Computers and Chemical Engineering 139 (2020) 106902

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

HY-POP: Hyperparameter optimization of machine learning models through parametric programming

William W. Tso[a,b], Baris Burnak[a,b], Efstratios N. Pistikopoulos[a,b,*]

[a] Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, TX 77843, USA
[b] Texas A&M Energy Institute, Texas A&M University, College Station, TX 77843, USA

ARTICLE INFO

Article history:
Received 21 January 2020
Revised 7 April 2020
Accepted 29 April 2020
Available online 18 May 2020

Keywords:
Machine learning
Hyperparameter optimization
Parametric programming
Bilevel optimization
Model selection
Regularization

ABSTRACT

Fitting a machine learning model often requires presetting parameter values (hyperparameters) that control how an algorithm learns from the data. Selecting an optimal model that minimizes error and generalizes well to unseen data becomes a problem of tuning or optimizing these hyperparameters. Typical hyperparameter optimization strategies involve discretizing the parameter space and implementing an iterative search procedure to approximate the optimal hyperparameter and model selection through cross-validation. Instead, for machine learning algorithms that are formulated as linear or quadratic programming (LP/QP) models, an exact solution to the hyperparameter optimization problem is obtainable through parametric programming without any approximation. First, the hyperparameter optimization problem is posed more naturally as a bilevel optimization. Second, using parametric programming theory, the bilevel optimization is reformulated into a single level problem. Exact solutions to the hyperparameter optimization problem for LASSO regression and LP L_1-norm support vector machine (SVM) are derived and validated on example data.

© 2020 Elsevier Ltd. All rights reserved.

1. Introduction

Let X be a $i \times j$ data matrix, and Y be a $i \times 1$ response vector, where i is the sample size and j is the number of predictors or features. In supervised learning (regression or classification) problems, it is assumed that there exists a function f that maps the relationship between a set of input predictors $X = (X_1, X_2, \ldots, X_j)$, where X_j is the jth column vector of X, and output responses Y.

$$Y = f(X) + \epsilon \tag{1}$$

f represents the learnable information that X provides about Y, while ϵ is a random error term containing information that is unmeasured or unavailable in the data for the learning process. Because f is not exactly known, machine learning algorithms are needed to estimate f and predict Y.

$$\hat{Y} = \hat{f}(X) \tag{2}$$

$\hat{f}$ represents the estimate for f, and $\hat{Y}$ is the resulting prediction. ϵ is not included in the prediction because it averages out to be zero.

The accuracy of this estimation and prediction is the squared error between Y and $\hat{Y}$. Assuming $\hat{f}$ is fixed, the expected error of a single predicted point $\hat{y}_0$ from one observation x_0, a given row vector of X, is decomposed into reducible and irreducible quantities (James et al., 2013).

$$\begin{aligned}\mathbf{E}[y_0 - \hat{y}_0]^2 &= \mathbf{E}[y_0 - \hat{f}(x_0)]^2 \\ &= [f(x_0) + \epsilon - \hat{f}(x_0)]^2 \\ &= \underbrace{[f(x_0) - \hat{f}(x_0)]^2}_{\text{reducible}} + \underbrace{Var(\epsilon)}_{\text{irreducible}}\end{aligned} \tag{3}$$

Because $\hat{f}$ is not a perfect estimate for f, this inaccuracy introduces some error. This error is reducible because it is possible to improve the fit of $\hat{f}$ by using a better performing algorithm. Even if $\hat{f}$ were to exactly match f, the prediction of Y still has some error associated with it due to ϵ. This error is irreducible because a model cannot account for information that is not contained in the data while the algorithm is learning. This provides an upper bound (Abu-Mostafa et al., 2012) on the accuracy of any estimated $\hat{f}$.

As such, the goal of any machine learning method is to minimize the reducible error in order to maximize the accuracy of $\hat{f}$ to be closer to its upper bound. The reducible error is made up of

* Corresponding author at: Texas A&M Energy Institute, 1617 Research Pkwy, College Station, TX 77843, USA.
E-mail address: stratos@tamu.edu (E.N. Pistikopoulos).

https://doi.org/10.1016/j.compchemeng.2020.106902
0098-1354/© 2020 Elsevier Ltd. All rights reserved.

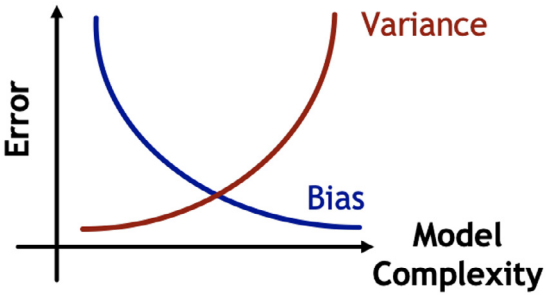

Fig. 1. More complex models generally have lower bias and higher variance, while less flexible models generally have higher bias and lower variance.

two components: bias & variance (James et al., 2013).

$$\underbrace{[f(x_0) - \hat{f}(x_0)]^2}_{\text{reducible}} = \underbrace{Var[\hat{f}(x_0)]}_{\text{variance}} + \underbrace{Bias[\hat{f}(x_0)]^2}_{\text{bias}} \tag{4}$$

Bias is the error introduced by approximating a real-world phenomena, which is often complex, with a simpler model that has less fidelity. Variance measures the sensitivity of $\hat{f}$ to the training data set and how much its fit would change if estimated using different data. In general, as a model becomes more complicated or flexible (including more parameters to estimate), bias decreases and variance increases (James et al., 2013; Abu-Mostafa et al., 2012). A flexible $\hat{f}$ fits closer to given training data set (lowering bias), but is more sensitive to training data variability (raising variance). The reverse of this bias & variance trade-off for $\hat{f}$ is also generally true for a simpler model. Fig. 1 illustrates the bias & variance trade-off with respect to model complexity.

Therefore, selecting an optimal $\hat{f}$ from a set of various candidate ones, ranging in complexity from simple linear to highly nonlinear, involves balancing the bias & variance trade-off. The ideal machine learning method is one that simultaneously achieves low bias and low variance (Wilson and Sahinidis, 2017). Possessing these characteristics gives the learned model a higher probability of generalizing well to unseen data during model training and predict more accurately (Abu-Mostafa et al., 2012). Too much variance leads to $\hat{f}$ overfitting the data, and too much bias leads to $\hat{f}$ underfitting the data (Fig. 2).

The most utilized approach for finding an optimal $\hat{f}$ is to incorporate an additional regularization term in a machine learning algorithm's loss function formulation (Hastie et al., 2009). Typically, the basic loss function for a supervised learning problem is the minimization of the mean squared error (MSE) between Y and $\hat{Y}$, where N is the sample size.

$$\min \quad \frac{1}{N}\|Y - \hat{Y}\|_2^2 = \frac{1}{N}\|Y - \hat{f}(X)\|_2^2 \tag{5}$$

A regularization term contains an exogenous penalty parameter (hyperparameter) whose value is set prior to training the model. This hyperparameter λ controls the importance and weight of the regularization term, which affects the resulting optimization solution of a machine learning algorithm. A common regularization term (Hastie et al., 2009) is λ penalizing the q-norm of w, the model weights of $\hat{f}$ (an example is $\hat{f}(X) = Xw$), raised to the power p.

$$\min_{w} \quad \frac{1}{N}\|Y - \hat{f}(X)\|_2^2 + \lambda\|w\|_q^p \tag{6}$$

In general, machine learning algorithms may have multiple hyperparameters that are prespecified (Bengio, 2000; Foo et al., 2008). Moreover, hyperparameters do not only exist inside a regularization term. Any external parameter that is not inferred by the machine learning model and affects the performance, speed, or quality of the learning process is considered a hyperparameter. This work focuses on a hyperparameter regularizing the model fit, but similar extensions are possible to other hyperparameters, such as those located inside kernel functions.

In Eq. (6), the aim of regularization is to control the complexity of $\hat{f}$ that is fitted. As the value of λ varies from 0 to ∞, the resulting estimated $\hat{f}$ will have different reducible error realizations. Likewise, the bias and variance values for each $\hat{f}$ are different. By including a regularization term, the reducible errors for several $\hat{f}$ candidates are comparable and are an implicit function of the hyperparameter. Finding an optimal $\hat{f}$ with low bias and low variance amounts to correctly tuning λ. However, what is the best value for λ is not known *a priori*. Therefore, the selection of an optimal machine learning model is really a hyperparameter optimization problem (Fig. 3).

The rest of the paper is organized as follows. In Section 2, different strategies for addressing the hyperparameter optimization problem are summarized. In Section 3, some background on parametric programming theory and its implications for the hyperparameter optimization problem are briefly described. In Section 4, the conversion of K-fold cross-validation for tuning hyperparameters into a bilevel optimization problem is discussed. The usage of parametric programming to reformulate the bilevel optimization into a single level problem is also presented. In Sections 5 and 6, this bilevel & parametric optimization approach for hyperparameter tuning (HY-POP) is applied on LASSO regression and a L_1-norm support vector machine (SVM), respectively. In the former, it is validated that HY-POP leads to the same optimal λ and $\hat{f}$ as previous results that developed a closed-form solution to LASSO regression. The latter extends HY-POP to achieve a new understanding of LP L_1-norm SVM. These examples highlight the ability of HY-POP to explicitly solve the hyperparameter optimization for machine learning algorithms that are represented as linear programming or quadratic programming (LP/QP) models. It is noted that these models are chosen to demonstrate HY-POP, and an analysis of model sensitivity is not the aim of this work. Finally, some concluding remarks are made and future directions are suggested.

2. Hyperparameter optimization

Commonly used strategies for hyperparameter optimization (Bergstra et al., 2011; Claesen and De Moor, 2015; Hutter et al., 2015; Luo, 2016) involve dividing the parameter space into D evenly or randomly discretized points and performing an iterative optimization procedure through K-fold cross-validation (Fig. 4). First, the data is split into K subsets. Within each subset, the data is further separated into training and testing sets. Next, for each discretized λ value, a separate optimization problem for the machine learning model (Eq. (6)) is constructed on the training data in each fold and solved to estimate $\hat{f}$. The validation error $\frac{1}{N}\|Y - \hat{Y}\|_2^2$ is then computed using the estimated $\hat{f}$ and testing data in the same fold to predict Y. Finally, after iterating through all the λ values, the validation errors for each λ across all folds are averaged together. The optimal $\hat{f}$ is the one with the λ value that gives the smallest mean validation error.

