Pattern Recognition 170 (2026) 111961

Contents lists available at ScienceDirect

Pattern Recognition

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

Leveraging pre-trained models for kernel machines

Yawen Chen [a], Zeyi Wen [b,c], Jian Chen [d,*], Jin Huang [e]

[a] School of Artificial Intelligence, Henan University, China
[b] The Hong Kong University of Science and Technology (Guangzhou), China
[c] The Hong Kong University of Science and Technology, Hong Kong SAR, China
[d] South China University of Technology, China
[e] South China Normal University, China

ARTICLE INFO

Keywords:
Kernel machines
Pre-training
Model inferring

ABSTRACT

Pre-training techniques have successfully promoted the training of neural networks. Since neural networks and kernel machines share similar properties, such as both learning the problems by the non-linear projection on features and both being capable of handling complex tasks, the idea of pre-training may also help kernel machines achieve promising training speed. However, existing pre-training-based kernel machine solvers show limited improvements on efficiency when the hyper-parameter varies. To effectively reduce the training cost, we propose a novel method that can make efficient use of pre-trained models to infer kernel machine models with different hyper-parameters. Our pre-training-based method is built on top of theoretical foundations. The difference between the model inferred based on pre-training and the optimal model is theoretically bounded by a constant. Experimental results show that our method can save an order of magnitude of training time compared with the existing approach while producing competitive accuracy.

1. Introduction

Pre-training has advanced large-scale neural network training in many real-world problems from natural language processing [1] to computer vision [2]. The convergence speed of neural network training as well as model quality can be substantially improved using pre-trained models. Inspired by these advancements, we explore whether similar benefits could extend to kernel machine training. This potential stems from key similarities between neural networks and kernel machines. First, neural networks have feature transformation, and kernel machines have the similar property of implicit data mapping to feature spaces through kernel functions. Second, several studies demonstrate that kernel machine based solutions can achieve comparable performance with neural networks when solving the popular sentiment analysis problem [3] and the random label problems [4] with robust generalization. These similarities suggest that applying pre-training to kernel machines is promising and may alleviate the expensive computational cost in kernel machine learning.

Existing approaches have investigated pre-training strategies for kernel methods. However, these approaches have shown notable limitations. Most of them are effective only when the data vary [5] and are restricted to specific kernel machine forms [6]. Several methods [7,8] avoid the training by deducing the solution path to the kernel machine with hyper-parameter variations. Nevertheless, they exhibit computational complexity comparable to training a single kernel machine from scratch [9]. The marginal efficiency improvements are resulted primarily from the expensive kernel matrix computations required by conventional pre-training-based approaches.

To overcome this limitation, we propose a novel kernel machine training method that can achieve substantial computational advantages. Our key idea is to infer the new kernel machine model with different hyper-parameters based on pre-trained models while eliminates the need for repeated kernel matrix computations. We theoretically prove that the difference between the inferred model and the optimal one (i.e., training from scratch) is bounded by a constant. To further generalize our pre-training-based method, we provide a theoretical foundation for using pre-trained models when the regularization constant of the pre-trained model is different from the targeted model. We demonstrate that the kernel machine model is inversely proportional to the regularization constant. Thus, we can obtain the new kernel machines by multiplying the pre-trained model by a constant factor. Moreover, we take the Support Vector Machine (SVM) as an example to further elaborate our theoretical results. Our method benefits from a linear time complexity.

A paramount application of our pre-training-based method is for model selection, where a model can be quickly trained for the new

* Corresponding author.
E-mail addresses: ywchen@henu.edu.cn (Y. Chen), wenzeyi@ust.hk (Z. Wen), ellachen@scut.edu.cn (J. Chen), huangjin@m.scnu.edu.cn (J. Huang).

https://doi.org/10.1016/j.patcog.2025.111961
Received 18 February 2023; Received in revised form 2 June 2025; Accepted 2 June 2025
Available online 20 June 2025

hyper-parameters based on the previously trained model. Experimental results on the model selection show that our method can help reduce the elapsed time of model selection by an order of magnitude. Our method can achieve competitive test accuracy as the optimal model training from scratch. Furthermore, the experimental study shows that the gap between the objective values of the pre-training-based model and the optimal one is extremely close. Our findings indicate that the idea of pre-training is an excellent facilitator for hyper-parameter tuning in kernel machines. In summary, we make the following major contributions in this paper.

- To accelerate the kernel machine training, we propose a method that can make efficient use of the pre-trained models to infer new kernel machine models with different hyper-parameters.
- We develop theoretical foundations for reusing pre-trained kernel machines and further verify the theoretical results on common kernel machines, i.e., SVMs.
- We conduct extensive experiments to evaluate our proposed method. The experimental results show that our method can save an order of magnitude of elapsed time when performing hyper-parameter tuning, and the model inferred based on the pre-trained model yields competitive performance to the optimal model on prediction.

2. Related work

To efficiently train the kernel machine algorithms [10], existing approaches try to reduce the size of training data [11,12], or narrow down the hyper-parameter searching space [13]. Pre-training, which has been widely studied in neural networks such as BERT [1], GPT [14] and DeepSeek [15], however, has not been adequately explored for traditional machine learning.

2.1. Pre-training in kernel machine initialization

Some studies consider previously trained kernel machines as efficient initialization but have notable limits as described below. Wen et al. [5] explored three alpha seeding strategies based on k-fold CV. The dual solution in the previous CV round are adjusted in three different ways to initialize the solution in the next round which leads to minimal violation of the optimality condition. Tommasi and Caputo [16] interpolate the model trained on the old data set into the objective of a new problem with a scaling factor to control how close are the new and old models. Tsai et al. [6] invented incremental and decremental algorithms for linear SVM classifiers. The new dual solution is initialized by multiplying the original optimal dual solution by a ratio. Multiple incremental and decremental learning [17] implement Cholesky factorization for kernel matrix to efficiently update the model when multiple instances are added or removed simultaneously. Gâlmeanu and Andonie [18] introduced a weighted incremental–decremental SVM (WIDSVM) to handle concept drift. WIDSVM adjusts the weights of samples and constraints based on previous SVM solutions while preserving the optimality for historical data fulfilled. Chen et al. [19] reformulated the infinitesimal annealing algorithm to continuously update the solutions of transductive SVMs with streaming samples. Xu et al. [20] use incremental SVMs to generate initial solution population for the new multi-objective optimization problems. Budget online learning [21] constrains a budget number of support vectors and updates the pre-trained SVM with efficient matrix computation when new instances are added. Chu et al. [22] proposed a warm start method for linear classifiers such as linear SVMs. It increases the dual solution by a certain number when the regularization constant changes.

While existing studies leverage pre-trained models to warm-start training, they suffer from key limitations. Most approaches [5,18,19] are tailored to specific kernel machine types, which restricts their generalization to diverse models. More importantly, these initialization methods [6,17,20] typically require the fixed hyper-parameter across tasks. This severely limits their applicability to scenarios with varying hyper-parameter spaces.

2.2. Pre-training in solution deducing

In addition to initialization, some researchers use pre-trained models to compute a direct numerical solution path to the machine learning problem with new hyper-parameters. Hastie et al. [9] derived the regularization path for the SVM which uses the pre-trained SVM to compute the exact solution to an SVM problem with different regularization constants. In addition to the standard SVMs, Gu et al. [23] computed the solution surface and validation error surface for the biased SVM which has two regularization parameters (C_+, C_-). Based on the surfaces, the solution and regularization parameters with the global minimum CV error can be found. Gu and Sheng [24] presented a new equivalent dual objective and a robust regularization path based on lower upper decomposition for ν-support vector classification (SVC), which avoids exceptions and singularities in standard ν-SVC path. Zhai et al. [25] proposed to compute numerical solutions for both the regularization parameter and the ramp parameter used in robust SVMs. The computational complexity of these approaches [9,24,25] can be unified as $\mathcal{O}(c_1 n^2 m + c_2 n m^2)$, where m is the average number of in-bound samples, n is the number of training samples, and c_1, c_2 are coefficients that scale with n and m, respectively. Suzumura and Ogawa [26] identified the necessary and sufficient conditions for local optimal solutions in robust SVMs and developed a homotopy approach to derive the solution based on these conditions. Gunter and Zhu [7] proposed a regularization path for support vector regression. The total complexity of computing the exact regularization path in each iteration is $\mathcal{O}(nm)+\mathcal{O}(nm^2)$. Gu [27] tracked the solution of support vector ordinal regression (SVOR) which solves multiple binary classifications when the regularization parameter varies. Its complexity is comparable to the training of a single SVOR [27].

While regularization parameters are important, hyper-parameters in kernel functions are equally crucial in governing kernel machine behavior. Wang et al. [8] introduced a kernel path that calculates the exact solution to the problem with neighborhood kernel or regularization parameters of the pre-trained model, yet it faces numerical problems and exhibits an exponential time complexity of $\mathcal{O}(n^2 + m^3)$. Therefore, Giesen et al. [28] proposed an ϵ-approximate kernel path to avoid the high computation cost of exact path. Despite the authors' methodological contributions, our empirical findings in Section 6.1 show that the approximate kernel path method performs substantially slower than training from scratch. Its excessively long execution time makes it computationally infeasible for practical applications.

