of objective values found. These results for dataset *Circles* are summarised as box plots in Figure SM6, where the effect of the lower-level initialisation is particularly pronounced for the relaxation method. The algorithms are sensitive to the initial choice of (C, γ). This is due to the highly non-convex nature of the hyperparameter space. The choice of lower-level variable initialisation only has a very small effect on most datasets. However, overall, it is clear that `LL-Centre` encourages the algorithms to converge to higher quality solutions and have a low computational cost. Therefore, we will use the `LL-Centre` initialisation strategy for the remainder of our experiments.

An interesting visualisation of the iteration of the sequential relaxation method solving problem $(\mathcal{M})$, parametrised by the Ionosphere dataset, is given in Figure SM7. This shows the values of the complementarity variables α_i and $\underline{v}_i$ for the first fold and first four indices $i = 1, 2, 3, 4$. Note that the variables are initially infeasible for the MPCC but feasible for the subproblem $(\mathcal{R}_\tau)$. As we proceed down the rows, the variables z converge to either the axis corresponding to $\alpha_i = 0$ or the axis corresponding $\underline{v}_i = 0$, and therefore generating an MPCC feasible final solution.

SM5.2 Details of the classification datasets

We now present the data used for our experiments. We use eighteen datasets altogether. Six of these (namely, *Circles*, *Hastie*, *Linear*, *Moons*, *Swiss roll* and *XOR*) are synthetically generated with code

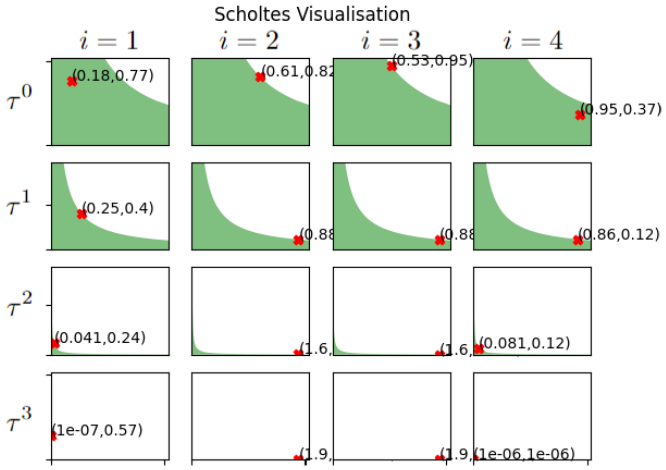

Fig. SM7 The columns correspond to fold $k = 1$ and indices $i = 1, 2, 3, 4$ (Left to right). The rows correspond to sequentially smaller relaxation parameters τ^t (top to bottom). The feasible regions for the relaxed program ($\mathcal{R}_{\tau^t}$) are coloured in green. Each of the 16 small plots has a red point for the value of the complementarity variables α_i (horizontal axis) and $\underline{v}_i$ (vertical axis) that were found to solve ($\mathcal{R}_{\tau^t}$).

from scikit-learn [85]. This is useful as globally and locally, optimal classifiers can be known from inspection, but could still be hard for a SVM to find. This allows us to test the solution methods objectively. The remaining twelve datasets come from real-world observations. In these cases, the optimal values are unknown, and we must compare our solutions to those achieved by other authors in the literature. We perform standard preprocessing on these datasets before passing them to the solvers. All rows with missing values are dropped. For continuous features, we subtract the mean and divide by the standard deviation. The resulting scaled data has a mean of zero and a standard deviation of one. This helps avoid scaling issues that can cause our problem to become ill-conditioned.

Regarding categorical features, we either drop the column entirely or apply a one-hot encoding where $0 - 1$ indicator variables are added when they are relevant to the classification. Finally, we partition the multi-class labels into binary classes, which we replace with the labels -1 and +1. A better classifier can sometimes be achieved through principal component analysis or other feature manipulation, however, we stick to these very simple preprocessing steps to allow a direct comparison to the original papers where the data was published.

Below, we list the datasets we use with a short description of any adjustments we make and a citation to the original publication. Each dataset is annotated with a tuple (n, d) where n is the number of observations and d is the dimension of the feature vectors.

1. **Banknote** $(1372, 4)$ The four features of this data set are digitally extracted from images of banknotes. Each banknote is then labelled as either genuine or forged. It is mostly balanced with 762 genuine and 610 forged examples. The banknote identification dataset was collected by Volker Lohweg [86].

2. **BUPA Liver Disorders** $(345, 6)$ Five features come from various blood tests relevant to liver disorders. A sixth feature comes from the patient's response to the question, "How many half-pint equivalents of alcoholic beverages do you drink per day?". If the patient is found to have any form of liver disorder, the patient is given a positive label. The data was published by Richard S. Forsyth and BUPA Medical Research Ltd. [87]

3. **Circles** (n, d) A synthetic dataset where half the examples are scattered uniformly over the surface of the d-dimensional unit ball and given a positive label. The other half are scattered on the surface of a smaller ball within the interior of the unit ball and given a negative label. Gaussian noise with

mean zero and standard deviation σ^2 is added to each point. For our experiments in Section 5, we parametrise with $n = 54, d = 2, \sigma^2 = 0.2$. See figure SM4, for an illustration.