K-fold cross-validation with grid (Liu et al., 2006) or random search (Bergstra and Bengio, 2012) for hyperparameter optimization is a generalizable way to approximate optimal model selection. The advantages of K-fold cross-validation are that, in most cases, it captures the actual test error as an implicit function of λ well-enough and calculates a $\hat{f}$ that balances bias & variance (James et al., 2013; Abu-Mostafa et al., 2012). The disadvantages of K-fold cross-validation are that sometimes $K \times D$ optimization problems may become a computational burden to solve and possibly lead to inexact solutions that are troublesome due to poor discretization. One way to bypass solving $K \times D$ optimization prob-

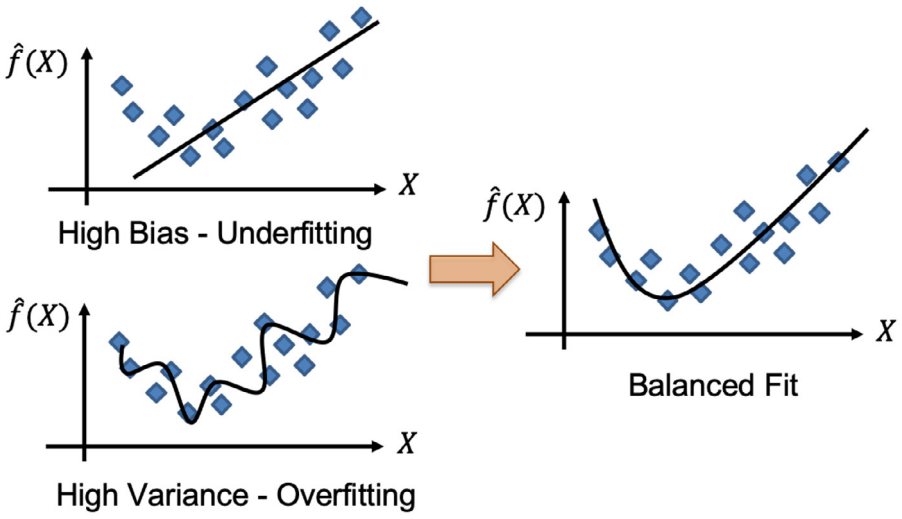

Fig. 2. Finding a good $\hat{f}$ fit requires balancing bias and variance.

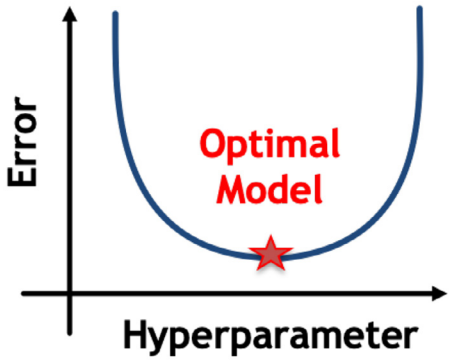

Fig. 3. Optimal model selection amounts to determining the optimal hyperparameter.

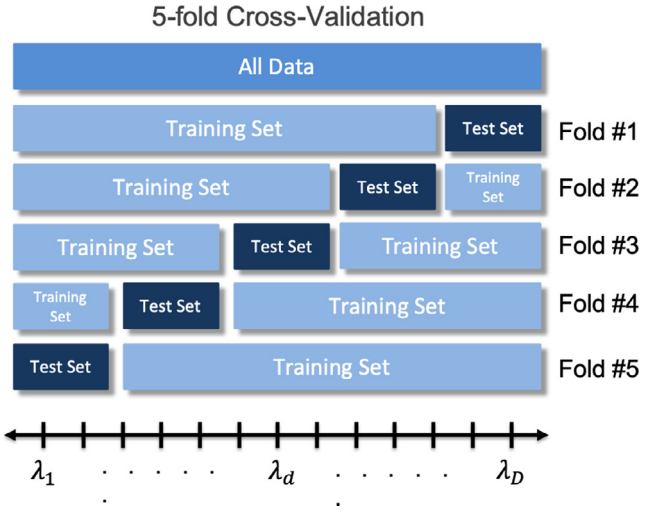

Fig. 4. An overview of *K*-fold cross-validation for hyperparameter optimization.

lems in *K*-fold cross-validation is through a bilevel optimization approach (Bennett et al., 2006; Klatzer and Pock, 2015), where the training and testing objectives are modeled together. This is discussed further in Section 4.

Another common method to tune hyperparameters is using Bayesian optimization (Snoek et al., 2012; Eggensperger et al., 2013). This approach still involves iteratively exploring the hyperparameter space, where a probabilistic model of the validation error as a function of λ is built from prior evaluations. The probabilistic model is then used to approximate the optimal hyperparameter by assigning probability values to its location and selecting the one with the highest probability. Gradient-based methods for hyperparameter optimization (Bengio, 2000) pose the validation error as a nonlinear objective function, with λ as the decision variable, and uses gradient descent to find a locally optimal λ. As such, there is no guarantee that the hyperparameter optimization is solved to a global optimum. Bayesian and gradient-based methods have been applied to many machine learning algorithms including LASSO regression (Gao et al., 2010; Barratt and Sharma, 2018) and SVM (Gold et al., 2005; Keerthi et al., 2007). Most recently, black-box or derivative-free optimization (Rios and Sahinidis, 2013), which treats the validation error as an unknown function to be interpolated from point evaluations in numerical experiments and optimizes the error through iterative evaluations in the hyperparameter space, has been applied to hyperparameter tuning for neural networks (Diaz et al., 2017; Koch et al., 2018).

3. A parametric programming perspective

The developments in this work first translate the *K*-fold cross-validation for hyperparameter optimization into a bilevel optimization problem and then solve it through parametric programming. This methodology gives the exact solution and does not require any approximation, probabilistic modeling or iterative searching of the hyperparameter space. The parametric programming approach applies for hyperparameter optimization problems that have a machine learning algorithm that is explicitly formulated as a LP/QP model. Moreover, for these problems, global optimality of λ is guaranteed from parametric programming theory.

Parametric programming is an optimization strategy, popularized by explicit model predictive control (MPC) (Bemporad et al., 2002), that determines the optimal solution as a function of a varying parameter θ, without exhaustively traversing the entire parameter space. The general form is seen in Eq. (7). The objective (loss) function F, inequality constraints g, and equality constraints h are all functions of the decision variables u and parameter θ. The optimal solution comprises a set of finite areas (critical regions in Fig. 5), where a particular solution is valid for a given realization of θ, along with explicit expressions relating the decision variables

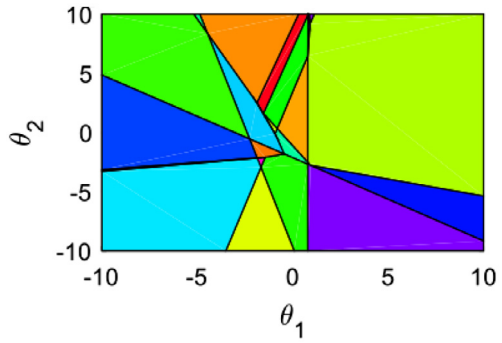

Fig. 5. An example of critical regions comprising the optimal solution to a parametric programming problem (Eq. (7)).

u to the θ parameter ($u = A\theta + b$). Using this, the objective (loss) function is also solely expressed as a function $J(\theta)$.

$$\begin{aligned} J(\theta) = \min_{u} \; & F(u,\theta) \\ \text{s.t.} \; & g(u,\theta) \le 0 \\ & h(u,\theta) = 0 \end{aligned} \tag{7}$$

In the classic explicit MPC problem (Bemporad et al., 2002), decision variables u are the control inputs and parameters θ are the system states. For the hyperparameter optimization setup, decision variables u are the model weights w to $\hat{f}$ and parameters θ are the hyperparameters λ. In this work, $\hat{f}$ is assumed to be linear with respect to w.

$$\hat{f}(X, w) = \phi(X)w \tag{8}$$

w is a $j \times 1$ vector of model weights and ϕ is a vector of basis functions transforming each predictor of X such that $\phi(X) = [\phi_1(X_1), \phi_2(X_2), \ldots, \phi_j(X_j)]$. $\hat{f}$ is a function of w because the model weights are learned from training the machine learning model.

By viewing Eq. (6) as an optimization problem in the form of Eq. (7), a new understanding of the hyperparameter optimization problem is reached. From this parametric programming perspective, a different affine expression $w = A_r\lambda + b_r$ governs each critical region r of λ and the learned model is redefined as $\hat{f}(X, \lambda) = \phi(X)A_r\lambda + \phi(X)b_r$. Having $\hat{f}$ as an explicit function of λ has important ramifications for the bilevel optimization approach to K-fold cross-validation discussed later in Section 4.

The algorithmic procedure (graph, geometrical, combinatorial) for computing the critical regions and affine expressions for Eq. (7) depends on the optimization problem structure (linear, nonlinear, convex, differentiable) and the nature of the variables & parameters (continuous or binary). All algorithms employ an active set strategy, where each identified critical region has a unique combination of active constraints that represent the optimal solution. In general, parametric programming is also extendable to the case of multiple varying parameters (multi-parametric programming). The reader is referred to several review papers and books for further discussion on multi-parametric programming theory (Pistikopoulos, 2009; 2012; Oberdieck et al., 2016a; Pistikopoulos et al., 2011; 2020) and its applications (Diangelakis et al., 2017; 2018; Burnak et al., 2019; Ogumerem and Pistikopoulos, 2019; Onel et al., 2019a; Tian et al., 2020).

4. Bilevel optimization of *K*-fold cross-validation

Within each kth fold of cross-validation for hyperparameter optimization, there are two different objectives. On the training set level, the goal is to minimize the training error in Eq. (6), where Y is the output response from the training data. On the testing set level, the validation error $\frac{1}{N}\|Y - \hat{Y}\|_2^2$ is evaluated using the learned $\hat{f}$ from model training to predict the output response Y from the testing data. After model training and recording the validation errors across all λ values for every fold, the goal is to select the optimal $\hat{f}$ that minimizes the mean validation error across all folds. Overall, K-fold cross-validation seeks λ such that when the optimal training is solved for each training set, the validation error over the test errors is minimized.