Although current methods provide a theoretical shortcut to the solution path, they incur notable computational costs. Their computational burdens such as $\mathcal{O}(c_1 n^2 m+c_2 n m^2)$ [9,24,25] or $\mathcal{O}(n^2+m^3)$ [8], are similar to the $\mathcal{O}(n^2 d)$ or $\mathcal{O}(n^3 d)$ [29] complexity of single kernel machine training as demonstrated in the previous study [9,24,27], where d is the feature dimensionality. Furthermore, the kernel path [8] even suffers from numerical issues which lead to instability in convergence. While some approximation techniques aim to mitigate these costs, they still rely on maintaining and updating kernel matrices [28] which imposes a non-negligible overhead empirically. In contrast, our method circumvents the recomputation of kernel matrix during model inference. The computational complexity of our method inferring the new kernel machine model is $\mathcal{O}(n)$ which is much less than the existing solution path algorithms.

3. Preliminaries on kernel machines

A kernel machine projects a non-linear problem into a feature space where the problem may be solved linearly [30,31]. Formally, suppose we have a training data set $\{X, \boldsymbol{y}\}$. The data set consists of n training instances where $\{X \in \mathbb{R}^{n\times d}, \boldsymbol{y} \in \mathbb{R}^n\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$, and $(\boldsymbol{x}_i, y_i)$ denotes the instance $\boldsymbol{x}_i \in \mathbb{R}^d$ with its label y_i. Instead of solving the primal problem, we focus on solving the dual problem

where the kernel machine training aims to find the optimal weight vector $\boldsymbol{\alpha}$ that satisfies the following objective.

$$\min L(\boldsymbol{\alpha}) = \frac{1}{n}\sum_{i=1}^{n} l(f(\boldsymbol{x}_i), y_i) + \frac{\lambda}{2}\|\sum_{j=1}^{n}\alpha_j\phi(\boldsymbol{x}_j)\|^2,$$
$$\text{subject to} \quad h_i(X,\boldsymbol{\alpha},\Theta) = 0, \forall i \in \{1,\dots,n_h\}, \quad (1)$$
$$g_j(X,\boldsymbol{\alpha},\Theta) \le 0, \forall j \in \{1,\dots,n_g\}, \ \lambda > 0,$$

where $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_n]^T$ is an n-dimension column vector, each dimension of which corresponds to the contribution of a training instance to the kernel machine. The regularization constant is denoted as λ and Θ is the set of hyper-parameters in kernel machines (i.e., $\Theta = \{\lambda, \theta_1, \theta_2, \dots\}$). The function $\phi(\cdot)$ maps the instances from their original data space to a higher dimensional feature space induced by the kernel function. Based on the *reproducing property* [32], we have the decision function $f(\boldsymbol{x}_i) = \sum_{j=1}^{n}\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ where $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a positive-definite kernel and $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)\rangle$. The inner product $\langle\cdot,\cdot\rangle$ is defined on the *reproducing kernel Hilbert space*. The loss function (i.e., L_1 loss) can be defined as follows.

$$l(f(\boldsymbol{x}_i), y_i) = |\sum_{j=1}^{n}\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - y_i|. \quad (2)$$

The loss is widely used in kernel machines (e.g., SVMs) and is robust to the outliers [33]. We use $h_i(X,\boldsymbol{\alpha},\Theta)$ to represent the ith equality constraint function, and use $g_j(X,\boldsymbol{\alpha},\Theta)$ to represent the jth inequality constraint function. The numbers of equality constraints and inequality constraints are n_h and n_g, respectively. The equality and inequality constraints are affine functions, and the inequality constraints are convex and continuously differentiable, which are common in kernel machines such as SVMs [34].

4. Leveraging pre-trained models for kernel machine training

To mitigate the computational burden in kernel machines, we propose an inference-based approach which efficiently reuses the pre-trained models instead of training from scratch. Our method circumvents the prohibitive cost of kernel matrix recomputation, which has limited the efficiency advantages of using pre-training for kernel machines [7,8]. In the following, we theoretically show that the kernel machine model with a new kernel hyper-parameter can be inferred using the computation of the pre-trained model. To further expand the use of pre-trained models, we theoretically analyze the relationship between the optimal model and the regularization constant. At last, we summarize the entire procedure of our method.

4.1. Reusing pre-trained models for new kernel parameters

The models we aim to train may share the same parameters except for the kernel parameter with pre-trained models. This scenario commonly occurs in many applications, for example, when performing kernel parameter tuning. To avoid the tedious kernel machine training for different kernel parameters, we propose to reuse the computation of pre-trained models to infer new kernel machines. We consider the kernel machine problem as stated in Problem (1) with kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. For clarity, we assume that only one hyper-parameter of the kernel is different from the pre-trained model, and the other hyper-parameters in the kernel are unchanged. The alterable kernel parameter is denoted as σ. Thus the kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be simplified as $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{ij}(\sigma)$. Through our investigation of the behavior of optimal solutions, while changing kernel parameters, we raise a convergence theorem for the solution to the kernel machine problem as follows.

Theorem 1. *Suppose the kernel machine Problem (1) is solved with a fixed regularization constant λ and a continuous positive-definite kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{ij}(\sigma)$. Let $\boldsymbol{\alpha}^* \in \mathbb{R}^n$ and $\boldsymbol{\alpha}'^* \in \mathbb{R}^n$ be the optimal solutions to Problem (1) where the kernel parameters take the values of σ_0 and σ', separately. The parameter σ' is equal to the parameter σ_0 moved by a small step size Δ where $\sigma' = \sigma_0/\Delta$ and $\Delta > 1$. Then the ith dimension of the optimal solution $\boldsymbol{\alpha}'^*$ which is denoted as $\alpha_i'^*$ converges to an approximation $\hat{\alpha}_i'^*$ within a constant range $r(y_i)$ as follows.*

$$(\alpha_i'^* - \hat{\alpha}_i'^*)^2 \le r(y_i), \quad \forall i \in \{1,\dots,n\}, \quad (3)$$

where $\hat{\alpha}_i'^*$ is computed with the centroid of all the pre-trained weights belonging to class y_i using the form below.

$$\hat{\alpha}_i'^* = \frac{\sum_{j\in I(y_i)}\alpha_j^*}{|I(y_i)|} = c(y_i), \quad \forall i \in \{1,\dots,n\}, \quad (4)$$

where α_j^* is the jth element in $\boldsymbol{\alpha}^*$. The centroid is denoted as $c(y_i)$ and the set $I(y_i)$ includes the indices of all the instances in class y_i.

The proof of Theorem 1 is available in Appendix A. According to Theorem 1, when the kernel parameter σ_0 moves to σ' by a small step size Δ, the optimal $\alpha_i'^*$ can be approximated by the centroid of the previous/pre-trained model. The gap between the optimal $\alpha_i'^*$ and the approximated one is bounded by $r(y_i)$ which is a constant. Therefore, the optimal solution $\boldsymbol{\alpha}'^*$ with respect to σ' can be inferred using the equation below.

$$\alpha_i'^* \approx c(y_i) = \frac{\sum_{j\in I(y_i)}\alpha_j^*}{|I(y_i)|}, \quad \forall i \in \{1,\dots,n\}. \quad (5)$$

4.2. Reusing pre-trained models for new regularization constants

Another important hyper-parameter in kernel machines that pre-training can take advantage of, is the regularization constant λ. Next, we show the fundamental theorem of reusing pre-trained models for kernel machine models with different regularization constants. In particular, the theorem establishes a relationship between the solution and the regularization constant.

Theorem 2. *Suppose the kernel machine Problem (1) is solved with a continuous positive-definite kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{ij}(\sigma)$. Let $\boldsymbol{\alpha}^*$ be the optimal solution to Problem (1) where the regularization constant λ_0 and kernel parameter σ_0 are used. Then the following relationship between the weight vector and the regularization constant always holds.*

$$\lambda_0\boldsymbol{\alpha}^{*T} = \boldsymbol{s}^T K(\sigma_0)^{-1},$$

where $\boldsymbol{s} \in \mathbb{R}^n$ is a constant vector with respect to λ and $\boldsymbol{\alpha}$. The matrix $K(\sigma_0)$ is the kernel matrix where $K(\sigma_0) = [k_{ij}(\sigma_0)]_{n\times n}$.

The proof of Theorem 2 can be found in Appendix B. Theorem 2 is a generalization to the warm start strategy [22]. From Theorem 2, we derive that the optimal $\boldsymbol{\alpha}^*$ is inversely proportional to the regularization constant. Therefore, when a regularization constant λ_0 decreases by a small step size Δ to λ' (i.e., $\lambda' = \frac{\lambda_0}{\Delta}$ with $\Delta > 1$), the solution $\boldsymbol{\alpha}'$ to Problem (1) with the regularization constant λ' can be initialized as follows.

$$\boldsymbol{\alpha}' = \Delta\boldsymbol{\alpha}^*. \quad (6)$$

4.3. Summary of our pre-training-based method

With Theorems 1 and 2, we summarize our pre-training-based method for kernel machines below. The time complexity of our method is $\mathcal{O}(n)$.