4. **Cleveland** $(297, 14)$ The Cleveland Heart Disease data set was introduced by Detrano et al. [88]. Its features represent the physical attributes of a patient, such as age, sex, blood pressure, cholesterol, and heart rate, as recorded by their doctors. The labels represent a positive or negative diagnosis of heart disease. Originally, the dataset had 303 examples, 76 attributes and five classes. We adjust the dataset in the following manner. First, we select only 14 features that are commonly used by other authors. Then, we remove examples with missing values. Finally, we replace the different classes of heart disease with a positive label and the absence of heart disease with a negative label so that we can train a binary classifier.

5. **Glass** $(138, 9)$ Each sample is described in terms of its oxide content (sodium, magnesium, aluminium, etc.) and refractive index. The samples are labelled by their glass type, which can be useful to determine for criminal investigations. This data was collected by B. German [89] and is openly available on the UCI Machine Learning Repository. Since it is a multi-class dataset, we make 'building windows float processed' (70 examples) into the positive class and combine the vehicle, containers, tableware and lamp glass types (68 examples) into the negative class.

6. **Hastie** $(100, 10)$ This is the simulated data set from equation 10.2 of "The Elements of Statistical Learning" [53]. It has ten features that are independently and identically sampled according to a unit-independent Gaussian distribution. The labels are determined as $y_i = 1$ if $\|x_i\|^2 > 9.34$ otherwise $y_i = -1$.

7. **Ionosphere** $(351, 34)$ This data was collected from radar antennas in the ionosphere in Goose Bay, Labrador [90] and used to classify structure in the ionosphere. It is somewhat unbalanced, with 225 positive labels and 126 negative labels.

8. **Iris** $(100, 4)$ The famous Iris flower data set was introduced by Anderson and Fisher [91, 92]. Its four features represent the lengths and widths of the sepals and petals. The labels represent the species as either Setosa, Virginica or Versicolor. We discard all examples of the Versicolor class, so we end up with balanced binary labels of 50 Setosa and 50 Virginica.

9. **Linear** (n, d) A simple synthetic dataset where the data points are scattered randomly and uniformly over the space $[0, 1]^d$. A random hyperplane is chosen. The labels are then assigned as $+1$ if they lie above the hyperplane and -1 if they lie below. This is the only linearly separable dataset. In our experiments we choose $n = 100, d = 10$.

10. **MNIST** $(600, 784)$ Low resolution, 28x28 pixel, images of handwritten digits 0, 1, 2, ..., 9. The dataset is massive with 60,000 examples. We subsample 300 examples of the digit 0 and 300 examples of the digit 1 in order to convert to a binary classification problem.[93]

11. **Mushrooms** $(300, 117)$ This dataset details observations of gilled mushrooms in the Agaricus and Lepiota families [94]. All of the features are categorical, for example, the feature 'cap shape' could take the value bell, conical, convex, flat, knobbed or sunken. We add indicator variables for every category, which makes 117 features. Each example is labelled as either poisonous or edible.

12. **Moons** $(54, 2)$ A synthetic two-dimensional data set where points of the different classes are scattered in interweaving half-circles around each other, which can only be separated by a very nonlinear 'S'-shaped boundary.

13. **Palmer** $(342, 5)$ Penguins had their features, such as the culmen length, culmen depth, flipper length, body mass and sex, recorded and were labelled by their species. We perform a few modifications: drop the rows with missing values; replace the sex with a binary 1,0 indicator; and replace

the species with a positive label if Gentoo and a negative label otherwise. The data was collected by Palmer Station Antarctica LTER and K. Gorman [95] and is openly available on GitHub.

14. **Pima** $(768, 8)$ The features of the Pima Indians Diabetes Database are measurements such as BMI, insulin level, and age from female patients of Pima Indian heritage. The labels represent the diagnosis of diabetes. It is quite unbalanced with 500 negative examples but only 268 positive. The dataset was provided by the National Institute of Diabetes and Digestive and Kidney Diseases and introduced to the literature by Smith et al. [96].

15. **Sonar** $(208, 60)$ This data was collected to improve mine detection in the ocean. The many features are those collected from the sonar signal. The labels are positive if the sonar signals bounced off a metal cylinder and negative if they bounced off rocks. Terry Sejnowski made the data available, and it was first published by Gorman and Sejnowski [97].

16. **Swiss Roll** (n, d) Swiss Roll is a common synthetic dataset that can be tricky for an SVM as the two classes are rolled inside one another. It is presented in the book Machine Learning: An Algorithmic Perspective [63, Chapter 6, figure 6.13]. We use the Scikit-learn function make_swiss_roll to generate the data with Gaussian noise with standard deviation 0.1. In our experiments we choose $n = 90, d = 3$.

17. **Wisconsin** $(569, 30)$ This first breast cancer dataset was presented in the work of Street et al. [98]. Its thirty features are detailed measurements of the breast and tumour, such as radius, concavity, and compactness. Each example is then labelled as malignant or benign.