In this setup, the dual objectives are captured using a bilevel optimization (Colson et al., 2007; Sinha et al., 2017) formulation (Eq. (9)). Note this is an example formulation, and modifications may be necessary depending on the particular machine learning algorithm, as seen later in Section 6. However, the general concepts described in this section remain valid. In the inner level, the objective is to minimize each kth fold's training error with a regularization penalty, the decision variables are the model weights w_k, and the parameter is λ. In the outer level, the objective is to minimize the mean squared validation error across $|K|$ folds, the decision variable is λ, and the parameters are w_k.

$$\begin{aligned} \min_{\lambda} \quad & \frac{1}{|K|}\sum_{k=1}^{|K|}\frac{1}{N_k^{tst}}\|y_k^{tst} - \hat{y}_k\|_2^2 \\ \text{s.t.} \quad & \min_{w_k} \; \frac{1}{N_k^{trn}}\|y_k^{trn} - \hat{f}_k\|_2^2 + \lambda\|\hat{w}_k\|_q^p \qquad \forall k \in K \end{aligned} \tag{9}$$

K is the set of all data folds. For each kth fold, N_k^{trn} is the training set size, N_k^{tst} is testing set size, y_k^{trn} is a $N_k^{trn} \times 1$ vector of output responses in the training set, y_k^{tst} is a $N_k^{tst} \times 1$ vector of output responses in the testing set, $\hat{f}_k$ is the trained machine learning model of form $\hat{f}_k(X, w_k) = \phi(X)w_k$, and $\hat{y}_k$ is a $N_k^{tst} \times 1$ vector of predicted responses from $\hat{f}_k$ using the testing set. Again, $\phi(X)$ is vector of basis functions transforming the columns of X and w_k is a $j \times 1$ vector. Note that $\hat{f}_k$ is estimated using X from the training set, $\hat{f}_k(X_k^{trn}, w_k) = \phi(X_k^{trn})w_k$, and then X from the testing set is inputted with w_k fixed to predict the output response, $\hat{y}_k = \hat{f}_k(X_k^{tst}, w_k) = \phi(X_k^{tst})w_k$.

This observation of K-fold cross-validation as a bilevel optimization problem has also been noted by earlier works (Bennett et al., 2006; Klatzer and Pock, 2015; Pedregosa, 2016; Franceschi et al., 2018; MacKay et al., 2019). Some of these authors (Bennett et al., 2006; Klatzer and Pock, 2015) attempted to solve the bilevel optimization by replacing the inner level optimization problem with its Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions are Lagrangian and complementarity constraints that represent the optimality of the inner level problem, reducing the bilevel optimization into a single level constrained optimization problem after reformulation. However, even if both levels are LPs, nonlinear terms containing Lagrange multipliers and decision variables arise within the complementarity constraints from reformulating the bilevel optimization using the KKT approach. This renders the single level optimization to be a mixed-integer nonlinear programming (MINLP) problem, which is a very difficult to solve to global optimality. Sometimes, even achieving a feasible solution is also challenging. This becomes more difficult when the optimization problems on the two levels are nonlinear programs (NLP) since the Lagrangian constraints may also now be nonlinear.

If the original machine learning algorithm with regularization penalty in the inner level of Eq. (9) is well-posed as a LP ($p = 1$ & $q = 1$) or QP ($p = 2$ & $q = 2$), parametric programming is another viable strategy for reformulating the bilevel optimization into a single level optimization that is a mixed-integer quadratic (MIQP) problem. Although both methods will give the same optimal λ, the advantage of parametric programming is that it preserves useful information about the optimal solution profile that the KKT ap-

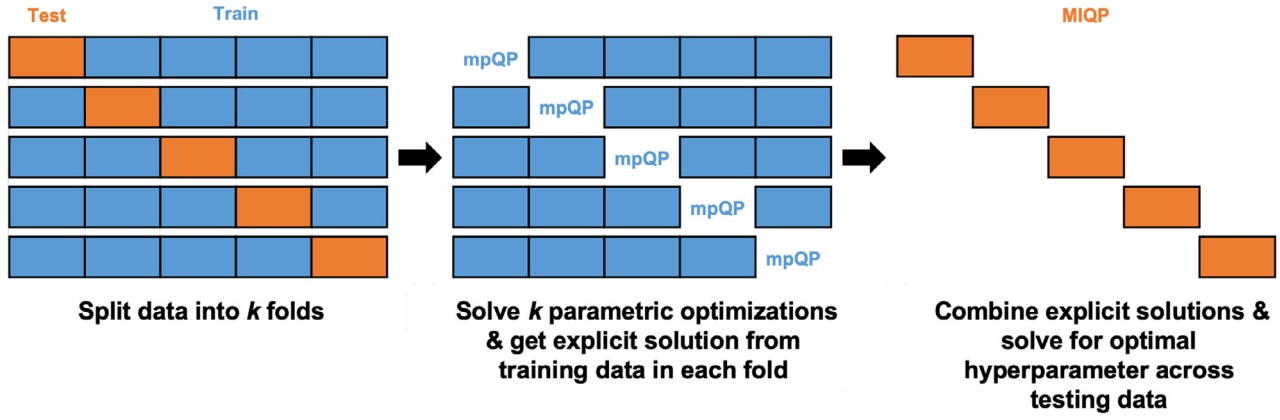

Fig. 6. HY-POP solves for the explicit solutions for the training optimization through parametric programming and passes along this information to solve for the optimal hyperparameter in a single optimization.

proach does not. The KKT approach only provides a single optimal solution, and a MINLP is more difficult to solve than a MIQP. Through parametric programming, the model weights are derived as explicit affine functions of the hyperparameter ($w_k = A_{kr}\lambda + b_{kr}$) for each kth fold and critical region r of λ. As such, $\hat{f}_k$ is expressible as a function of λ, and the machine learning model becomes $\hat{f}_k(X,\lambda) = \phi(X)A_{kr}\lambda + \phi(X)b_{kr}$. Note that exactly one critical region is active (one corresponding pair of A_{kr} and b_{kr} coefficients are nonzero) for each kth fold. This is because λ is the single decision variable in the outer optimization in Eq. (9).

This new form of $\hat{f}$ is important because the training error (inner level objective in Eq. (9)) and the mean validation error (outer level objective in Eq. (9)) are both now explicit functions of just λ. This is easily seen through substituting the affine function of $w_k(\lambda)$ into the inner and outer objectives. For the training error, $\hat{f}_k(X_k^{trn}, w_k) = \hat{f}_k(X_k^{trn}, \lambda)$, and for the validation error, $\hat{y}_k = \hat{f}_k(X_k^{tst}, w_k) = \hat{f}_k(X_k^{tst}, \lambda)$. The affine function of $w_k(\lambda)$, capturing the optimal solution in the inner optimization, passes information between the two levels of Eq. (9) to reduce the bilevel optimization into a single level.

By utilizing parametric programming, the implicit function of error versus λ discussed in Sections 1 and 2 is no longer unknown and now has a closed-form expression. This is a huge advantage for parametric programming over using KKT conditions for hyperparameter optimization because, in addition to the optimal λ, the complete training & validation error versus λ profiles are given. Having these profiles makes it easier to understand the trained machine learning model and visualize the prediction results. Effective data visualization helps to build more interpretable and sparse models, which are important for improved surrogate modeling of highly complex systems and increased understanding of which predictors most influence the outcome.

The model formulation of the HY-POP approach for the hyperparameter optimization example in Eq. (9) is formally described below. Bilevel optimization through parametric programming (B-POP) (Avraamidou and Pistikopoulos, 2019a) for other applications have also been demonstrated in previous works (Faísca et al., 2007; Domínguez and Pistikopoulos, 2010; Oberdieck et al., 2017; Avraamidou and Pistikopoulos, 2017; 2019b). Fig. 6 shows an overview of the HY-POP strategy for hyperparameter optimization. Multi-parametric quadratic programming models (mpQP) is the general form of machine learning models that are allowed for the inner level optimization problem.

The first step is to replace the inner level optimization in Eq. (9) with constraints that define the optimal solution profile from the critical regions. Since there are $|K|$ folds, there are $|K|$ training optimization problems to replace with parametric programming. The critical regions for each kth training optimization are separately calculated, and then they are all combined together into the appropriate constraint set. These constraints control the affine expressions of $w_k(\lambda)$ and restrict only one critical region to be active for each fold to represent the optimal training. This is conveyed through introducing Big-M constraints (Eq. (10)), critical region bound constraints (Eq. (11)), and a SOS1 constraint (Eq. (12)).

$$w_k \le A_{kr}\lambda + b_{kr} + M(1 - y_{kr}^{CR}) \qquad \forall k \in K, \forall r \in R^k$$
$$w_k \ge A_{kr}\lambda + b_{kr} + M(y_{kr}^{CR} - 1) \qquad \forall k \in K, \forall r \in R^k \tag{10}$$

M is an appropriately large-enough constant value. For each kth fold, R^k is the set of all critical regions that comprise the optimal training solution, w_k is a $j \times 1$ vector of model weights to the trained machine learning model, A_{kr} and b_{kr} are $j \times 1$ coefficient vectors in the affine expression for w_k from a critical region r, and y_{kr}^{CR} are binary variables fixing/relaxing w_k for active/inactive critical regions. The Big-M constraints determine which critical region r in each fold k defines w_k for the machine learning model $\hat{f}_k$.

$$\sum_{r=1}^{|R^k|} LB_{kr}^{CR} y_{kr}^{CR} \le \lambda \le \sum_{r=1}^{|R^k|} UB_{kr}^{CR} y_{kr}^{CR} \qquad \forall k \in K \tag{11}$$

$$\sum_{r=1}^{|R^k|} y_{kr}^{CR} = 1 \qquad \forall k \in K \tag{12}$$

To ensure that only one critical region is active for each fold, Eq. (12) enforces this discrete decision. The lower (LB_{kr}^{CR}) and upper (UB_{kr}^{CR}) bounds to λ in a critical region r from fold k define the range of values for λ such that a particular affine expression for w_k and an resulting optimal solution apply. The upper bound of a critical region is the lower upper bound of the next critical region in the same fold. Overall, the bounds from each fold span the same hyperparameter range. To enforce that the same λ value is utilized across all folds, when selecting an active critical region in each fold to represent the optimal training, Eq. (11) defines that the lower and upper bounds of λ from these $|K|$ critical regions must overlap each other. Together, Eqs. (10)–(12) replace the inner optimization in Eq. (9).