- When the kernel parameter changes from σ_0 to σ' by a step size Δ (i.e., $\sigma' = \frac{\sigma_0}{\Delta}$ with $\Delta > 1$), the optimal solution $\boldsymbol{\alpha}'^*$ with respect to σ' can be inferred using the pre-trained weight $\boldsymbol{\alpha}^*$ with respect to σ_0 as Eq. (5).
- When the regularization constant changes from λ_0 to λ' by a step size Δ (i.e., $\lambda' = \frac{\lambda_0}{\Delta}$ with $\Delta > 1$), the solution $\boldsymbol{\alpha}'$ with respect to λ' can be initialized using the pre-trained weight $\boldsymbol{\alpha}^*$ with respect to σ_0 as Eq. (6).

4.4. Reusing pre-trained models in kernel machine hyper-parameter tuning

Algorithm 1: Pre-training in kernel machine hyper-parameter tuning.

Input: Training instances X_{tr} with labels $\boldsymbol{y}_{tr}$, validation instances X_{val} with labels $\boldsymbol{y}_{val}$, initial kernel parameter σ_0, initial regularization constant λ_0, and the number of iterations for calibration τ.

Output: The best solution $\boldsymbol{\alpha}^*$ with hyper-parameters λ^* and σ^*

1 $\boldsymbol{\alpha}_0 \leftarrow \mathbf{0}$, $Acc^* \leftarrow \mathbf{0}$, $\lambda \leftarrow \lambda_0$
2 **for** $i \leftarrow 1$ **to** $\mathcal{T}_r$ **do** //$\mathcal{T}_r$ is the total number of iterations for tuning σ
3 $\quad\sigma \leftarrow \sigma_0$
4 $\quad\boldsymbol{\alpha}' \leftarrow$InitAlphaForRegularizer($\boldsymbol{\alpha}_0, \Delta_r$) // Equation (6)
5 $\quad\boldsymbol{\alpha}'^* \leftarrow$ TrainModel($X_{tr}, \boldsymbol{y}_{tr}, \boldsymbol{\alpha}', \lambda, \sigma$)
6 $\quad Acc \leftarrow$EvaluateModel($X_{val}, \boldsymbol{y}_{val}, \boldsymbol{\alpha}'^*, \lambda, \sigma$)
7 $\quad$**if** $Acc > Acc^*$ **then** $Acc^* \leftarrow Acc$, $\boldsymbol{\alpha}^* \leftarrow \boldsymbol{\alpha}'^*$, $\lambda^* \leftarrow \lambda$, $\sigma^* \leftarrow \sigma$
8 $\quad\boldsymbol{\alpha}_0 \leftarrow \boldsymbol{\alpha}'^*$
9 $\quad$**for** $j \leftarrow 1$ **to** $\mathcal{T}_k$ **do** //$\mathcal{T}_k$ is the total number of iterations for tuning γ
10 $\quad\quad\sigma \leftarrow \sigma/\Delta_k$, $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}'^*$ //Δ_k is the step size of tuning γ
11 $\quad\quad\boldsymbol{\alpha}' \leftarrow$InferAlphaForKernel($\boldsymbol{\alpha}$) // Equation (5)
12 $\quad\quad$**if** j % τ == *0* **then** //calibration
13 $\quad\quad\quad\boldsymbol{\alpha}'^* \leftarrow$ TrainModel($X_{tr}, \boldsymbol{y}_{tr}, \boldsymbol{\alpha}', \lambda, \sigma$)
14 $\quad\quad$**else** $\boldsymbol{\alpha}'^* \leftarrow \boldsymbol{\alpha}'$
15 $\quad\quad Acc \leftarrow$EvaluateModel($X_{val}, \boldsymbol{y}_{val}, \boldsymbol{\alpha}'^*, \lambda, \sigma$)
16 $\quad\quad$**if** $Acc > Acc^*$ **then** $Acc^* \leftarrow Acc$, $\boldsymbol{\alpha}^* \leftarrow \boldsymbol{\alpha}'^*$, $\lambda^* \leftarrow \lambda$, $\sigma^* \leftarrow \sigma$
17 $\quad\lambda \leftarrow \lambda/\Delta_r$ //Δ_r is the step size of tuning σ

We provide the pseudocode of our method in the context of hyper-parameter tuning in Algorithm 1. In the outer loop (Line 2), we tune the regularization constant λ. The weight $\boldsymbol{\alpha}$ is initialized using Eq. (6) (Line 4) and the model is then trained with the initialized $\boldsymbol{\alpha}$ (Line 5). We evaluate the model on the validation data and select the hyper-parameters based on the highest validation accuracy (Lines 6–7). In the inner loop, we select the hyper-parameter σ for kernel functions (Line 9). The optimal weight of the pre-trained model is used to infer the weight of model with the next σ (Lines 10 and 11). We notice that if we train and infer the model alternatively, the training time can only be reduced by half at most because we need to train the model every other iteration. Otherwise, if we constantly infer the solution for every upcoming kernel parameter by using the same pre-trained optimal solution, the inferred solution will always be the same due to the unchanged centroid $c(y_i)$. In this way, the inferred solution will be more inaccurate for the problems with new kernel parameters as the kernel parameter gradually changes. The intuition is that when the gap between the new kernel parameter and the initial one becomes larger, the feature space that the problem mapped to is more diverse and so do the optimal solutions. Therefore we design a calibration procedure that uses the inferred solution as initialization and trains the model based on the initialization every τ iterations (Line 12–14) to obtain the optimal solution. The calibration not only reduces the training cost but also mitigates the accumulation of inaccuracy.

5. Theoretical results on non-linear SVMs

The findings in Section 4 are applicable to any kernel machines that satisfy the form in Section 3. Here, we use the non-linear SVM as an example to show how to apply our theoretical results to a specific kernel machine.

First, we demonstrate that non-linear SVMs satisfy Theorem 1. The dual objective of SVM training using positive-definite kernels is defined below.

$$\begin{aligned} &\text{minimize} \quad W(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q(\sigma)\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha}, \\ &\text{subject to} \quad \boldsymbol{\alpha}^T\boldsymbol{y} = \mathbf{0},\ 0 \le \alpha_i \le C,\ \forall i \in \{1,\dots,n\}, \end{aligned} \tag{7}$$

The matrix $Q(\sigma)$ is an $n \times n$ matrix and the element on the ith row and jth column is $Q_{ij}(\sigma)$ where $Q_{ij}(\sigma) = y_i y_j k_{ij}(\sigma)$. The penalty parameter C serves as the regularization constant in SVMs where $C = \frac{1}{\lambda}$. Please note that the hinge loss used in the SVM can be treated as a variant of L_1 loss with the loss value equal to 0 when $1 - y_i(\sum_j \alpha_j k_{ij}(\sigma) + b)$ is less than zero. Suppose when Problem (7) is solved with the hyper-parameter set Θ' where $\Theta' = \{C, \sigma'\}$, the optimal solution to Problem (7) is $\boldsymbol{\alpha}'^*$. As the equality and inequality constraints are affine functions and are convex, for each in-bound support vector $\boldsymbol{x}_i$ where $i \in I'_{inb}$, we obtain the following KKT conditions.

$$\boldsymbol{\alpha}'^{*T} Q(\sigma')_i + \beta_i'^* y_i - 1 = 0,\ \forall i \in I'_{inb} = \{i | 0 < \alpha_i'^* < C\},$$

where $\beta_i'^*$ is the optimal Lagrange multiplier for the ith constraint and $Q_i(\cdot)$ is the ith column in $Q(\cdot)$.

Using the KKT conditions, we can derive that the optimal solution $\alpha_i'^*$ converges to an approximation $\hat{\alpha}_i'^*$ within a constant range $r_{inb}(y_i)$ as follows.

$$(\alpha_i'^* - \hat{\alpha}_i'^*)^2 \le r_{inb}(y_i), \quad \forall i \in I'_{inb}, \tag{8}$$

where the approximation is the centroid of all the in-bound pre-trained weights belonging to class y_i and is computed with the form below.

$$\hat{\alpha}_i'^* = \frac{\sum_{j \in I_{inb}(y_i)} \alpha_j^*}{|I_{inb}(y_i)|} = c_{inb}(y_i), \quad \forall i \in I'_{inb},$$

where $c_{inb}(y_i)$ is the centroid and α_j^* is the jth optimal solution to Problem (7) with hyper-parameters $\{C, \sigma_0\}$. The set $I_{inb}(y_i)$ includes the indices of the in-bound pre-trained weights in class y_i where $I_{inb}(y_i) = \{j | 0 < \alpha_j^* < C \wedge y_j = y_i\}$. Hence, we have Eq. (8) which is consistent with Inequality (3) in Theorem 1. For the bounded instance, its weight α_i^* tends to stay at the bound 0 or C when the parameter changes, and thus we infer the solution directly with the pre-trained solution where $\alpha_i'^* = \alpha_i^*$.

The detailed proof is available in Appendix C and we can easily prove that SVMs satisfy Theorem 2 following the similar flow in the proof of Theorem 2.

6. Experimental studies

We empirically study the efficiency and effectiveness of our method in the context of hyper-parameter tuning, which is paramount in machine learning.