18. **XOR** $(n, 2)$ A two-dimensional synthetic data set where points are scattered uniformly at random across the plane $[-1, 1]^2$. They are then labelled as positive if the product of their coordinates is positive and negative otherwise. This leads to the top right and bottom left quadrants being the opposite class to the top left and bottom right quadrants. It can be a particularly problematic dataset for classifiers. In our experiments we choose $n = 100$.

These datasets are selected because they have quite different structures. The optimal (C, γ) hyperparameter choices are, for Ionosphere $(1.00e+01, 3.16e-02)$, Palmer $(1.00e+06, 1.00e-05)$, Moons $(1.78e+01, 5.62e+00)$ and Wisconsin $(3.16e+03, 1.78e-04)$. Furthermore, they show various different spreads of objective values and runtimes.

SM5.3 Details of the derivative-free methods

Grid search [52, p. 79][63] considers a sequence of plausible candidate choices for each hyperparameter and then forms a grid of every combination. At each coordinate of the grid (representing a choice for each hyperparameter) the algorithm trains, evaluates and then discards K fresh SVMs according to K-fold cross-validation. This is repeated independently for every combination in the Cartesian product that constitutes the grid. Finally, the hyperparameter combination that results in the best mean validation accuracy across the K classifiers is returned. In our experiments we select a grid of size 10×10 where ten choices of C are picked log-uniformly between 10^{-4} and 10^{6} and ten choices of γ are picked log-uniformly between 10^{-5} and 10^{4}.

Random search [52, p. 81][99] proceeds similarly to grid search; however, rather than a systematic traversal of a grid, at each iteration the hyperparameters are drawn randomly and independently from predefined distributions. This has the advantage of being able to search in-between the grid points of a grid search. In our experiments we select 100 random points from a log-uniform distribution with the same upper and lower bounds as the grid search.

Bayesian optimisation [100–103] is a more sophisticated probabilistic method. It maintains a model approximation of the function $\tilde{f}$ that maps hyperparameter selections to their corresponding mean validation accuracy and an acquisition function representing the expected improvement in $\tilde{f}$.

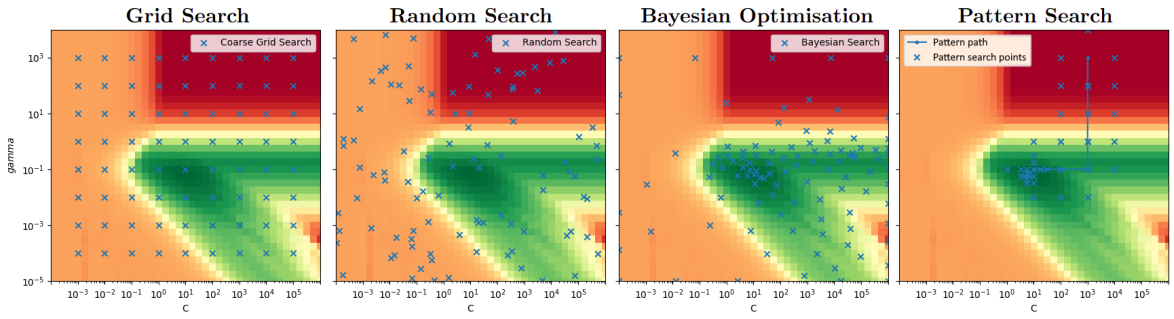

Fig. SM8 The heat map shows the hyperparameter space for the dataset Ionosphere. Green regions correspond to (C, γ) combinations, which allow the SVM to achieve high validation accuracy. The heat map is repeated four times for the grid, random, Bayesian and pattern search methods. On each, blue crosses are plotted on all the (C, γ) combinations evaluated by that search method.

At each iteration, the next choice of hyperparameters is determined by maximising the acquisition function - then the observed validation accuracy is used to update the approximation of $\tilde{f}$. The meta-hyperparameters of the surrogate model, including those controlling the balance between exploration and exploitation and the smoothness assumptions of $\tilde{f}$, must themselves be carefully chosen. Bayesian optimisation has much more computationally expensive iterations, but is often able to find points near the global optimum. We use the scikit-optimize implementation [104] with the following setup

```
skopt.gp_minimize(func,
  dimensions=[Real(1e-4, 1e+6, "log-uniform"), Real(1e-5, 1e+4, "log-uniform")],
  acq_func="gp_hedge", n_calls=100, random_state=42, xi=0.01, kappa=1.96,
)
```

Pattern search [105, Algorithm 9.2] differs from the previous three DFO methods in that it is a local rather than a global search. The algorithm maintains a current best hyperparameter selection (C, γ) and step size h. At each iteration, it evaluates new hyperparameters $\{(C \pm h, \gamma)\} \cup \{(C, \gamma \pm h)\}$ one step size away along each of the coordinate axes. If it finds a better objective value, it updates the current best selection and continues. Otherwise, it reduces its step size. Once the step size is sufficiently small, it terminates—at a local minimum if the function landscape is well-behaved. This method can be made more sophisticated with pattern moves, momentum or a line search. Although it is less commonly used in machine learning, it provides an interesting comparison to the relaxation and penalisation algorithms, which are also local methods. In our experiments steps are made in log-space. The step size is initialised to $h = 1$ and halved whenever a smaller objective cannot be found, down to 10^{-6}. All four methods are visualised in Figure SM8.