The second step is to substitute $\hat{y}_k = \phi(X_k^{tst})w_k$ into the objective (loss) function of the outer level in Eq. (9). Since the inner level decision variable w_k is a function of the outer level decision variable λ from parametric programming, the bilevel optimization is converted into a single level optimization. Eq. (13) constitutes the HY-POP reformulation of the hyperparameter optimization example in Eq. (9), assuming the machine learning algorithm is a LP

($p = 1$ & $q = 1$) or QP ($p = 2$ & $q = 2$) and $\hat{f}_k$ is linear with respect to its model weights w_k. Because the outer validation error objective is in MSE form and binary variables y_{kr}^{CR} are added for the critical regions, this single level optimization is also a MIQP.

$$\begin{aligned}
\min_{\lambda, w_k, y_{kr}^{CR}} \quad & \frac{1}{|K|}\sum_{k=1}^{|K|}\frac{1}{N_k^{tst}}\|y_k^{tst} - \phi(X_k^{tst})w_k\|_2^2 \\
\text{s.t.} \quad & w_k \le A_{kr}\lambda + b_{kr} + M(1 - y_{kr}^{CR}) \qquad \forall k \in K, \forall r \in R^k \\
& w_k \ge A_{kr}\lambda + b_{kr} + M(y_{kr}^{CR} - 1) \qquad \forall k \in K, \forall r \in R^k \\
& \sum_{r=1}^{|R^k|} LB_{kr}^{CR} y_{kr}^{CR} \le \lambda \le \sum_{r=1}^{|R^k|} UB_{kr}^{CR} y_{kr}^{CR} \qquad \forall k \in K \\
& \sum_{r=1}^{|R^k|} y_{kr}^{CR} = 1 \qquad \forall k \in K
\end{aligned} \tag{13}$$

Depending on the actual machine learning algorithm, the example formulations of Eqs. (9) and (13) may need some modifications. One instance of this is the LP L_1-norm SVM in Section 6. However, the general concept shown here of using parametric programming to connect the two levels of a bilevel optimization problem (assuming LP or QP) through affine expressions relating model weights to the hyperparameter remains valid. In Section 6, similar steps, as performed for Eqs. (9) and (13), are taken to formulate the K-fold cross-validation for LP L_1-norm SVM hyperparameter optimization through a HY-POP approach. Nevertheless, the example formulations of Eqs. (9) and (13) are useful for LASSO regression in the next section.

5. LASSO regression

LASSO (Tibshirani, 1996) is a popular regression technique that performs model selection through regularization. It introduces a L_1-norm penalty on β, a vector of regression coefficients (model weights), to the ordinary least squares (OLS) model. This attempts to improve the regression fit by reducing the variance observed in the OLS estimation for β and better balance the bias & trade-off. LASSO regression is one useful method to build sparse surrogate models for data-driven optimization (Beykal et al., 2018b; 2018a). The LASSO regression form is shown in Eq. (14).

$$\min_{\beta} \quad \tfrac{1}{2N}\|Y - \phi(X)\beta\|_2^2 + \lambda\|\hat{\beta}\|_1 \tag{14}$$

Eq. (14) is a parametric programming problem in the form of Eq. (7) and fits the problem structure shown in the inner level of Eq. (9). Because the L_1-norm regularization term is nonlinear, $\|\beta\|_1 = \sum_j |\beta_j|$, Eq. (14) is first reformulated before it is solved through parametric programming. It is observed that β_j is piecewise linear. After substituting $\alpha_j = |\beta_j|$ and adding two constraints to describe the piecewise behavior, the parametric quadratic programming (pQP) model for LASSO regression is the following.

$$\begin{aligned}
\min_{\beta, \alpha} \quad & \frac{1}{2N}\|Y - \phi(X)\beta\|_2^2 + \lambda\sum_j \alpha_j \\
\text{s.t.} \quad & \alpha_j \ge \beta_j \qquad \forall j \in J \\
& \alpha_j \ge -\beta_j \qquad \forall j \in J
\end{aligned} \tag{15}$$

To optimize λ in Eq. (15) and select an optimal LASSO model, the bilevel optimization depiction of K-fold cross-validation from Eq. (9) is implemented. Eq. (15) is the inner optimization problem, and the outer level objective still is to minimize the validation MSE across all folds. The critical regions representing the optimal solution profile to Eq. (15) for each kth fold are the similar to those in Eqs. (10)–(12), with the only difference being the parametric programming solution now accounts for the two added constraints for the reformulation of $|\beta_j|$. The final HY-POP formulation (MIQP) of the LASSO hyperparameter optimization is in Eq. (16). β_k is a $j \times 1$ vector of regression coefficients.

Table 1
Model statistics for training LASSO pQPs (Eq. (15)).

Fold #	X_k^{trn} $(i \times j)$	# Constraints × # Variables	# Critical regions
1	23 × 7	14 × 14	5
2	23 × 7	14 × 14	6
3	23 × 7	14 × 14	6
4	23 × 7	14 × 14	6
5	24 × 7	14 × 14	6

$$\begin{aligned}
\min_{\lambda, \beta_k, y_{kr}^{CR}} \quad & \frac{1}{|K|}\sum_{k=1}^{|K|}\frac{1}{N_k^{tst}}\|y_k^{tst} - \phi(X_k^{tst})\beta_k\|_2^2 \\
\text{s.t.} \quad & \beta_k \le A_{kr}\lambda + b_{kr} + M(1 - y_{kr}^{CR}) \qquad \forall k \in K, \forall r \in R^k \\
& \beta_k \ge A_{kr}\lambda + b_{kr} + M(y_{kr}^{CR} - 1) \qquad \forall k \in K, \forall r \in R^k \\
& \sum_{r=1}^{|R^k|} LB_{kr}^{CR} y_{kr}^{CR} \le \lambda \le \sum_{r=1}^{|R^k|} UB_{kr}^{CR} y_{kr}^{CR} \qquad \forall k \in K \\
& \sum_{r=1}^{|R^k|} y_{kr}^{CR} = 1 \qquad \forall k \in K
\end{aligned} \tag{16}$$

Eq. (16) is a new structure for the hyperparameter optimization of LASSO regression through K-fold cross-validation. Next, this HY-POP formulation is validated on an ammonia reactor data example and against a coordinate descent algorithm (with grid search) from the glmnet package in R.

5.1. Ammonia reactor data example

A dataset of 29 samples are collected from different sources on the performance of an industrial ammonia synthesis reactor (Tso et al., 2018; Demirhan et al., 2019). The exact data values are included in the supplementary material. These values are normalized and centered before training. Reactor temperature T & pressure P, inlet molar concentration of hydrogen x_{H_2}, nitrogen x_{N_2}, ammonia x_{NH_3} & inert species x_{Inert}, and the molar ratio between hydrogen & nitrogen $\frac{x_{H_2}}{x_{N_2}}$ are 7 predictors for the reactor conversion y_X. It is assumed that the predictors are linear, $\phi(X) = X$. Therefore, the proposed LASSO model that is trained has the following form in Eq. (17). An intercept term is not included because the data is centered.

$$y_X = \beta_1 T + \beta_2 P + \beta_3 x_{H_2} + \beta_4 x_{N_2} + \beta_5 x_{NH_3} + \beta_6 x_{Inert} + \beta_7 \frac{x_{H_2}}{x_{N_2}} \tag{17}$$

While this is a small dataset, the goal is not a comprehensive computation study, but to validate that the HY-POP approach correctly identifies the optimal λ & β, compared to an established coordinate descent algorithm. The data is randomly divided into 5 folds for cross-validation. The fold identification of the data points is also provided in the supplementary material. Each training optimization problem (Eq. (15)) is formulated in MATLAB 2019b. An in-house developed and state-of-the-art software, the Parametric Optimization (POP) toolbox (Oberdieck et al., 2016b), is then used to solve for the critical regions, using the built-in QP solver from MATLAB and the geometrical algorithm. A geometrical algorithm is selected over a graph or combinatorial approach since Eq. (15) only has one hyperparameter. Because of this, the former is more efficient at identifying the active set of constraints for each critical region than the latter two. The model statistics for each training optimization problem (Eq. (15)) is depicted in Table 1. An example result of the critical regions for fold #2 is shown in Fig. 7.

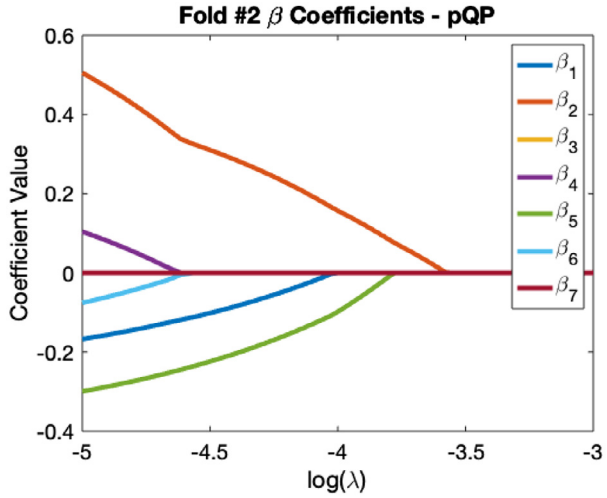

Fig. 7. LASSO regularization path for the ammonia reactor data in fold #2 from the training pQP (Eq. (15)) solved using POP (Oberdieck et al., 2016b).