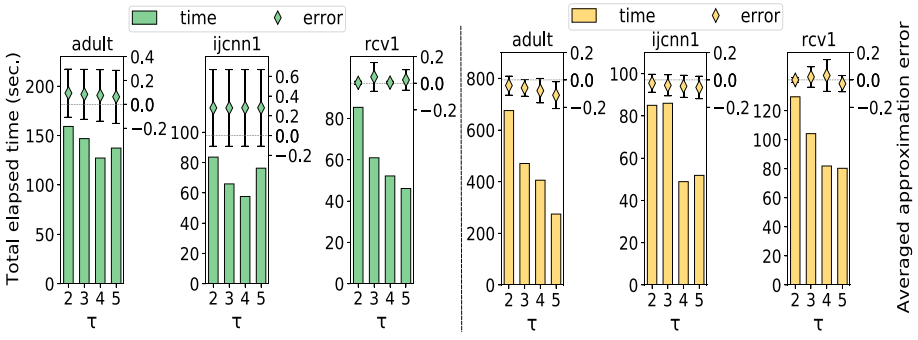
6.1. Data sets and experimental setting

Data sets. Table 1 gives the details of data sets evaluated in our experiments which are binary classification problems downloaded from LIBSVM website [35]. The data sets cover various real-world tasks. For example, *rcv1* is a collection of newswire articles and is used for text categorization. The *real-sim* task tries to classify the real and simulated vehicles from the given documents. The *ijcnn1* data set involves text decoding tasks and the *covtype* data set is used to recognize the forest types. Except for the data sets tested in our experimental studies (i.e., as listed in Table 1), our method is applicable to a broader range of problems addressable by kernel machines, such as image classification with *mnist* data set [36] and sentimental analysis on Wikipedia comments [29]. We leave the exploration of more diverse applications to our future work.

Experimental setting. Our experiments were conducted on a machine with an Intel(R) Xeon(R) E5-2678 CPU and 125 GB main memory running on Linux OS. We choose SVMs with Gaussian kernel (i.e., $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \exp(-\sigma\|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2^2)$) to evaluate our method. Our method and the baselines are implemented in C++ with ThunderSVMs [37]. ThunderSVMs train SVMs using SMO [38] solver with high computing performance. In our experiments, we aim to select the best hyper-parameters, where the regularization constant C is selected in the range $[2^{-3}, 2^{11}]$, and the

Table 1
Data set information.

data set	#train. ins.	#test ins.	#features	data set	#train. ins.	#test ins.	#features
adult	22,696	9,865	123	ijcnn1	49,990	91,701	22
covtype	464,810	116,202	54	rcv1	20,242	677,399	47,236
epsilon	400,000	100,000	2,000	real-sim	57,847	14,462	20,958



(a) $C = 2^{-3}$ (b) $C = 2^{3}$

Fig. 1. Influence of different calibration parameters.

kernel parameter σ is selected in $[2^{-15}, 2^{0}]$. Following common settings in grid search [22,39], the step sizes Δ_r and Δ_k of tuning C and σ are set as 2, respectively. Five-fold cross-validation is used and 20% of the training data serve as the validation set.

Baselines. We compare our method with existing hyper-parameter tuning methods which are: (i) single instance replacement (SIR) [5], (ii) warm start (WS) [22], and (iii) basic tuning. SIR accelerates the cross-validation by initializing the new weights with those from the previous round so that the change of optimality condition of the current training is minimized. Warm start [22] initializes the weight of the model with new regularization constant by multiplying the weight of the pre-trained model with the step size Δ_r which is similar to Eq. (6). In basic tuning, the model is trained from scratch with the weight $\boldsymbol{\alpha}$ initialized to $\mathbf{0}$, which is the default setting in ThunderSVMs [37]. We regard the model obtained in basic tuning as the optimal model in our experiments. Note that we have also evaluated the solution path algorithm [28] but could not collect sufficient empirical results due to the extremely long execution time. It took more than 2 months to finish the grid search on even our smallest tested data set *adult*, which is not practical for industrial applications.

6.2. Selection of the calibration parameter

To fully exploit the strength of our method, we need to first decide the calibration parameter τ. As shown in Algorithm 1, choosing the calibration parameter is a trade-off between the predictive accuracy and the training cost. Therefore we design an experiment to select the calibration parameter that can achieve the best overall performance. We conduct kernel parameter selection using our method and vary τ from 2 to 5 on three data sets considering generality.

Fig. 1 shows the total elapsed time of kernel parameter searching when using each calibration parameter. The averaged approximation error over all the kernel parameters is also added into Fig. 1. The approximation error is computed by subtracting the predictive accuracy of the optimal model from that of the inferred model. Our method achieves better predictive accuracy than the basic method where the approximation error is positive. The approximation error becomes negative and its absolute value becomes larger with the calibration parameter τ increasing. Moreover, when τ is greater than 4, the time saved is not as notable as when it is smaller than 4. Based on the findings, training the model to optimal every 4 iterations yields an overall good performance and hence we recommend to use $\tau = 4$ in applications as well as in our experiments.

6.3. Analysis on efficiency

We perform a grid search on kernel parameters and regularization constants to observe the efficiency of our method compared with baselines. Our method trains the models to optimal when γ is equal to $\{2^{0}, 2^{-4}, 2^{-8}, 2^{-12}\}$, and infers models for other hyper-parameters for that the calibration parameter is set to 4. We record the total elapsed time of the grid search and the results are depicted in Fig. 2. The model selection with our method is 3 to 10 times faster than the baselines on all the tested data sets. The time saving is more prominent for larger data sets such as *covtype* and *epsilon*.

6.4. Analysis on prediction performance

To verify that our method not only takes less training time than the baselines but also achieves competitive performance on prediction, we study the overall prediction performance of our method.

Approximation errors. The approximation error on the test accuracy for each set of tested hyper-parameters is shown in Fig. 3. When γ is equal to one of the values in $\{2^{0}, 2^{-4}, 2^{-8}, 2^{-12}\}$, the model is trained to optimal and thus the approximation error is 0 which we do not show in Fig. 3. The best approximation errors tend to gather in the lower left triangle of each heat map. The predictive performance is either superior to the optimal or satisfactory with an approximation error that is less than −0.02. In conclusion, if there is at least one of the hyper-parameters, either the regularization constant or the kernel parameter is small, our method is then able to infer models that achieve competitive test accuracy as the optimal models. We recommend that practitioners use our method in such circumstances.

Averaged accuracy. We show the training and test accuracy averaged over all the hyper-parameters tested in the grid search in Table 2 where "error" indicates the approximation error. The accuracy achieved by different baselines is arranged in the same column because they produce the same accuracy. The baselines need to fine-tune the models to optimal after utilizing the pre-trained models and thus have the same accuracy under the same hyper-parameters [5]. Our method infers models with a less than 6.13% averaged loss on the test accuracy for most data sets. The best average approximation error is obtained on the *covtype* problem which is −1.12%. However, the averaged test accuracy achieved by our method on *real-sim* is much lower than the baselines. We check the optimal models trained on *real-sim* and find that in certain

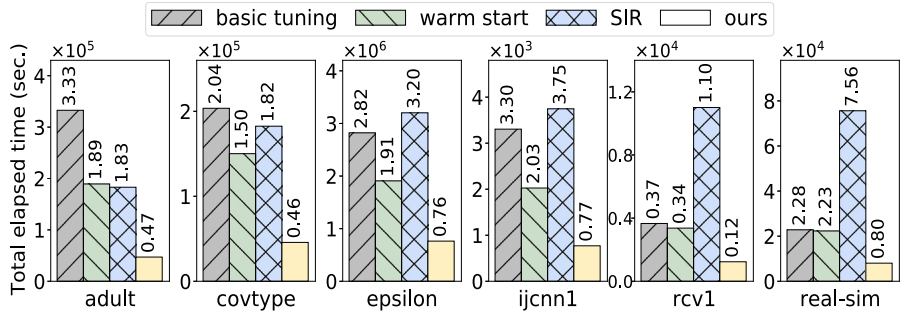


Fig. 2. Total elapsed time of grid search.

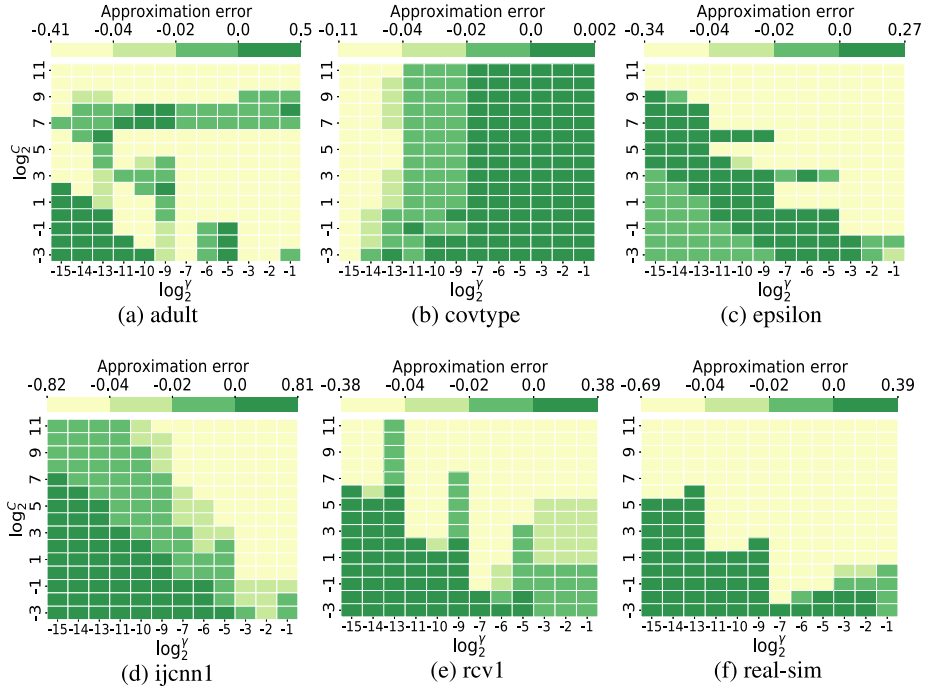


Fig. 3. Approximation error on the test accuracy of our method with each $\{C, \sigma\}$ pair.