Our MPCC optimisation method has two clear advantages over DFO. The first advantage comes from the idea that two choices of hyperparameters with similar values often result in similar SVM states. For example, an optimal SVM with $(C, \gamma) = (15, 0.3)$ is likely to have many of the same support vectors as the optimal SVM with $(C, \gamma) = (12, 0.2)$. This means only a few decision variables α_i^k for $(k, i) \in I$ must be changed to move between optima for the adjusted hyperparameters. When one of grid, random, Bayesian or pattern search moves between two nearby hyperparameter selections in this manner, it recalculates the optimal decision variables from scratch, whereas the MPCC optimisation method only adjusts the required variables. The second advantage comes from the fact that the MPCC optimisation approach makes use of the first and second-order derivatives of the objective function with respect to the hyperparameters to move in a descent direction through the hyperparameter space.

		Penalisation (Exact)	Penalisation (Sequential)	Relaxation (Exact)	Relaxation (Sequential)	Semismooth Newton	Grid	Random	Bayesian	Pattern
Ionosphere	Runtime Q_1	**4**	9	16	55	386	64	69	400	83
	Runtime Q_2	**11**	12	29	90	589	98	106	880	132
	Runtime Q_3	**17**	**17**	40	149	1369	135	136	2152	133
	Objective Q_1	1.05	0.84	1.01	1.01	1.32	0.40	0.40	0.35	1.01
	Objective Q_2	0.62	0.47	0.42	0.40	0.87	0.37	0.38	**0.33**	0.47
	Objective Q_3	0.54	**0.33**	0.40	0.34	0.75	0.35	0.36	**0.33**	**0.33**
	Iterations	188	208	546	1569	891	49	50	50	8
	Validation acc.	84%	**89%**	86%	88%	79%	88%	87%	**89%**	**89%**
	Test acc.	83%	**89%**	86%	86%	79%	88%	87%	**89%**	**89%**
Palmer	Runtime Q_1	**1**	106	111	200	579	71	73	389	105
	Runtime Q_2	**1**	133	168	616	798	108	111	968	160
	Runtime Q_3	**27**	201	569	808	1639	143	143	2200	206
	Objective Q_1	0.89	0.69	0.91	0.42	1.26	0.20	0.04	0.04	0.90
	Objective Q_2	0.80	0.36	0.19	0.19	0.63	0.09	0.04	**0.03**	0.90
	Objective Q_3	0.44	**0.03**	0.19	0.19	0.46	0.04	0.04	**0.03**	0.04
	Iterations	54	235	290	849	516	49	50	50	8
	Validation acc.	90%	**99%**	95%	95%	90%	**99%**	**99%**	**99%**	**99%**
	Test acc.	90%	**99%**	95%	95%	90%	**99%**	**99%**	**99%**	**99%**
Moons	Runtime Q_1	**1**	2	2	6	3	49	49	447	72
	Runtime Q_2	**1**	2	2	8	4	82	104	678	103
	Runtime Q_3	**1**	3	3	16	112	104	111	979	217
	Objective Q_1	0.95	0.92	0.49	0.49	0.76	0.35	0.35	0.32	0.81
	Objective Q_2	0.60	0.38	0.48	0.32	0.54	0.34	0.34	**0.31**	0.47
	Objective Q_3	0.56	**0.31**	0.34	**0.31**	0.48	0.32	0.33	**0.31**	**0.31**
	Iterations	65	335	348	1125	665	49	50	50	8
	Validation acc.	83%	**89%**	88%	**89%**	74 %	**89%**	**89%**	**89%**	**89%**
	Test acc.	83%	87%	87%	**89%**	74 %	**89%**	**89%**	87%	87%
Wisconsin	Runtime Q_1	18	59	147	198	317	68	72	172	77
	Runtime Q_2	27	70	247	467	547	82	85	210	97
	Runtime Q_3	47	83	369	560	1431	94	95	591	104
	Objective Q_1	0.76	0.09	0.40	0.17	0.96	0.10	0.09	**0.08**	1.00
	Objective Q_2	0.52	**0.08**	0.18	0.13	0.51	0.09	**0.08**	**0.08**	0.12
	Objective Q_3	0.27	**0.08**	0.18	0.12	0.39	**0.08**	**0.08**	**0.08**	**0.08**
	Iterations	85	230	852	1565	979	49	50	50	8
	Validation acc.	93%	**97%**	94%	96%	90%	**97%**	**97%**	**97%**	96%
	Test acc.	93%	**97%**	94%	96%	90%	**97%**	**97%**	**97%**	96%

Table SM3 A summary of the solution methods (columns) performance on four of the eighteen datasets (rows). The first quartile (Q_1) is the runtime/objective value achieved by 75% of initialisations; the second quartile (Q_2) is the median; and the third quartile (Q_3) is the runtime/objective value achieved by the top 25% of initialisations.