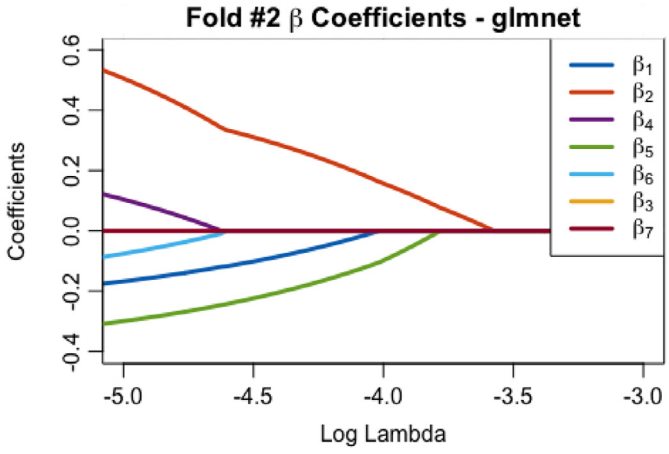

Fig. 8. LASSO regularization path for the ammonia reactor data in fold #2 solved using a coordinate descent algorithm from glmnet in R.

The piecewise linear relationship between β & λ is referred to as the LASSO regularization path (Hastie et al., 2009). This path helps to visualize the importance of each feature on the model prediction and interpret the model through selecting the features with more influence toward this prediction. Less important features typically have β values that approach 0 faster as λ gets larger. For fold #2 in Fig. 7, it appears that pressure is the most influential feature on the model's prediction since its β value is the last one to reach 0.

We expected this piecewise behavior from the affine expressions $\beta(\lambda)$ provided by the parametric programming solution to Eq. (15). Each line segment piece represents a critical region, where a unique $\beta(\lambda)$ function is valid for the values of λ. Having the β change in a piecewise linear fashion has also been previously observed in least angle regression (LAR) (Efron et al., 2004) and coordinate descent (Friedman et al., 2010), the first efficient algorithms developed to solve LASSO regression. While these methods were specifically tailored for LASSO regression (Efron et al., 2004; Friedman et al., 2010), parametric programming is the general theory of solving problems in the form presented in Eq. (7). Therefore, both LAR and coordinate descent algorithms can actually be viewed as specialized parametric programming approaches. In Fig. 8, it observed that the coordinate descent algorithm (with 10^3 evenly discretized points for $\lambda \in [10^{-3}, 1]$) gives exactly same regularization path as Fig. 7. This verifies that the critical regions for each fold exactly represent the optimal solution profile for the pQP (Eq. (15)).

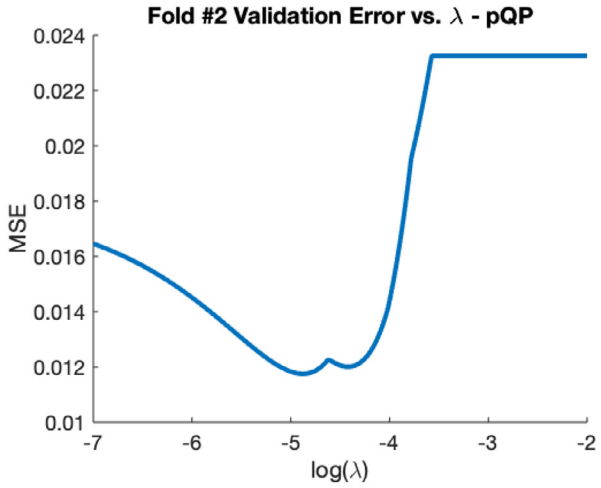

Fig. 9. Validation error for the ammonia reactor data in fold #2 from the training pQP (Eq. (15)) solved using POP (Oberdieck et al., 2016b).

Table 2
Optimal values and CPU times for 5-fold cross-validation on ammonia reactor data.

Method	Optimal MSE	Optimal λ	CPU time[a] (s)
pQP + MIQP	0.0479	0.0140	1.44 ± 0.06
`cv.glmnet`[b]	0.0478	0.0137	0.19 ± 0.01

[a] Averaged over 10 runs.
[b] Grid of 10^3 points.

With $\beta(\lambda)$ given from parametric programming, calculating the training and validation errors for each fold, the objectives in Eqs. (15) and (16), respectively, are simple function evaluations. Likewise, these errors are also piecewise functions with respect to λ, but they are not linear due to the squaring of the error term. Fig. 9 is an example of this nonlinear piecewise behavior for the testing error in fold #2. With error as a function of λ, finding the optimal λ is an easy calculation, pinpointing the minimum of these validation error profiles aggregated across all folds. Profiles of the LASSO regularization path and validation error for all folds are found in the supplementary material.

To find this minimum mean validation error, after computing the critical regions, the MIQP for hyperparameter optimization (Eq. (16)) is formulated in MATLAB 2019b and solved using IBM ILOG CPLEX Optimization Studio 12.9. Overall, the model has 421 constraints, 36 continuous variables, and 29 binary variables. The MIQP solution is compared to result given from using `cv.glmnet`, the cross-validation function included in glmnet, with a grid of 10^3 evenly discretized points for $\lambda \in [10^{-3}, 1]$. Fig. 10 highlights that the resulting mean validation error profiles calculated from these two methods. The shaded blue and gray areas represent one standard error above and below the mean validation error.

The validation error profiles appear exactly the same, confirming that the HY-POP approach leads to the same solution as the established coordinate descent algorithm in glmnet. Table 2 depicts some computational results. From minimizing the MIQP, the HY-POP approach calculates an optimal λ of 0.140, while `cv.glmnet` calculates an optimal λ of 0.1365. This slight difference in value is attributed to numerical sensitivity in the algorithmic computations and how the λ space is the discretized.

Not controlling for programming environment, the coordinate descent algorithm appears faster that the HY-POP approach. This is

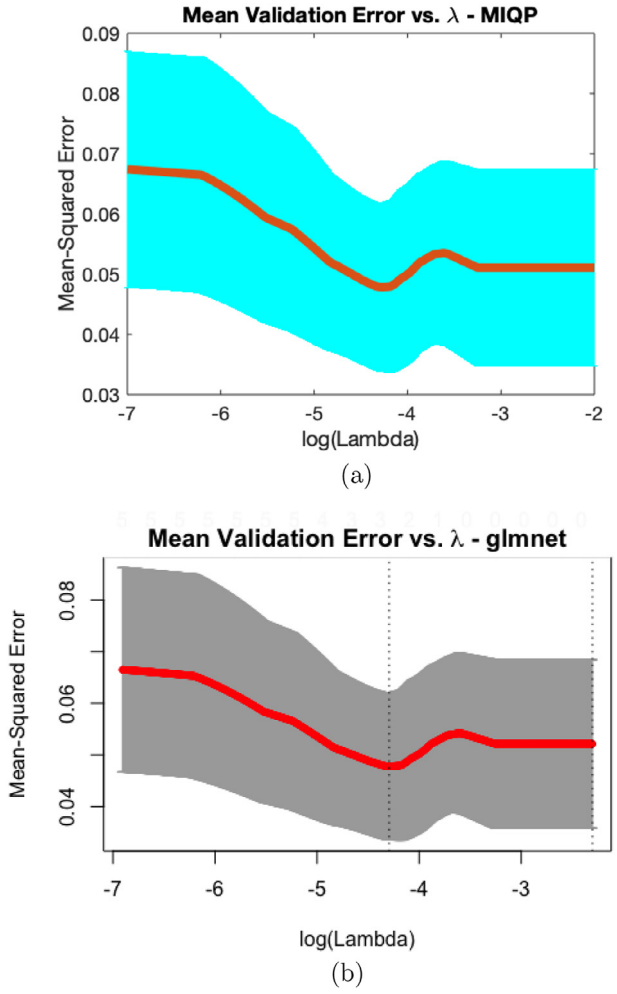

Fig. 10. Mean validation error for the hyperparameter optimization of LASSO regression on the ammonia reactor data through 5-fold cross-validation. (a) Error profile solved using the MIQP (Eq. (16)). (b) Error profile solved using `cv.glmnet`.

expected because the former is a tailored method with warm starts for solving the pQP of LASSO regression, while the latter uses a generic algorithm applicable to any pQP with no specializations for LASSO regression. Moreover, the MATLAB code has not been optimized for speed. One potential idea for speed-up is to parallelize the geometrical algorithm so that multiple starting points are used to initialize. Albeit for a small example, it is promising though that the generic POP solver is only a order of magnitude slower. For the HY-POP approach here, about 84% of the CPU time is spent determining the critical regions for the folds. Time spent constructing the model is included in the CPU time.

However, depending on the size of X_k^{trn}, the number of folds, and the granularity of how λ is discretized, these computational conclusions may change. These are subjects for further study. There may be corner cases where a much finer discretization of λ is needed for a good approximation, and an effective grid search would require more computations. The tractability of a parametric problem depends on the number of decision variables and constraints. In the pQP for LASSO (Eq. (15)), the problem size only grows with the number of features j in X_k^{trn}. β_j and α_j are decision variables that scale with j, and the number of constraints also scales with j. Therefore, it is likely that Eq. (15) becomes more difficult to solve as j grows, and the sample size i of X_k^{trn} has a less significant impact since it only participates in the objective function. Any active set based strategy to solve parametric problems relies on choosing n active constraints for n decision variables to square the system. Since the number of decision variables and constraints scale together equally, it is expected that X_k^{trn} with a reasonably larger number of features and sample sizes will remain tractable for the LASSO pQP. What is the upper bound on j for maintaining tractability requires further investigation.

6. LP L_1-norm support vector machine (SVM)

SVM (Boser et al., 1992) is a common classification technique that identifies a maximal margin hyperplane separating different classes of labeled responses and utilizes it as a decision boundary. It has been applied for fault detection and process monitoring in chemical engineering (Onel et al., 2018; 2019b). The standard general form is the C-parameterized SVM, where a hyperparameter C penalizes the slack variables ε_i controlling how much margin violation to tolerate for each misclassified observation i.

$$\begin{aligned} \min_{w,b,\varepsilon} \quad & \frac{1}{m}\|w\|_q^p + C\sum_i \varepsilon_i^d \\ \text{s.t.} \quad & y_i(\phi(x_i)w + b) \geq 1 - \varepsilon_i \qquad \forall i \in I \\ & \varepsilon_i \geq 0 \qquad \forall i \in I \end{aligned} \tag{18}$$

I is the set of all training observations, w is a $j \times 1$ vector of model weights, b is a constant bias, and m is a constant value. For the ith observation, x_i is a $1 \times j$ vector of predictors, y_i is the given class label, and ε_i is the slack variable. Binary classification is assumed, where the y_i is either $+1$ or -1.