Table 2
Averaged predictive accuracy in grid search.

data set	averaged training accuracy (%)			averaged test accuracy (%)		
	basic/SIR/WS	ours	error	basic/SIR/WS	ours	error
adult	81.77	75.83	−5.94	78.73	74.56	−4.17
covtype	97.36	96.25	−1.10	97.30	96.18	−1.12
epsilon	79.33	70.28	−9.05	74.21	68.08	−6.13
ijcnn1	86.63	82.30	−4.33	86.54	81.97	−4.57
rcv1	87.44	83.07	−4.37	85.44	81.27	−4.17
real-sim	86.54	67.67	−18.87	86.43	67.61	−18.83

ranges of hyper-parameters (e.g., when the value the regularization constant is large), the optimal solutions and their objective values vary widely from one hyper-parameter to another. As a result, the pre-training-based model may not well approximate the diverse optimal models of the next hyper-parameters. A smaller calibration parameter will help mitigate this degradation in accuracy.

6.5. Quality of our inferred models

As our method infers the model for a new kernel parameter based on the pre-trained model instead of training, we show the detailed accuracy and objective values to confirm the high quality of our inferred models.

6.5.1. Detailed accuracy comparison

We choose the regularization constant $C = 2^{-3}$ and the kernel parameter γ which is ranged from $\{2^{-5}, 2^{-6}, 2^{-7}\}$ as examples to present the detailed accuracy and approximation error produced by our method. Our method infers the models for the three γ values based on the pre-trained model where γ is equal to 2^{-4}. The results are presented in Table 3. We do not show the results of the warm start baseline in this section as it is proposed for regularization constant tuning.

According to Table 3, the approximation error on test accuracy is quite small, which indicates the high quality of our pre-training-based models. It is remarkable that our method even produces better accuracy than the baselines for *epsilon*, *rcv1* and *real-sim* problems. The test accuracy achieved on *rcv1* is improved by 32.65% at most with

Table 3
Comparison on predictive accuracy ($C = 2^{-3}$).

data set	γ	training accuracy (%)			test accuracy (%)		
		basic/SIR	ours	error	basic/SIR	ours	error
adult	2^{-4}	83.92	83.92	0.00	84.29	84.29	0.00
	2^{-5}	83.87	83.80	0.07	84.36	84.61	**0.25**
	2^{-6}	83.76	82.03	-1.73	84.34	82.54	**-1.79**
	2^{-7}	83.54	76.55	-7.00	84.23	77.17	-7.06
covtype	2^{-4}	86.06	86.06	0.00	85.85	85.85	**0.00**
	2^{-5}	86.06	86.06	0.00	85.85	85.85	**0.00**
	2^{-6}	86.06	86.06	0.00	85.85	85.85	**0.00**
	2^{-7}	86.06	86.06	0.00	85.85	85.85	**0.00**
epsilon	2^{-4}	72.63	72.63	0.00	71.58	71.60	0.02
	2^{-5}	56.90	71.08	14.18	55.10	70.02	**14.92**
	2^{-6}	50.42	67.04	16.62	49.70	64.98	**15.28**
	2^{-7}	50.42	59.85	9.43	49.70	58.50	**8.80**
ijcnn1	2^{-4}	90.64	90.64	0.00	90.58	90.58	0.00
	2^{-5}	90.29	90.29	0.00	90.50	90.50	**0.00**
	2^{-6}	90.29	90.29	0.00	90.50	90.50	**0.00**
	2^{-7}	90.29	90.29	0.00	90.50	90.50	**0.00**
rcv1	2^{-4}	94.56	94.56	0.00	93.80	93.80	0.00
	2^{-5}	93.40	94.35	0.95	92.77	93.67	**0.90**
	2^{-6}	61.18	93.25	32.07	59.65	92.30	**32.65**
	2^{-7}	51.83	86.40	34.57	52.47	84.85	**32.38**
real-sim	2^{-4}	90.28	90.28	0.00	90.04	90.04	0.00
	2^{-5}	84.11	87.68	3.57	83.81	87.41	**3.61**
	2^{-6}	74.78	82.09	7.31	74.56	81.73	**7.16**
	2^{-7}	69.65	73.07	3.42	69.62	72.89	**3.27**

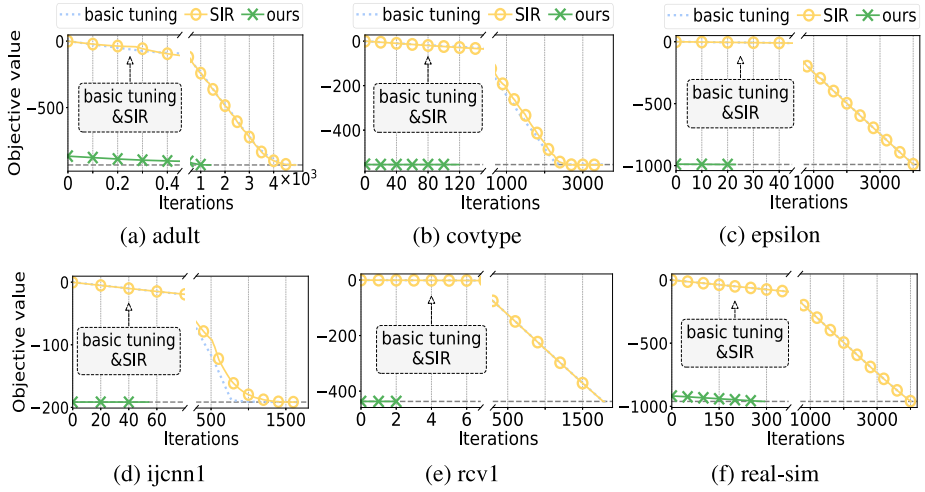


Fig. 4. Trajectories of objective value decreasing with $\{C = 2^{-3}, \gamma = 2^{-7}\}$.

our method. The basic method turns out to underfit the *rcv1* problem with a training accuracy of around 50%. The underfitting behavior results from the basic method failing to converge as shown in Fig. 4(e). On the contrary, our method infers a solution that is close to the optimal solution and thus maintains the performance close to the optimal. Similar behaviors can be observed on *epsilon* and *real-sim*.

6.5.2. *Variation of the objective in model training*

We compare the objective values of the solutions inferred using our method with the solutions trained using the baselines. We take the hyper-parameters $\{C = 2^{-3}, \gamma = 2^{-7}\}$ as an example and show the trajectories of objective value decreasing in Fig. 4. Our method infers the model with γ equal to 2^{-7} based on the pre-trained model where γ is equal to 2^{-4}. In order to observe the difference between our inferred objective and the optimal objective, after the new models were inferred, we further applied the SMO solver [38] to fine-tune the inferred model until optimal.

As depicted in Fig. 4, the objective value of our inferred model (i.e., the first cross marker on the curve of our method) is close to the optimal one. When we further fine-tune the inferred model, the training takes much fewer iterations to meet the optimality condition, compared with basic tuning or SIR. For example, in Fig. 4(d), the number of iterations of our method is about 30 times fewer than the baselines. Thus, the model inferred based on the pre-trained model can approximate the optimal model extremely well. The accuracy and objective curves with other hyper-parameters can be found in D which are also consistent with our findings.

7. Conclusion

In this work, we have proposed a novel method that leverages the pre-trained model to infer new kernel machines. We have provided the theoretical foundation to guarantee that the model inferred based on pre-trained models is of high quality and close to the optimal one. In our experiments, the gap between the objective value of the pre-training-based model and that of the optimal one is extremely close, which further verifies the correctness of our theorems. The competitive predictive accuracy indicates the model quickly obtained from the pre-trained model is as good as those trained from scratch. Moreover, the results have also confirmed that our method performs an order of magnitude faster than the existing approach. Our findings in this paper

can encourage more researchers to investigate pre-training techniques on more traditional machine learning algorithms.

In the future, we plan to integrate our method into existing machine learning pipelines to facilitate its practical adoption by domain practitioners. Concurrently, we will further advance the theoretical foundations of leveraging pre-training in kernel machines. A key focus will be deriving a more rigorous error bound that accommodates specific hyper-parameter variations. We will also expand our evaluation to a wider range of real-world applications including computer vision and natural language processing tasks to further demonstrate the broader applicability of our method.

CRediT authorship contribution statement

Yawen Chen: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Zeyi Wen:** Supervision, Methodology, Investigation, Formal analysis. **Jian Chen:** Writing – review & editing, Supervision, Resources, Conceptualization. **Jin Huang:** Writing – review & editing, Supervision, Resources.

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.

Acknowledgments

This work was supported by the China Postdoctoral Science Foundation (Grant No. 2024M760785), the National Natural Science Foundation of China (Grant No. 62376099), the Natural Science Foundation of Guangdong Province (Grant No. 2024A1515010989), and the Key R&D and Promotion Projects of Henan Province (Grant No. 252102210047).