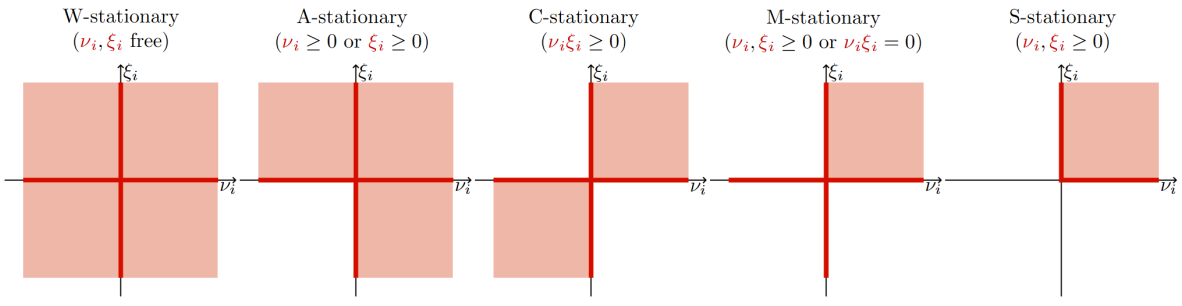

Fig. SM9 This diagram highlights the differences in the W, A, C, M and S stationary conditions. Areas coloured in red show possible values the multipliers ν_i and ξ_i may take for a stationary point $\bar{z}$ where $G_i(\bar{z}) = H_i(\bar{z}) = 0$

References

[1] Arrow, K.J., Debreu, G.: Existence of an equilibrium for a competitive economy. Econometrica **22**(3), 265–290 (1954). Accessed 2024-07-29

[2] Baumrucker, B., Renfro, J.G., Biegler, L.T.: MPEC problem formulations and solution strategies with chemical engineering applications. Computers & Chemical Engineering **32**(12), 2903–2913 (2008)

[3] Gabriel, S.A., Leuthold, F.U.: Solving discretely-constrained mpec problems with applications in electric power markets. Energy Economics **32**(1), 3–14 (2010)

[4] Lawphongpanich, S., Hearn, D.W.: An MPEC approach to second-best toll pricing. Mathematical Programming **101**(1), 33–55 (2004)

[5] Luo, Z.-Q., Pang, J.-S., Ralph, D.: Mathematical Programs with Equilibrium Constraints. Cambridge University Press, ??? (1996). https://www.cambridge.org/core/books/mathematical-programs-with-equilibrium-constraints/03981C32ABDD55A4001BF58BA0C57444

[6] Flegel, M.L.: Constraint qualifications and stationarity concepts for mathematical programs with equilibrium constraints. PhD thesis, Universität Würzburg (2005). http://d-nb.info/975013661/34

[7] Pang, J.-S., Fukushima, M.: Complementarity constraint qualifications and simplified B-stationarity conditions for mathematical programs with equilibrium constraints. Computational Optimization and Applications **13**, 111–136 (1999)

[8] Scheel, H., Scholtes, S.: Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. Mathematics of Operations Research **25**(1), 1–22 (2000)

[9] Schwartz, A.: Mathematical programs with complementarity constraints: Theory, methods and applications. PhD thesis, Universität Würzburg (2011). https://tu-dresden.de/mn/math/numerik/schwartz/ressourcen/dateien/Schwartz_Dissertation.pdf

[10] Ye, J.J.: Necessary and sufficient optimality conditions for mathematical programs with equilibrium constraints. Journal of Mathematical Analysis and Applications **307**(1), 350–369 (2005)

[11] Scholtes, S.: Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. SIAM Journal on Optimization **11**(4), 918–936 (2001)

[12] Dempe, S., Zemkoho, A.B.: On the Karush–Kuhn–Tucker reformulation of the bilevel optimization problem. Nonlinear Analysis: Theory, Methods & Applications **75**(3), 1202–1218 (2012). Variational Analysis and Its Applications

[13] Gfrerer, H., Ye, J.J.: New constraint qualifications for mathematical programs with equilibrium constraints via variational analysis. SIAM Journal on Optimization **27**(2), 842–865 (2017)

[14] Kim, Y., Leyffer, S., Munson, T.: MPEC methods for bilevel optimization problems. In: Dempe, S., Zemkoho, A. (eds.) Bilevel Optimization: Advances and Next Challenges, pp. 335–360. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52119-6_12

[15] Qiu, S., Chen, Z.: Solving Mathematical Programs with Equilibrium Constraints as Nonlinear Programming: A New Framework (2015). https://arxiv.org/abs/1510.07145

[16] Ralph, D., Wright, S.J.: Some properties of regularization and penalization schemes for MPECs. Optimization Methods and Software **19**(5), 527–556 (2004)

[17] Anitescu, M.: On using the elastic mode in nonlinear programming approaches to mathematical programs with complementarity constraints. SIAM Journal on Optimization **15**(4), 1203–1236 (2005). Published version of Preprint ANL/MCS-P864-1200, Argonne National Laboratory, Argonne, IL, 2000

[18] Leyffer, S., López-Calva, G., Nocedal, J.: Interior methods for mathematical programs with complementarity constraints. SIAM Journal on Optimization **17**(1), 52–77 (2006)