The power d to which ε_i is raised in the objective (loss) function dictates whether Eq. (18) is the L_1-norm ($d = 1$) or the L_2-norm ($d = 2$) SVM. When $p = 1$ & $q = 1$, Eq. (18) is the LP SVM, and when $p = 2$ & $q = 2$, it is the QP SVM. Typically, $m = 1$ for the LP and $m = 2$ for the QP. Since LASSO regression in previous section is an example of the HY-POP approach applied to a QP, in this section, LP L_1-norm SVM (Bradley and Mangasarian, 1998) is used as an example of the HY-POP approach implemented on a LP. The LP L_1-norm SVM (Bradley and Mangasarian, 1998) is shown below.

$$\begin{aligned} \min_{w,b,\varepsilon} \quad & \sum_j |w_j| + C\sum_i \varepsilon_i \\ \text{s.t.} \quad & y_i(\phi(x_i)w + b) \geq 1 - \varepsilon_i \qquad \forall i \in I \\ & \varepsilon_i \geq 0 \qquad \forall i \in I \end{aligned} \tag{19}$$

Eq. (19) is a parametric programming problem in the form of Eq. (7), but does not have a squared error loss as LASSO regression (Eq. (14)) did. Instead, the objective is to minimize the sum of margin violations, while selecting which predictors are more important to construct the hyperplane. The C hyperparameter is associated with the slack variables ε accounting for this margin violation sum, instead of the model weights w for LASSO regression. An absolute value reformulation of $\sum_j = |w_j|$ is performed for Eq. (19), where $|w_j| = p_j + q_j$ and $w_j = p_j - q_j$. The parametric linear programming (pLP) for LP L_1-norm SVM is the following.

$$\begin{aligned} \min_{p,q,b,\varepsilon} \quad & \sum_j p_j + q_j + C\sum_i \varepsilon_i \\ \text{s.t.} \quad & y_i(\phi(x_i)p - \phi(x_i)q + b) \geq 1 - \varepsilon_i \qquad \forall i \in I \\ & \varepsilon_i \geq 0 \qquad \forall i \in I \\ & p_j \geq 0 \qquad \forall j \in J \\ & q_j \geq 0 \qquad \forall j \in J \end{aligned} \tag{20}$$

Eq. (20) is the inner training optimization to the bilevel problem to optimize C. For each kth fold, the critical regions describing the optimal solution profile to Eq. (20) are in the same form as

those in Eqs. (10)–(12), taking into account the reformulation of $|w_j|$. The only difference here is there are two sets of Big-M constraints (Eq. (10)) to account for w and b. The critical regions for Eq. (20) for all folds are defined using the following constraints, where w is evaluated from $p - q$.

$$
\begin{aligned}
& w_k \le A_{kr}C + b_{kr} + M(1 - y^{CR}_{kr}) && \forall k \in K, \forall r \in R^k \\
& w_k \ge A_{kr}C + b_{kr} + M(y^{CR}_{kr} - 1) && \forall k \in K, \forall r \in R^k \\
& b_k \le G_{kr}C + h_{kr} + M(1 - y^{CR}_{kr}) && \forall k \in K, \forall r \in R^k \\
& b_k \ge G_{kr}C + h_{kr} + M(y^{CR}_{kr} - 1) && \forall k \in K, \forall r \in R^k \\
& \sum_{r=1}^{|R^k|} LB^{CR}_{kr} y^{CR}_{kr} \le C \le \sum_{r=1}^{|R^k|} UB^{CR}_{kr} y^{CR}_{kr} && \forall k \in K \\
& \sum_{r=1}^{|R^k|} y^{CR}_{kr} = 1 && \forall k \in K
\end{aligned} \tag{21}
$$

For critical region r in the kth fold, the affine expressions for the model weights are $w_k = A_{kr} + b_{kr}$ and the constant bias is $b_k = G_{kr} + h_{kr}$. These define the decision boundary $w_k\phi(x_{ik}) + b_k$ that classifies each observation into either $+1$ or -1 class, depending on the sign of its evaluated value (positive is $+1$ and negative is -1).

The outer level objective in the bilevel optimization is to minimize the misclassification rate across all folds. For the kth fold with N^{tst}_k sample points in the testing set, the misclassification rate is defined below.

$$
\frac{1}{N^{tst}_k} \sum_{i=1}^{N^{tst}_k} F(y_{ik} \ne \hat{y}_{ik}) \tag{22}
$$

Here, $\hat{y}_{ik}$ is the predicted class label for the ith observation in the testing set for the kth fold, and y_{ik} is the given class label. F is an indicator function that equals 1 if $y_{ik} \ne \hat{y}_{ik}$ and 0 if $y_{ik} = \hat{y}_{ik}$. If $F(y_{ik} \ne \hat{y}_{ik}) = 0$, then the ith observation is correctly classified. Otherwise, it is misclassified.

To capture this discrete decision and identify a function form for F, $\hat{y}_{ik}$ is redefined as a binary variable equal to 1 for a prediction belonging to the $+1$ class and equal to 0 for a prediction belonging to the -1 class. During testing validation in the outer level, the $+1$ and -1 class labels y_{ik} are also redefined as 1 and 0, respectively, to accommodate the binary variable $\hat{y}_{ik}$. However, during training (Eq. (20)) and to compute the critical regions (Eq. (21)), the class labels y_{ik} remain $+1$ and -1. In this way, the misclassification rate is rewritten as a quadratic loss function.

$$
\frac{1}{N^{tst}_k} \sum_{i=1}^{N^{tst}_k} (y_{ik} - \hat{y}_{ik})^2 \tag{23}
$$

$(y_{ik} - \hat{y}_{ik})^2$ equals 1 for a misclassified observation, when $y_{ik} = 1$ & $\hat{y}_{ik} = 0$ or $y_{ik} = 0$ & $\hat{y}_{ik} = 1$. $(y_{ik} - \hat{y}_{ik})^2$ equals 0 for a correctly classified observation, when $y_{ik} = 1$ & $\hat{y}_{ik} = 1$ or $y_{ik} = 0$ & $\hat{y}_{ik} = 0$. The reason for a redefinition of the class labels during testing validation is clear in Eq. (23).

To connect the critical regions in the inner level describing the trained decision boundaries ($w_k\phi(x_{ik}) + b_k$) to the misclassification rate in the outer level, two additional Big-M constraints are needed to handle the redefinition of class labels made between the training and testing validation.

$$
\begin{aligned}
& w_k\phi(x_{ik}) + b_k \ge M(\hat{y}_{ik} - 1) && \forall k \in K, \forall i \in I^k \\
& w_k\phi(x_{ik}) + b_k \le M\hat{y}_{ik} && \forall k \in K, \forall i \in I^k
\end{aligned} \tag{24}
$$

I^k is the set of all observations in the testing set in the kth fold. x_{ik} are the predictors for the ith observation in the kth fold. When the decision boundary is positive ($+1$ class), $\hat{y}_{ik} = 1$. When the decision boundary is negative (-1 class), $\hat{y}_{ik} = 0$. Therefore, Eq. (24) captures the binary decision of predicting a class label and connects the learned w_k & b_k from the critical regions (Eq. (21)) to the misclassification rate in Eq. (23).

Table 3
Model statistics for training L_1-norm SVM pLPs (Eq. (20)).

Fold #	X^{trn}_k $(i \times j)$	# Constraints × # Variables	# Critical regions
1	92 × 9	202 × 111	68
2	93 × 9	204 × 112	86
3	93 × 9	204 × 112	80
4	93 × 9	204 × 112	77
5	93 × 9	204 × 112	69

The final HY-POP formulation (MIQP) of LP L_1-norm SVM hyperparameter optimization comprises of Eqs. (21), (23), and (24) and is shown below.

$$
\begin{aligned}
\min_{C, w_k, b_k, \hat{y}_{ik}, y^{CR}_{kr}} \quad & \frac{1}{|K|} \sum_{k=1}^{|K|} \frac{1}{N^{tst}_k} \sum_{i=1}^{N^{tst}_k} (y_{ik} - \hat{y}_{ik})^2 \\
\text{s.t.} \quad & w_k\phi(x_{ik}) + b_k \ge M(\hat{y}_{ik} - 1) && \forall k \in K, \forall i \in I^k \\
& w_k\phi(x_{ik}) + b_k \le M\hat{y}_{ik} && \forall k \in K, \forall i \in I^k \\
& w_k \le A_{kr}C + b_{kr} + M(1 - y^{CR}_{kr}) && \forall k \in K, \forall r \in R^k \\
& w_k \ge A_{kr}C + b_{kr} + M(y^{CR}_{kr} - 1) && \forall k \in K, \forall r \in R^k \\
& b_k \le G_{kr}C + h_{kr} + M(1 - y^{CR}_{kr}) && \forall k \in K, \forall r \in R^k \\
& b_k \ge G_{kr}C + h_{kr} + M(y^{CR}_{kr} - 1) && \forall k \in K, \forall r \in R^k \\
& \sum_{r=1}^{|R^k|} LB^{CR}_{kr} y^{CR}_{kr} \le C \le \sum_{r=1}^{|R^k|} UB^{CR}_{kr} y^{CR}_{kr} && \forall k \in K \\
& \sum_{r=1}^{|R^k|} y^{CR}_{kr} = 1 && \forall k \in K
\end{aligned} \tag{25}
$$

Eq. (25) represents a new construction for the hyperparameter optimization of LP L_1-norm SVM through K-fold cross-validation. This HY-POP formulation is next demonstrated on a breast cancer data example.