Appendix A. Proof of Theorem 1

Proof. In order to solve the optimization Problem (1), we first transform the Problem (1) into the following form with Lagrangian multipliers.

$$L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\mu}) = \frac{1}{n}\sum_{i=1}^{n} l(\sum_{j=1}^{n}\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j), y_i) + \frac{\lambda}{2}\|\sum_{j=1}^{n}\alpha_j\phi(\boldsymbol{x}_j)\|^2 + \boldsymbol{\beta}^T\boldsymbol{h}(X,\boldsymbol{\alpha},\Theta) + \boldsymbol{\mu}^T\boldsymbol{g}(X,\boldsymbol{\alpha},\Theta). \tag{A.1}$$

The transformation is inspired by the proof of Lemma 4 in the paper [40]. The ith element of vector $\boldsymbol{\beta}\in\mathbb{R}^{n_h}$ and vector $\boldsymbol{\mu}\in\mathbb{R}^{n_g}$ are the Lagrangian multipliers β_i and μ_i respectively. With the positive-definite kernels defined in Theorem 1, the hyper-parameter set Θ is equivalent to $\{\lambda,\sigma\}$. The n_h dimensional vector function $\boldsymbol{h}(X,\boldsymbol{\alpha},\Theta)$ treats the constraint $h_i(X,\boldsymbol{\alpha},\Theta)$ as its ith dimension; similarly, the n_g dimensional vector function $\boldsymbol{g}(X,\boldsymbol{\alpha},\Theta)$ is formed by all the inequality constraints and its jth dimension is $g_j(X,\boldsymbol{\alpha},\Theta)$.

Then we compute the Karush-Kuhn–Tucker (KKT) conditions [40] for the Problem (A.1) as follows.

$$\begin{aligned}\frac{\partial L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\mu})}{\partial\alpha_p} =& \frac{1}{n}\sum_{i=1}^{n}\nabla_{\alpha_p} l(\sum_{j=1}^{n}\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j), y_i) + \lambda\boldsymbol{\alpha}^T K_p(\sigma) \\ &+ \boldsymbol{\beta}^T\frac{\partial\boldsymbol{h}(X,\boldsymbol{\alpha},\Theta)}{\partial\alpha_p} + \boldsymbol{\mu}^T\frac{\partial\boldsymbol{g}(X,\boldsymbol{\alpha},\Theta)}{\partial\alpha_p} = 0, \\ \text{subject to}\quad & h_i(X,\boldsymbol{\alpha},\Theta)=0,\ \forall i\in\{1,\dots,n_h\}, \\ & \mu_j g_j(X,\boldsymbol{\alpha},\Theta)=0,\ g_j(X,\boldsymbol{\alpha},\Theta)\le 0,\ \mu_j\ge 0,\ \forall j\in\{1,\dots,n_g\}.\end{aligned} \tag{A.2}$$

Since the matrix X of instances is the same for all the constraints, we omit the matrix X in all the constraints for the rest of the paper. We use $h_i(\boldsymbol{\alpha},\Theta)$ and $g_j(\boldsymbol{\alpha},\Theta)$ as the shorthand for $h_i(X,\boldsymbol{\alpha},\Theta)$ and $g_j(X,\boldsymbol{\alpha},\Theta)$, respectively. Then the KKT conditions [40] in Eq. (A.2) can be rewritten as follows.

$$\begin{aligned}\frac{\partial L(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\mu},\boldsymbol{\delta})}{\partial\alpha_p} =& \widetilde{K}_p(\sigma)^T\boldsymbol{e} + \lambda\boldsymbol{\alpha}^T K_p(\sigma) + \boldsymbol{\beta}^T\frac{\partial\boldsymbol{h}(\boldsymbol{\alpha},\Theta)}{\partial\alpha_p} + \boldsymbol{\mu}^T\frac{\partial\boldsymbol{g}(\boldsymbol{\alpha},\Theta)}{\partial\alpha_p} = 0, \\ \text{subject to}\quad & h_i(\boldsymbol{\alpha},\Theta)=0,\ \forall i\in\{1,\dots,n_h\}, \\ & \mu_j g_j(\boldsymbol{\alpha},\Theta)=0,\ g_j(\boldsymbol{\alpha},\Theta)\le 0,\ \mu_j\ge 0,\ \forall j\in\{1,\dots,n_g\},\end{aligned} \tag{A.3}$$

where $\boldsymbol{e}$ is an n-dimension vector with all the elements equal to 1. Let $K_p(\sigma)$ and $\widetilde{K}_p(\sigma)$ denote the pth column in the kernel matrix $K(\sigma)$ and matrix $\widetilde{K}(\sigma)$, respectively. The elements in the ith row and jth column of matrices $K(\sigma)$ and $\widetilde{K}(\sigma)$ are defined separately as $K_{ij}(\sigma)=k_{ij}(\sigma)$, and

$$\widetilde{K}_{ij}(\sigma)=\begin{cases}-k_{ij}(\sigma), & y_i > f(\boldsymbol{x}_i);\\ k_{ij}(\sigma), & y_i < f(\boldsymbol{x}_i);\\ 0, & y_i = f(\boldsymbol{x}_i).\end{cases}$$

With the above definition of $\widetilde{K}_{ij}(\sigma)$, L_1 loss defined in Eq. (2) is continuously differentiable and convex over variable α.

Suppose when the hyper-parameter set $\Theta'=\{\lambda,\sigma'\}$ is used where $\sigma'=\frac{\sigma_0}{\Delta}$ and $\Delta>1$, the optimal solutions to Problem (A.1) are $\boldsymbol{\alpha}'^*$, $\boldsymbol{\beta}'^*$ and $\boldsymbol{\mu}'^*$. For all i in $\{1,\dots,n\}$, the optimums $\beta_i'^*$ and $\mu_i'^*$ satisfy the following inequalities.

$$\beta_{min}\le\beta_i'^*\le\beta_{max},\ \ \mu_i'^*\le\mu_{max},$$

where β_{min}, β_{max} and μ_{max} are constants. Substituting the optimums (i.e., $\boldsymbol{\alpha}'^*$, $\boldsymbol{\beta}'^*$ and $\boldsymbol{\mu}'^*$) of Problem (A.1) into Eq. (A.3), and adding $-\lambda\boldsymbol{c}^T K_p(\sigma')$ on both sides, we can obtain

$$\lambda(\boldsymbol{\alpha}'^{*T}-\boldsymbol{c}^T)K_p(\sigma') = B_p-\Omega_p,\ \forall p\in\{1,\dots,n\},$$

$$B_p=-\widetilde{K}_p(\sigma')^T\boldsymbol{e}-\lambda\boldsymbol{c}^T K_p(\sigma'),\ \ \Omega_p=\boldsymbol{\beta}'^{*T}\frac{\partial\boldsymbol{h}(\boldsymbol{\alpha}'^*,\Theta')}{\partial\alpha_p}+\boldsymbol{\mu}'^{*T}\frac{\partial\boldsymbol{g}(\boldsymbol{\alpha}'^*,\Theta')}{\partial\alpha_p},$$

where $\boldsymbol{c}=[c(y_1)\dots c(y_n)]^T$. The above equations can be integrated into one formula as

$$\lambda(\boldsymbol{\alpha}'^{*T}-\boldsymbol{c}^T) = (B-\Omega)K(\sigma')^{-1} = -\boldsymbol{e}^T U-\lambda\boldsymbol{c}^T-\Omega K(\sigma')^{-1}, \tag{A.4}$$

where $B=[B_1\ B_2\ \dots\ B_n]$ and $\Omega=[\Omega_1\ \Omega_2\ \dots\ \Omega_n]$. The matrix U is a diagonal matrix. The element u_{pp} on the diagonal of matrix U equals -1 when y_i is larger than $f(\boldsymbol{x}_i)$ or equals 1 when y_i is smaller than $f(\boldsymbol{x}_i)$, otherwise equals 0.