[19] DeMiguel, V., Friedlander, M.P., Nogales, F.J., Scholtes, S.: A two-sided relaxation scheme for mathematical programs with equilibrium constraints. SIAM Journal on Optimization **16**(2), 587–609 (2005)

[20] Hoheisel, T., Kanzow, C., Schwartz, A.: Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints. Mathematical Programming **137**, 257–288 (2013)

[21] Lin, G.-H., Fukushima, M.: A modified relaxation scheme for mathematical programs with complementarity constraints. Annals of Operations Research **133**(1), 63–84 (2005)

[22] Steffensen, S., Ulbrich, M.: A new relaxation scheme for mathematical programs with equilibrium constraints. SIAM Journal on Optimization **20**(5), 2504–2539 (2010)

[23] Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**, 273–297 (1995)

[24] Vapnik, V.: The Nature of Statistical Learning Theory. Springer, ??? (1999). https://doi.org/10.1007/978-1-4757-3264-1

[25] Tefas, A., Kotropoulos, C., Pitas, I.: Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication. IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(7), 735–746 (2001)

[26] Rahman, M.S., Rahman, M.K., Kaykobad, M., Rahman, M.S.: isGPT: An optimized model to identify sub-golgi protein types using SVM and random forest based feature selection. Artificial intelligence in medicine **84**, 90–100 (2018)

[27] Smith, N., Gales, M.: Speech recognition using SVMs. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14. MIT Press, ??? (2001). https://proceedings.neurips.cc/paper_files/paper/2001/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf

[28] Wolpert, D.H.: The lack of a priori distinctions between learning algorithms. Neural Computation **8**(7), 1341–1390 (1996)

[29] Wainer, J., Fonseca, P.: How to tune the RBF SVM hyperparameters? an empirical evaluation of 18 search algorithms. Artificial Intelligence Review **54**(6), 4771–4797 (2021)

[30] Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M.: Bilevel programming for hyperparameter optimization and meta-learning. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1568–1577. PMLR, ??? (2018). https://proceedings.mlr.press/v80/franceschi18a.html

[31] Li, Z., Qian, Y., Li, Q.: A unified framework and a case study for hyperparameter selection in

machine learning via bilevel optimization. In: 2022 5th International Conference on Data Science and Information Technology (DSIT), pp. 1–8 (2022). IEEE. http://doi.org/10.1109/DSIT55514.2022.9943929

[32] Okuno, T., Takeda, A., Kawana, A., Watanabe, M.: On lp-hyperparameter learning via bilevel nonsmooth optimization. Journal of Machine Learning Research **22**(245), 1–47 (2021)

[33] Moore, G., Bergeron, C., Bennett, K.P.: Model selection for primal SVM. Machine Learning **85**, 175–208 (2011)

[34] Moore, G.M.: Bilevel programming algorithms for machine learning model selection. PhD thesis, Rensselaer Polytechnic Institute (2010). https://www.proquest.com/dissertations-theses/bilevel-programming-algorithms-machine-learning/docview/860000550/se-2

[35] Moore, G.M., Bergeron, C., Bennett, K.P.: Nonsmooth bilevel programming for hyperparameter selection. In: 2009 IEEE International Conference on Data Mining Workshops, pp. 374–381 (2009). IEEE. https://doi.org/10.1109/ICDMW.2009.74

[36] Gao, L.L., Ye, J., Yin, H., Zeng, S., Zhang, J.: Value function based difference-of-convex algorithm for bilevel hyperparameter selection problems. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 7164–7182. PMLR, ??? (2022). https://proceedings.mlr.press/v162/gao22j.html

[37] Li, Q., Li, Z., Zemkoho, A.: Bilevel hyperparameter optimization for support vector classification: theoretical analysis and a solution method. Mathematical Methods of Operations Research **96**(3), 315–350 (2022)

[38] Jiang, W., Siddiqui, S.: Hyper-parameter optimization for support vector machines using stochastic gradient descent and dual coordinate descent. EURO Journal on Computational Optimization **8**(1), 85–101 (2020)

[39] Bennett, K.P., Hu, J., Ji, X., Kunapuli, G., Pang, J.-S.: Model selection via bilevel optimization. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 1922–1929 (2006). IEEE. https://doi.org/10.1109/IJCNN.2006.246935

[40] Kunapuli, G., Bennett, K., Hu, J., Pang, J.-S.: Bilevel model selection for support vector machines. In: CRM Proceedings and Lecture Notes, vol. 45, pp. 129–158 (2008). https://gkunapuli.github.io/files/08bilevelML.pdf

[41] Kunapuli, G., Bennett, K.P., Hu, J., Pang, J.-S.: Classification model selection via bilevel programming. Optimization Methods and Software **23**(4), 475–489 (2008)

[42] Coniglio, S., Dunn, A., Li, Q., Zemkoho, A.: Bilevel hyperparameter optimization for nonlinear support vector machines. Unpublished (2022)

[43] Leyffer, S.: MacMPEC. https://wiki.mcs.anl.gov/leyffer/index.php/MacMPEC

[44] Ward, S., Zemkoho, A., Ahipasaoglu, S.: Mathematical programs with complementarity constraints and application to hyperparameter tuning. arXiv preprint arXiv:2504.13006 (2025)