6.1. Breast cancer data example

A dataset of 116 samples with 9 predictors for breast cancer (Patrício et al., 2018) is downloaded from the UCI Machine Learning Repository. The data values are normalized and randomly divided into 5 folds. The fold identification of the data points is provided in the supplementary material. Healthy patients are labeled $+1$, and cancer patients are labeled -1. The predictors are age, BMI, and levels of glucose, insulin, HOMA, leptin, adiponectinresistin, & MCP-1. It is assumed that the predictors are linear, $\phi(X) = X$. Therefore, the proposed LP L_1-norm SVM that is trained is the following.

$$
\begin{aligned}
y_{cancer} = {} & w_1 x_{age} + w_2 x_{BMI} + w_3 x_{glu} + w_4 x_{insu} + w_5 x_{HOMA} \\
& + w_6 x_{lep} + w_7 x_{adi} + w_8 x_{res} + w_9 x_{MCP1} + b
\end{aligned} \tag{26}
$$

Each training LP L_1-norm SVM (Eq. (20)) for each fold is formulated in MATLAB 2019b. The critical regions are solved using the POP toolbox (Oberdieck et al., 2016b) with the CPLEX LP solver and the geometrical algorithm. Since Eq. (20) also has only one hyperparameter, a geometrical algorithm is chosen. The model statistics for each training optimization problem (Eq. (20)) is depicted in Table 3. An example result of the critical regions for fold #3 is shown in Fig. 11. Like the LASSO regression, a piecewise relationship between the model weights w & b with the hyperparameter C is also observed here. We expected this piecewise behavior from

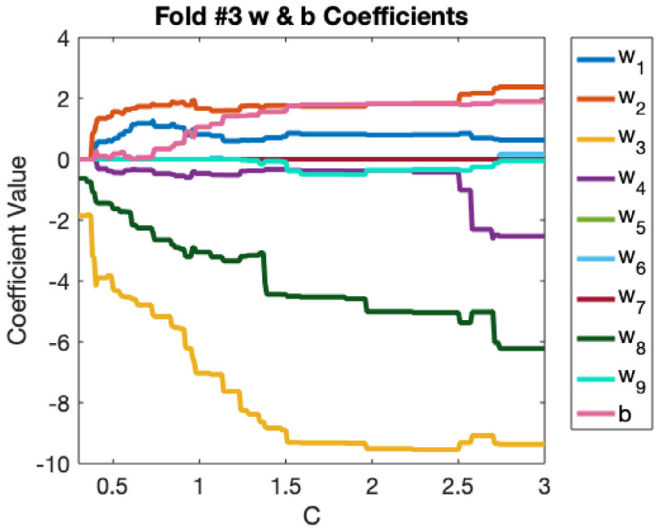

Fig. 11. SVM regularization path for the breast cancer data in fold #3 from the training pLP (Eq. (20)) solved using POP (Oberdieck et al., 2016b).

the affine expressions, $w(C)$ & $b(C)$, determined from the parametric programming solution to Eq. (20).

This piecewise relationship is referred to as the SVM regularization path (Hastie et al., 2009) because earlier investigations (Hastie et al., 2004; Zhu et al., 2004) noticed similarities between the algorithms (LAR and coordinate descent) used to solve SVM and LASSO regression. In general, this is expected because all these solution methods fall under the general theory of parametric programming (Eq. (7)). Interpretation of the SVM regularization path is different from LASSO regression because the former does not perform feature selection in its current formulation. Greater absolute values for w & b as C gets larger indicate more important features for building the decision boundary between the two classes. For fold #3 in Fig. 11, it appears that glucose level is the most significant factor for distinguishing breast cancer patients from healthy individuals compared to the other features.

Unlike the LASSO regression, there are many more individual line segments (critical regions) that are stitched together to represent the optimal solution profile in Fig. 11. This creates a very non-smooth behavior in $w(C)$ & $b(C)$ and the resulting misclassification error profile (Fig. 12). Profiles of the SVM regularization path and validation error for all folds are found in the supplementary material. The non-smoothness is expected because the objective (loss) function in Eq. (25) is quadratic with respect to a binary variable $\hat{y}_{ik}$ and not a continuous variable. The shaded blue area in Fig. 12b represent one standard error above and below the average misclassification rate.

Because of this non-smooth behavior in the misclassification error profile, characteristic of classification problems in machine learning, it is inherently more difficult to find an accurate approximation to an optimal C from a discretized grid search. The optimal C is more sensitive to the granularity of the discretization. This is one advantage of having the optimal solution in explicit form through a HY-POP approach. The overall MIQP model (Eq. (25)) for the breast cancer example has 7847 constraints, 51 continuous variables, and 496 binary variables. From minimizing the MIQP, the optimal C value is 1.4869 with an average misclassification error of 0.2681. Table 4 displays the confusion matrix. The specificity is 0.75, indicating that 75% of cancer patients are accurately identified as such. The sensitivity is 0.71875, indicating that about 72% of healthy individuals are correctly classified.

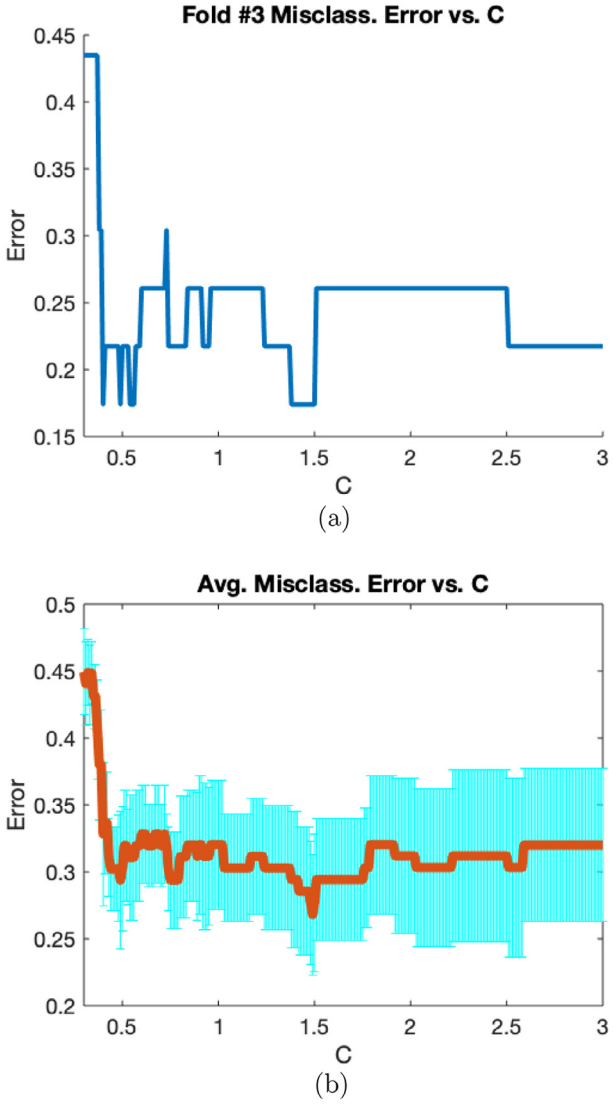

Fig. 12. (a) Misclassification error in fold #3 (b) Misclassification error averaged across all the folds.

Table 4
Confusion matrix for 5-fold cross-validation on breast cancer data.

		Predicted class	
		+1 (Healthy)	−1 (Cancer)
Actual Class	+1 (Healthy)	39	13
	−1 (Cancer)	18	46

Table 5
Optimal values and CPU times for 5-fold cross-validation on breast cancer data.

Method	Optimal error	Optimal C	CPU time[a](s)
pLP + MIQP	0.2681	1.4869	50.49 ± 0.61

[a] Averaged over 10 runs

Table 5 shows the computational results. A longer CPU time for this breast cancer example is observed than for the LASSO regression on the ammonia reactor data because there are many more critical regions to compute in this instance. Even though Eq. (20) is a pLP, it has more constraints in its problem than Eq. (15), which causes more active set explorations for the parametric programming algorithm to solve the former. pLPs are actually more diffi-

cult to solve than pQPs as well due to degeneracy issues. Likewise, the MIQP here is also larger than the one earlier for LASSO. About 87% of the CPU time is spent calculating the critical regions for the folds.

The complexity of the pLP for LP L_1-norm SVM (Eq. (20)) grows with the number of samples i and features j. Therefore, it is likely that Eq. (20) becomes more difficult to solve as X_k^{trn} gets larger in size. Specifically, the number of constraints scale with $2i + 2j$ and the number of variables scale with $i + 2j + 1$ in Eq. (20). Since active set based strategies to solve parametric problems depend on choosing n active constraints for n variables to square the system, the pLP becomes easier to solve when $2i + 2j$ is closer in value to $i + 2j + 1$, so that there are fewer unique active set combinations to explore. It is expected that the L_1-norm SVM pLP will remain tractable for a reasonably larger X_k^{trn}, provided the number of variables is close to the number of constraints. The upper bound on the computational performance of HY-POP for LP L_1-norm SVM is open for further study.

7. Conclusion

The novelty in this work is constructing hyperparameter optimization through K-fold cross-validation as a bilevel optimization problem that is exactly solvable as a single level optimization through parametric programming. We refer to this as the bilevel & parametric optimization approach to hyperparameter optimization (HY-POP).

This parametric programming perspective ties together previous studies that first recognized the regularization paths of LASSO regression and SVM as piecewise linear functions and extends these results to optimize hyperparameters in K-fold cross-validation. The HY-POP approach to hyperparameter optimization is demonstrated on ammonia reactor data, a QP example, and breast cancer data, a LP example.

Advantages of recognizing hyperparameter optimization as a parametric programming problem are threefold. First, the HY-POP approach to hyperparameter optimization is applicable to any general machine learning algorithm that is a LP/QP model. In fact, mixed-integer linear or quadratic (MILP/MIQP) models can also be used because parametric programming theory exists for these problem types. Second, when there multiple hyperparameters in a machine learning model (a common occurrence), there is theory to solve these problems exactly through multi-parametric programming. Third, and most importantly, no discretization of the hyperparameter space is required for HY-POP.

Finally, the aims of this work are to lay the introductory foundation and present an unified view to hyperparameter optimization of machine learning models from a parametric programming perspective. The multi-parametric programming, mixed-integer, and computational aspects of a HY-POP approach to hyperparameter optimization are subjects of further investigation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

William W. Tso: Conceptualization, Methodology, Software, Formal analysis, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Baris Burnak:** Conceptualization, Writing - review & editing. **Efstratios N. Pistikopoulos:** Writing - review & editing, Supervision.

Acknowledgments

WWT gratefully acknowledges financial support from Royal Dutch Shell and the Texas A&M Energy Institute. BB gratefully acknowledges financial support from the National Science Foundation (NSF CBET-1705423). WWT thanks Justin P. Katz for his assistance with the POP toolbox. WWT appreciates Burcu Beykal and Melis Onel for the initial conceptualization of Figs. 1, 2 and 4.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2020.106902.