Next, we take square on both sides of Eq. (A.4) and can obtain

$$\begin{aligned}&\lambda^2(\alpha_p'^*-c(y_p))^2 = [-u_{pp}-\lambda c(y_p)-\Omega K(\sigma')_p^{-1}]^2 \\ &\le 2[u_{pp}+\lambda c(y_p)]^2+2[\Omega K(\sigma')_p^{-1}]^2 \\ &= 2[u_{pp}+\lambda c(y_p)]^2+2\left[\sum_{i=1}^{n}\Omega_i K(\sigma')_{ip}^{-1}\right]^2 \\ &\le 2[u_{pp}+\lambda c(y_p)]^2+2\left[\sum_{i=1}^{n}\Omega_i^2\right]\left[\sum_{i=1}^{n}(K(\sigma')_{ip}^{-1})^2\right].\end{aligned} \tag{A.5}$$

The last step can be derived from Cauchy–Schwarz inequality. The last term $K(\sigma')_{ip}^{-1}$ on the right side of Eq. (A.5) is the element on the ith row and pth column of inverse $K(\sigma')^{-1}$ which is a constant as the kernel function and hyper-parameter σ' are given. The term Ω_i^2 on the right side of Eq. (A.5) satisfies the following inequality.

$$\Omega_i^2\le\hat{\Omega}_i,\forall i\in\{1,\dots,n\}, \tag{A.6}$$

where $\hat{\Omega}_i=2(\max\{|\beta_{min}|,|\beta_{max}|\})^2(\frac{\partial\boldsymbol{h}(\boldsymbol{\alpha}'^*,\Theta')}{\partial\alpha_i})^2+2\mu_{max}^2(\frac{\partial\boldsymbol{g}(\boldsymbol{\alpha}'^*,\Theta')}{\partial\alpha_i})^2$. Since the constraints $h_i(\boldsymbol{\alpha}'^*,\Theta')$ and $g_j(\boldsymbol{\alpha}'^*,\Theta')$ are affine functions, we can derive that the partial derivatives $\frac{\partial\boldsymbol{h}(\boldsymbol{\alpha}'^*,\Theta')}{\partial\alpha_i}$ and $\frac{\partial\boldsymbol{g}(\boldsymbol{\alpha}'^*,\Theta')}{\partial\alpha_i}$ are constants with respect to α_i. Finally, combining Eqs. (A.5) and (A.6), we have the following inequality.

$$\begin{aligned}\lambda^2(\alpha_p'^*-c(y_p))^2 &\le 2[u_{pp}+\lambda c(y_p)]^2+2\|\hat{\boldsymbol{\Omega}}\|_2^2\ \|(K(\sigma')^{-1})_p\|_2^2 \\ &\le 2[u_{pp}+\lambda c(y_p)]^2+2(z_1\|\hat{\boldsymbol{\Omega}}\|_\infty)^2[z_2\|(K(\sigma')^{-1})_p\|_\infty]^2\end{aligned}$$

Table D.4
Comparison on model accuracy ($C = 2^6$).

data set	γ	training accuracy (%)			test accuracy (%)		
		basic/SIR	ours	error	basic/SIR	ours	error
adult	2^{-12}	84.57	84.56	0.00	85.14	85.15	0.01
	2^{-13}	84.46	84.44	-0.02	85.08	85.22	**0.14**
	2^{-14}	84.18	83.30	-0.88	84.91	84.14	**-0.77**
	2^{-15}	83.91	78.93	-4.98	84.69	79.86	-4.83
covtype	2^{-12}	100.00	100.00	0.00	100.00	100.00	0.00
	2^{-13}	100.00	96.25	-3.75	100.00	95.88	-4.12
	2^{-14}	99.99	93.70	-6.29	100.00	93.45	-6.55
	2^{-15}	99.94	89.11	-10.83	99.95	89.33	-10.62
epsilon	2^{-12}	82.45	82.45	0.00	81.10	81.10	0.00
	2^{-13}	74.17	81.76	7.59	72.96	80.52	**7.56**
	2^{-14}	58.67	79.85	21.18	57.32	78.66	**21.34**
	2^{-15}	50.34	73.97	23.63	50.78	72.64	**21.86**
ijcnn1	2^{-12}	91.34	91.34	0.00	91.05	91.05	0.00
	2^{-13}	90.63	90.29	-0.34	90.51	90.50	**-0.01**
	2^{-14}	90.30	90.29	-0.01	90.50	90.50	**0.00**
	2^{-15}	90.29	90.29	0.00	90.50	90.50	**0.00**
rcv1	2^{-12}	95.56	95.56	0.00	94.68	94.68	0.00
	2^{-13}	94.63	95.16	0.53	93.96	94.12	**0.16**
	2^{-14}	93.41	92.27	-1.14	92.81	90.54	**-2.27**
	2^{-15}	63.70	77.58	13.88	62.06	74.59	**12.53**
real-sim	2^{-12}	94.84	94.83	-0.01	94.66	94.66	0.00
	2^{-13}	91.15	95.59	4.44	91.05	95.56	**4.51**
	2^{-14}	84.82	63.51	-21.31	84.42	62.76	-21.66
	2^{-15}	75.19	30.84	-44.35	74.98	30.88	-44.10

$$= 2[u_{pp} + \lambda c(y_p)]^2 + 2z_1^2 z_2^2 (\max_{1\le i\le n} \hat{\Omega}_i)^2 [\max_{1\le i\le n} |(K(\sigma')^{-1})_{ip}|]^2$$
$$= 2[1 + \lambda |c(y_p)|]^2 + 2z_1^2 z_2^2 (\max_{1\le i\le n} \hat{\Omega}_i)^2 [\max_{1\le i\le n} |(K(\sigma')^{-1})_{ip}|]^2.$$

The second inequality is obtained based on the equivalence relation between vector norms where z_1 and z_2 are constants. Finally, we can deduce the inequality $(\alpha_p'^* - c(y_p))^2 \le r(y_p)$ as demonstrated in Theorem 1 where $r(y_p) = \frac{2}{\lambda^2}[1+\lambda|c(y_p)|]^2 + \frac{2z_1^2 z_2^2}{\lambda^2}(\max_{1\le i\le n} \hat{\Omega}_i)^2 [\max_{1\le i\le n} |(K(\sigma')^{-1})_{ip}|]^2$ and $r(y_p)$ is a constant. Hence, the proof is completed. □

Appendix B. Proof of Theorem 2

Proof. To prove Theorem 2, we rewrite the KKT conditions shown in (A.3) as follows.

$$-\widetilde{K}_i(\sigma_0)^T \boldsymbol{e} - \boldsymbol{\beta}^{*T} \frac{\partial \boldsymbol{h}(\boldsymbol{\alpha}^*, \Theta_0)}{\partial \alpha_i} - \boldsymbol{\mu}^{*T} \frac{\partial \boldsymbol{g}(\boldsymbol{\alpha}^*, \Theta_0)}{\partial \alpha_i} = \lambda_0 \boldsymbol{\alpha}^{*T} K_i(\sigma_0), \ \forall i \in \{1, \dots, n\},$$

where $\Theta_0 = \{\lambda_0, \sigma_0\}$. Then all the equations above can be written into one linear system as shown below.

$$\lambda_0 \boldsymbol{\alpha}^{*T} K(\sigma_0) = \boldsymbol{s}^T, \tag{B.1}$$

where $\boldsymbol{s} \in \mathbb{R}^n$ and the ith element in $\boldsymbol{s}$ is denoted by $s_i = [-\widetilde{K}_i(\sigma_0)^T \boldsymbol{e} - \boldsymbol{\beta}^{*T} \frac{\partial \boldsymbol{h}(\boldsymbol{\alpha}^*, \Theta_0)}{\partial \alpha_i} - \boldsymbol{\mu}^{*T} \frac{\partial \boldsymbol{g}(\boldsymbol{\alpha}^*, \Theta_0)}{\partial \alpha_i}]$. The right side of the equation is independent of λ and $\boldsymbol{\alpha}$. According to Eq. (B.1), we can obtain the relationship between the optimal $\boldsymbol{\alpha}$ and the regularization constant as in Theorem 2. Hence, the proof is completed. □

Appendix C. Proof of Theorem 1 on Non-linear SVMs

Proof. We prove that Theorem 1 holds for non-linear SVMs with the dual objective defined in Eq. (7). The hinge loss used in the SVM can be treated as a variant of L_1 loss with the loss value equal to 0 when $1 - y_i(\sum_j \alpha_j k_{ij} + b)$ is less than zero. As the equality and inequality constraints are affine functions and are convex, we have the KKT conditions of the dual objective in SVMs as follows.

$$\nabla_{\alpha_i} W(\boldsymbol{\alpha}) + \beta_i y_i = \delta_i - \mu_i, \tag{C.1}$$

$$\text{subject to} \quad \delta_i \alpha_i = 0, \ \mu_i(\alpha_i - C) = 0, \ \delta_i \ge 0, \ \mu_i \ge 0, \ \forall i \in \{1, \dots, n\},$$

where β_i, δ_i and μ_i are the Lagrange multipliers for the constraints. The weight $\boldsymbol{\alpha}$ is an n-dimensional vector and α_i denotes the weight of instance $\boldsymbol{x}_i$. Suppose when Problem (C.1) is solved with the hyper-parameter set Θ' where $\Theta' = \{C, \sigma'\}$, the optimal solutions to Problem (C.1) are $\boldsymbol{\alpha}'^*$, $\boldsymbol{\beta}'^*$, $\boldsymbol{\delta}'^*$ and $\boldsymbol{\mu}'^*$. Then for each in-bound support vector $\boldsymbol{x}_i$, where $i \in I'_{inb} = \{i | 0 < \alpha_i'^* < C\}$, we have $\delta_i'^* = \mu_i'^* = 0$ and $\beta_i'^*$ satisfies $\beta_{min} \le \beta_i'^* \le \beta_{max}$. Based on these properties of $\delta_i'^*$, $\mu_i'^*$ and $\beta_i'^*$, we can obtain the following KKT conditions.

$$\boldsymbol{\alpha}'^{*T} Q(\sigma')_i + \beta_i'^* y_i - 1 = 0, \quad \forall i \in I'_{inb}, \tag{C.2}$$

where $Q_i(\cdot)$ is the ith column in $Q(\cdot)$. We follow the ideas in A and rewrite Eq. (C.2) as the inequality below where z is a constant.