[45] Solodov, M.V.: Constraint qualifications. In: Wiley Encyclopedia of Operations Research and Management Science, p. 341. John Wiley & Sons, Ltd, ??? (2011). Chap. 4. https://doi.org/10.1002/9780470400531.eorms0978

[46] Chen, Y., Florian, M.: The nonlinear bilevel programming problem: formulations, regularity and optimality conditions. Optimization **32**(3), 193–209 (1995)

[47] Qi, L., Wei, Z.: On the constant positive linear dependence condition and its application to SQP methods. SIAM Journal on Optimization **10**(4), 963–981 (2000)

[48] Mangasarian, O.L.: Nonlinear programming. McGraw-Hill, New York (1969)

[49] Dutta, J., Deb, K., Tulshyan, R., Arora, R.: Approximate KKT points and a proximity measure for termination. Journal of Global Optimization **56**(4), 1463–1499 (2013)

[50] Murray, W.: Analytical expressions for the eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions. Journal of Optimization Theory and Applications **7**(3), 189–196 (1971)

[51] Wright, M.H.: Some properties of the Hessian of the logarithmic barrier function. Mathematical Programming **67**(1), 265–295 (1994)

[52] Géron, A.: Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc., ??? (2022). https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

[53] Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction vol. 2. Springer, ??? (2009). https://doi.org/10.1007/978-0-387-84858-7

[54] Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the royal statistical society: Series B (Methodological) **36**(2), 111–133 (1974)

[55] Fan, R.-E., Chen, P.-H., Lin, C.-J.: Working set selection using second order information for training support vector machines. Journal of Machine Learning Research **6**(63), 1889–1918 (2005)

[56] Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A mathematical programming language. Management Science **36**(5), 519–554 (1990)

[57] Van Rossum, G., Drake, F.L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009). https://dl.acm.org/doi/book/10.5555/1593511

[58] Harris, C.R., Millman, K.J., Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M.H., Brett, M., Haldane, A., Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020)

[59] Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming **106**, 25–57 (2006)

[60] Byrd, R.H., Nocedal, J., Waltz, R.A.: KNITRO: An integrated package for nonlinear optimization. In: Di Pillo, G., Roma, M. (eds.) Large-Scale Nonlinear Optimization. Nonconvex Optim. Appl., vol. 83, pp. 35–59. Springer, Boston, MA (2006). https://doi.org/10.1007/0-387-30065-1_4

[61] Fletcher, R., Leyffer, S.: Numerical experience with solving MPECs as NLPs. Numerical Analysis Report NA/210, Department of Mathematics, University of Dundee, Dundee, UK **8** (2002)

[62] Leyffer, S.: Complementarity constraints as nonlinear equations: Theory and numerical experience. In: Dempe, S., Kalashnikov, V. (eds.) Optimization with Multivalued Mappings: Theory, Applications, and Algorithms, pp. 169–208. Springer, Boston, MA (2006). https://doi.org/10.1007/0-387-34221-4_9

[63] Marsland, S.: Machine Learning: an Algorithmic Perspective. Chapman and Hall/CRC, ??? (2011). https://github.com/hongzhonglu/machine-learning-books/blob/master/Machine%20Learning%20-%20An%20Algorithmic%20Perspective%202nd%20edition%202014.pdf

[64] Bengio, Y.: Gradient-based optimization of hyperparameters. Neural Computation **12**(8), 1889–1900 (2000)

[65] Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. Machine Learning **46**, 131–159 (2002)

[66] Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007)

[67] Waskom, M.L.: seaborn: statistical data visualization. Journal of Open Source Software **6**(60), 3021 (2021)

[68] Stackelberg, H.v.: Marktform und gleichgewicht. The Economic Journal **45**(178), 334–336 (1934)

[69] Dempe, S., Zemkoho, A.: Bilevel optimization. In: Springer Optimization and Its Applications vol. 161. Springer, ??? (2020). https://link.springer.com/book/10.1007/978-3-030-52119-6

[70] Dempe, S., Dutta, J.: Is bilevel programming a special case of a mathematical program with complementarity constraints? Mathematical Programming **131**, 37–48 (2012)

[71] Wendland, H.: Scattered Data Approximation vol. 17. Cambridge university press, ??? (2004). https://doi.org/10.1017/CBO9780511617539

[72] George, R.K., Ajayakumar, A.: A Course in Linear Algebra. Springer, ??? (2024). https://doi.org/10.1007/978-981-99-8680-4

[73] Zhang, F.: The Schur Complement and Its Applications vol. 4. Springer, ??? (2006). https://doi.org/10.1007/b105056

[74] Harder, F., Mehlitz, P., Wachsmuth, G.: Reformulation of the m-stationarity conditions as a system of discontinuous equations and its solution by a semismooth Newton method. SIAM Journal on Optimization **31**(2), 1459–1488 (2021)

[75] Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., Lopez, A.: A comprehensive survey on support vector machine classification: Applications, challenges and trends. Neurocomputing **408**, 189–215 (2020)