References

Abu-Mostafa, Y.S., Magdon-Ismail, M., Lin, H.-T., 2012. Learning From Data, 4. AML-Book New York, NY, USA.
Avraamidou, S., Pistikopoulos, E.N., 2017. A multiparametric mixed-integer bi-level optimization strategy for supply chain planning under demand uncertainty. IFAC-PapersOnLine 50 (1), 10178–10183.
Avraamidou, S., Pistikopoulos, E.N., 2019. B-POP: Bi-level parametric optimization toolbox. Comput. Chem. Eng. 122, 193–202.
Avraamidou, S., Pistikopoulos, E.N., 2019. A multi-parametric optimization approach for bilevel mixed-integer linear and quadratic programming problems. Comput. Chem. Eng. 125, 98–113.
Barratt, S., Sharma, R., 2018. Optimizing for generalization in machine learning with cross-validation gradients. arXiv:1805.07072.
Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.N., 2002. The explicit linear quadratic regulator for constrained systems. Automatica 38 (1), 3–20.
Bengio, Y., 2000. Gradient-based optimization of hyperparameters. Neural Comput. 12 (8), 1889–1900.
Bennett, K.P., Hu, J., Ji, X., Kunapuli, G., Pang, J.-S., 2006. Model selection via bilevel optimization. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings. IEEE, pp. 1922–1929.
Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13 (Feb), 281–305.
Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554.
Beykal, B., Boukouvala, F., Floudas, C.A., Pistikopoulos, E.N., 2018. Optimal design of energy systems using constrained grey-box multi-objective optimization. Comput. Chem. Eng. 116, 488–502.
Beykal, B., Boukouvala, F., Floudas, C.A., Sorek, N., Zalavadia, H., Gildin, E., 2018. Global optimization of grey-box computational systems using surrogate functions and application to highly constrained oil-field operations. Comput. Chem. Eng. 114, 99–110.
Boser, B.E., Guyon, I.M., Vapnik, V.N., 1992. A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. ACM, pp. 144–152.
Bradley, P.S., Mangasarian, O.L., 1998. Feature selection via concave minimization and support vector machines.. In: ICML, 98, pp. 82–90.
Burnak, B., Diangelakis, N.A., Katz, J., Pistikopoulos, E.N., 2019. Integrated process design, scheduling, and control using multiparametric programming. Comput. Chem. Eng. 125, 164–184.
Claesen, M., De Moor, B., 2015. Hyperparameter search in machine learning. arXiv:1502.02127.
Colson, B., Marcotte, P., Savard, G., 2007. An overview of bilevel optimization. Ann. Oper. Res. 153 (1), 235–256.
Demirhan, C.D., Tso, W.W., Powell, J.B., Pistikopoulos, E.N., 2019. Sustainable ammonia production through process synthesis and global optimization. AIChE J. 65 (7), e16498.
Diangelakis, N.A., Burnak, B., Katz, J., Pistikopoulos, E.N., 2017. Process design and control optimization: asimultaneous approach by multi-parametric programming. AIChE J. 63 (11), 4827–4846.
Diangelakis, N.A., Pappas, I.S., Pistikopoulos, E.N., 2018. On multiparametric/explicit NMPC for quadratically constrained problems. IFAC-PapersOnLine 51 (20), 400–405.
Diaz, G.I., Fokoue-Nkoutche, A., Nannicini, G., Samulowitz, H., 2017. An effective algorithm for hyperparameter optimization of neural networks. IBM Journal of Research and Development 61 (4/5). 9–1
Domínguez, L.F., Pistikopoulos, E.N., 2010. Multiparametric programming based algorithms for pure integer and mixed-integer bilevel programming problems. Comput. Chem. Eng. 34 (12), 2097–2106.
Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al., 2004. Least angle regression. Ann. Stat. 32 (2), 407–499.
Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., Leyton-Brown, K., 2013. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In: NIPS workshop on Bayesian Optimization in Theory and Practice, 10, p. 3.

Faísca, N.P., Dua, V., Rustem, B., Saraiva, P.M., Pistikopoulos, E.N., 2007. Parametric global optimisation for bilevel programming. J. Glob. Optim. 38 (4), 609–623.
Foo, C.-s., Do, C.B., Ng, A.Y., 2008. Efficient multiple hyperparameter learning for log-linear models. In: Advances in Neural Information Processing Systems, pp. 377–384.
Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M., 2018. Bilevel programming for hyperparameter optimization and meta-learning. arXiv:1806.04910.
Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. 33 (1), 1–22.
Gao, J., Kwan, P.W., Shi, D., 2010. Sparse kernel learning with lasso and Bayesian inference algorithm. Neural Netw. 23 (2), 257–264.
Gold, C., Holub, A., Sollich, P., 2005. Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. Neural Netw. 18 (5–6), 693–701.
Hastie, T., Rosset, S., Tibshirani, R., Zhu, J., 2004. The entire regularization path for the support vector machine. J. Mach. Learn. Res. 5 (Oct), 1391–1415.
Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.
Hutter, F., Lücke, J., Schmidt-Thieme, L., 2015. Beyond manual tuning of hyperparameters. KI-Künstliche Intell. 29 (4), 329–337.
James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning, 112. Springer.
Keerthi, S.S., Sindhwani, V., Chapelle, O., 2007. An efficient method for gradient-based adaptation of hyperparameters in SVM models. In: Advances in Neural Information Processing Systems, pp. 673–680.
Klatzer, T., Pock, T., 2015. Continuous hyper-parameter learning for support vector machines. In: Computer Vision Winter Workshop (CVWW), pp. 39–47.
Koch, P., Golovidov, O., Gardner, S., Wujek, B., Griffin, J., Xu, Y., 2018. Autotune: a derivative-free optimization framework for hyperparameter tuning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 443–452.
Liu, R., Liu, E., Yang, J., Li, M., Wang, F., 2006. Optimizing the hyper-parameters for SVM by combining evolution strategies with a grid search. In: Intelligent Control and Automation. Springer, pp. 712–721.
Luo, G., 2016. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Netw. Model. Anal. Health Inform. Bioinform. 5 (1), 18.
MacKay, M., Vicol, P., Lorraine, J., Duvenaud, D., Grosse, R., 2019. Self-tuning networks: bilevel optimization of hyperparameters using structured best-response functions. arXiv:1903.03088.
Oberdieck, R., Diangelakis, N.A., Avraamidou, S., Pistikopoulos, E.N., 2017. On unbounded and binary parameters in multi-parametric programming: applications to mixed-integer bilevel optimization and duality theory. J. Glob. Optim. 69 (3), 587–606.
Oberdieck, R., Diangelakis, N.A., Nascu, I., Papathanasiou, M.M., Sun, M., Avraamidou, S., Pistikopoulos, E.N., 2016. On multi-parametric programming and its applications in process systems engineering. Chem. Eng. Res. Des. 116, 61–82.
Oberdieck, R., Diangelakis, N.A., Papathanasiou, M.M., Nascu, I., Pistikopoulos, E.N., 2016. Pop–parametric optimization toolbox. Ind. Eng. Chem. Res. 55 (33), 8979–8991.
Ogumerem, G.S., Pistikopoulos, E.N., 2019. Parametric optimization and control toward the design of a smart metal hydride refueling system. AIChE J. 65 (10), e16680.
Onel, M., Burnak, B., Pistikopoulos, E.N., 2019. Integrated data-driven process monitoring and explicit fault-tolerant multiparametric control. Ind. Eng. Chem. Res..
Onel, M., Kieslich, C.A., Guzman, Y.A., Floudas, C.A., Pistikopoulos, E.N., 2018. Big data approach to batch process monitoring: simultaneous fault detection and diagnosis using nonlinear support vector machine-based feature selection. Comput. Chem. Eng. 115, 46–63.
Onel, M., Kieslich, C.A., Pistikopoulos, E.N., 2019. A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: application to the tennessee eastman process. AIChE J. 65 (3), 992–1005.
Patrício, M., Pereira, J., Crisóstomo, J., Matafome, P., Gomes, M., Seiça, R., Caramelo, F., 2018. Using Resistin, glucose, age and BMI to predict the presence of breast cancer. BMC Cancer 18 (1), 29.
Pedregosa, F., 2016. Hyperparameter optimization with approximate gradient. arXiv:1602.02355.
Pistikopoulos, E., 2009. Perspectives in multiparametric programming and explicit model predictive control. AIChE J. 55 (8), 1918–1925.
Pistikopoulos, E.N., 2012. From multi-parametric programming theory to MPC-on-a-chip multi-scale systems applications. Comput. Chem. Eng. 47, 57–66.
Pistikopoulos, E. N., Diangelakis, N. A., Oberdieck, R., 2020. Multi-Parametric Optimization and Control. Vol. 1.
Pistikopoulos, E.N., Georgiadis, M.C., Dua, V., 2011. Multi-Parametric Programming: Theory, Algorithms, and Applications, 1.
Rios, L.M., Sahinidis, N.V., 2013. Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. 56 (3), 1247–1293.
Sinha, A., Malo, P., Deb, K., 2017. A review on bilevel optimization: from classical to evolutionary approaches and applications. IEEE Trans. Evol. Comput. 22 (2), 276–295.
Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp. 2951–2959.
Tian, Y., Pappas, I., Burnak, B., Katz, J., Pistikopoulos, E.N., 2020. A systematic framework for the synthesis of operable process intensification systems–reactive separation systems. Comput. Chem. Eng. 134, 106675.
Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. J. R. Stat. Soc. 58 (1), 267–288.
Tso, W.W., Demirhan, C.D., Powell, J.B., Pistikopoulos, E.N., 2018. Toward optimal synthesis of renewable ammonia and methanol processes (RAMP). In: Computer Aided Chemical Engineering, 44. Elsevier, pp. 1705–1710.
Wilson, Z.T., Sahinidis, N.V., 2017. The ALAMO approach to machine learning. Comput. Chem. Eng. 106, 785–795.
Zhu, J., Rosset, S., Tibshirani, R., Hastie, T.J., 2004. 1-norm support vector machines. In: Advances in Neural Information Processing Systems, pp. 49–56.