[76] Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152 (1992). https://doi.org/10.1145/130385.130401

[77] Guyon, I., Boser, B., Vapnik, V.: Automatic capacity tuning of very large VC-dimension classifiers. In: Hanson, S., Cowan, J., Giles, C. (eds.) Advances in Neural Information Processing Systems, vol. 5. Morgan-Kaufmann, ??? (1992). https://proceedings.neurips.cc/paper_files/paper/1992/file/eaae339c4d89fc102edd9dbdb6a28915-Paper.pdf

[78] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004). https://doi.org/10.1017/CBO9780511804441

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

[80] Nicole, D., Takeda, K., Wolton, I., Cox, S.: Southampton high performance computing centre. In: High-Performance Computing, pp. 33–41. Springer, Boston, MA (1999). https://doi.org/10.1007/978-1-4615-4873-7_4

[81] Dirkse, S.P., Ferris, M.C.: MCPLIB: A collection of nonlinear mixed complementarity problems. Optimization Methods and Software **5**(4), 319–345 (1995)

[82] Dirkse, S., Ferris, M.C., Munson, T.: The PATH Solver. https://pages.cs.wisc.edu/%7Eferris/path.html

[83] Dirkse, S.P., Ferris, M.C.: The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. Optimization Methods and Software **5**(2), 123–156 (1995)

[84] Dirkse, S.P., Ferris, M.C.: A path search damped Newton method for computing general equilibria. Annals of Operations Research **68**(2), 211–232 (1996)

[85] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay: Scikit-learn: Machine learning in python. Journal of Machine Learning Research **12**(85), 2825–2830 (2011)

[86] Lohweg, V.: Banknote Authentication. UCI Machine Learning Repository (2013). https://doi.org/10.24432/C55P57

[87] Forsyth, R.S.: BUPA Liver Disorders. BUPA Medical Research Ltd. (1990). http://archive.ics.uci.edu/ml/machine-learning-databases/liver-disorders/bupa.data

[88] Detrano, R.C., Jánosi, A., Steinbrunn, W., Pfisterer, M.E., Schmid, J.-J., Sandhu, S., Guppy, K., Lee, S., Froelicher, V.: International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology **64**(5), 304–310 (1989)

[89] German, B.: Glass Identification. UCI Machine Learning Repository (1987). https://doi.org/10.24432/C5WW2P

[90] Sigillito, V., Wing, S., Hutton, L., Baker, K.: Ionosphere. UCI Machine Learning Repository (1989). https://doi.org/10.24432/C5W01B

[91] Anderson, E.: The species problem in Iris. Annals of the Missouri Botanical Garden **23**, 457–509 (1936)

[92] Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Eugenics **7**(2), 179–188 (1936)

[93] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[94] Lincoff, G.H., Knopf, A.A.: Mushroom. The Audubon Society Field Guide to North American Mushrooms (1981). https://doi.org/10.24432/C5959T

[95] Horst, A.M., Hill, A.P., Gorman, K.B.: Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data. (2020). https://doi.org/10.5281/zenodo.3960218

[96] Smith, J.W., Everhart, J.E., Dickson, W., Knowler, W.C., Johannes, R.S.: Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the Annual Symposium on Computer Application in Medical Care, p. 261 (1988). American Medical Informatics Association. https://pmc.ncbi.nlm.nih.gov/articles/PMC2245318/

[97] Gorman, R.P., Sejnowski, T.J.: Analysis of hidden units in a layered network trained to classify sonar targets. Neural Networks **1**(1), 75–89 (1988)

[98] Street, W.N., Wolberg, W.H., Mangasarian, O.L.: Nuclear feature extraction for breast tumor diagnosis. In: Acharya, R.S., Goldgof, D.B. (eds.) Biomedical Image Processing and Biomedical Visualization, vol. 1905, pp. 861–870. SPIE, ??? (1993). https://doi.org/10.1117/12.148698 . International Society for Optics and Photonics. https://doi.org/10.1117/12.148698

[99] Mantovani, R.G., Rossi, A.L., Vanschoren, J., Bischl, B., De Carvalho, A.C.: Effectiveness of random search in SVM hyper-parameter tuning. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2015). Ieee. https://doi.org/10.1109/IJCNN.2015.7280664

[100] Garnett, R.: Bayesian Optimization. Cambridge University Press, ??? (2023). https://bayesoptbook.com/

[101] Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast Bayesian optimization of machine learning hyperparameters on large datasets. In: Singh, A., Zhu, J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 528–536. PMLR, ??? (2017). https://proceedings.mlr.press/v54/klein17a.html

[102] Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1. GECCO'99, pp. 525–532. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999). https://dl.acm.org/doi/10.5555/2933923.2933973

[103] Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Pereira, F., Burges, C.J., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 25. Curran Associates, Inc., ??? (2012). https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[104] Head, T., Kumar, M., Nahrstaedt, H., Louppe, G., Shcherbatyi, I.: scikit-optimize. Zenodo (2021). https://doi.org/10.5281/zenodo.5565057

[105] Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, ??? (1999). https://doi.org/10.1007/978-0-387-40065-5