$$(\alpha_p'^* - c_{inb}(y_p))^2 = \left[\sum_{i\in I'_{inb}} (1 - \beta_i'^* y_i - \boldsymbol{c}_{inb} \bar{Q}(\sigma')_i) \bar{Q}(\sigma')^{-1}_{ip}\right]^2$$
$$\le \left[\sum_{i\in I'_{inb}} (1 - \beta_i'^* y_i - \boldsymbol{c}_{inb} \bar{Q}(\sigma')_i)^2\right] \left[\sum_{i\in I'_{inb}} (\bar{Q}(\sigma')^{-1}_{ip})^2\right]$$
$$\le 2z^2 [\max_{i\in I'_{inb}} |(Q(\sigma')^{-1})_{ip}|]^2 \left[\sum_{i\in I'_{inb}} \hat{\beta}^2 y_i^2 + (1 - \boldsymbol{c}_{inb} \bar{Q}(\sigma')_i)^2\right] = r_{inb}(y_p),$$

where $\hat{\beta} = \max\{|\beta_{min}|, |\beta_{max}|\}$. The matrix $\bar{Q} \in \mathbb{R}^{n \times n_{inb}}$ consists of every column Q_i from Q where $i \in I'_{inb}$. If $\bar{Q}$ is not a square matrix, we use the pseudo inverse to compute $\bar{Q}^{-1}$. Thus we have that the optimal solution $\alpha_i'^*$ converges to the centroid $c_{inb}(y_i)$ within a constant range $r_{inb}(y_i)$ for each i in I'_{inb} as shown in Eq. (8), where $c_{inb}(y_i) = \frac{\sum_{j\in I_{inb}(y_i)} \alpha_j^*}{|I_{inb}(y_i)|} = \hat{\alpha}_i'^*$.

Appendix D. Experiments on Kernel Parameter tuning

Detailed accuracy comparison. In addition to Table 3, we present the model accuracy for a larger regularization constant where $C = 2^6$ and smaller kernel parameters where γ is ranged from $\{2^{-13}, 2^{-14}, 2^{-15}\}$. Our method infers models for the three kernel parameters based on the pre-trained model with $\{C = 2^6, \gamma = 2^{-12}\}$. The model accuracy is listed in Table D.4. The accuracy achieved by our method is similar

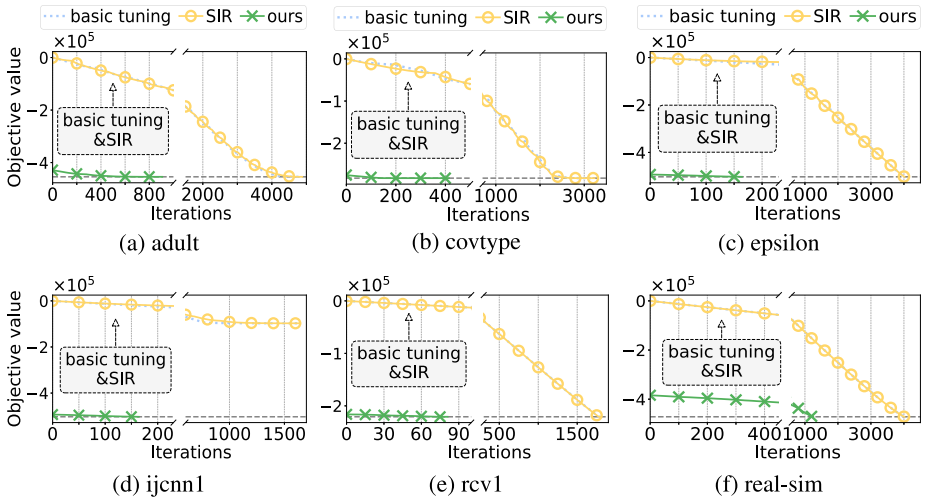


Fig. D.5. Trajectories of objective value decreasing with $\{C = 2^6, \gamma = 2^{-15}\}$.

to or better than that of the basic method on most tested data sets. There is a notable degradation on the predictive performance of our method on *real-sim*, because the inferred model is more different from the optimal when the optimal solutions distinguish widely from one hyper-parameter to another as explained in Section 6.4. We can choose a smaller calibration parameter to alleviate the accuracy degradation.

Variation of objective in model training. We show the trajectories of objective decreasing in the model training when the hyper-parameter is set as $\{C = 2^6, \gamma = 2^{-15}\}$ in Fig. D.5. Our method infers the model for $\{C = 2^6, \gamma = 2^{-15}\}$ and then the inferred model is trained to optimal. As shown in Fig. D.5, the objective value of our inferred model is close to the optimal except for *real-sim*. On the *real-sim* data set, the objective value of the solution inferred by our method is slightly far from the optimal one compared with the results on the other tested data sets. The objective curves for the rest tested hyper-parameters are similar to Fig. 4 and Fig. D.5 which we do not show here for the limited length of paper.

Data availability

I have shared the link to my data in the manuscript.

References

[1] D. Samuel, BERTs are generative in-context learners, in: Proceedings of the Advances in Neural Information Processing Systems, vol. 37, 2024, pp. 2558–2589.
[2] E. Fini, M. Shukor, X. Li, P. Dufter, M. Klein, D. Haldimann, S. Aitharaju, V.G.T. da Costa, L. Béthune, Z. Gan, et al., Multimodal autoregressive pre-training of large vision encoders, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2025.
[3] Z. Wen, Z. Zhou, H. Liu, B. He, X. Li, J. Chen, Enhancing SVMs with problem context aware pipeline, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, 2021, pp. 1821–1829.
[4] M. Belkin, S. Ma, S. Mandal, To understand deep learning we need to understand kernel learning, in: Proceedings of the International Conference on Machine Learning, 2018, pp. 541–549.
[5] Z. Wen, B. Li, K. Ramamohanarao, J. Chen, Y. Chen, R. Zhang, Improving efficiency of SVM k-fold cross-validation by alpha seeding, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2017, pp. 2768–2774.
[6] C. Tsai, C. Lin, C. Lin, Incremental and decremental training for linear classification, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, 2014, pp. 343–352.
[7] L. Gunter, J. Zhu, Computing the solution path for the regularized support vector regression, in: Proceedings of the Advances in Neural Information Processing Systems, vol. 18, 2005.
[8] G. Wang, D. Yeung, F.H. Lochovsky, A kernel path algorithm for support vector machines, in: Proceedings of the International Conference on Machine Learning, 2007, pp. 951–958.
[9] T. Hastie, S. Rosset, R. Tibshirani, J. Zhu, The entire regularization path for the support vector machine, J. Mach. Learn. Res. 5 (10) (2004) 1391–1415.
[10] Y. Gu, Z. Song, L. Zhang, Faster algorithms for structured linear and kernel support vector machines, in: Proceedings of the International Conference on Learning Representations, 2025.
[11] M. Tukan, C. Baykal, D. Feldman, D. Rus, On coresets for support vector machines, Theoret. Comput. Sci. 890 (2021) 171–191.
[12] V. Aceña, I. Martín de Diego, R. R. Fernández, J. M. Moguerza, Support subsets estimation for support vector machines retraining, Pattern Recognit. 134 (2023) 109117.
[13] X. Wang, Y. Jin, S. Schmitt, M. Olhofer, Recent advances in Bayesian optimization, ACM Comput. Surv. 55 (13s) (2023) 1–36.
[14] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F.L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., GPT-4, Technical Report, 2024, arXiv:2303.08774.
[15] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al., DeepSeek-V3, Technical Report, 2025, arXiv:2412.19437.
[16] T. Tommasi, B. Caputo, The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories, in: Proceedings of the British Machine Vision Conference, 2009.
[17] M. Karasuyama, I. Takeuchi, Multiple incremental decremental learning of support vector machines, IEEE Trans. Neural Netw. 21 (7) (2010) 1048–1059.
[18] H. Gâlmeanu, R. Andonie, Weighted incremental–decremental support vector machines for concept drift with shifting window, Neural Netw. 152 (2022) 528–541.
[19] H. Chen, Y. Yu, Y. Jia, B. Gu, Incremental learning for transductive support vector machine, Pattern Recognit. 133 (2023) 108982.
[20] D. Xu, M. Jiang, W. Hu, S. Li, R. Pan, G.G. Yen, An online prediction approach based on incremental support vector machine for dynamic multiobjective optimization, IEEE Trans. Evol. Comput. 26 (4) (2021) 690–703.
[21] L. Jian, S. Shen, J. Li, X. Liang, L. Li, Budget online learning algorithm for least squares SVM, IEEE Trans. Neural Networks Learn. Syst. 28 (9) (2016) 2076–2087.
[22] B.Y. Chu, C.H. Ho, C.H. Tsai, C.Y. Lin, C.J. Lin, Warm start for parameter selection of linear classifiers, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, 2015, pp. 149–158.
[23] B. Gu, V.S. Sheng, K.Y. Tay, W. Romano, S. Li, Cross validation through two-dimensional solution surface for cost-sensitive SVM, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2017) 1103–1121.
[24] B. Gu, V.S. Sheng, A robust regularization path algorithm for ν-support vector classification, IEEE Trans. Neural Netw. Learn. Syst. 28 (5) (2016) 1241–1248.
[25] Z. Zhai, B. Gu, C. Deng, H. Huang, Global model selection via solution paths for robust support vector machine, IEEE Trans. Pattern Anal. Mach. Intell. 47 (3) (2025) 1331–1347.
[26] S. Suzumura, K. Ogawa, M. Sugiyama, M. Karasuyama, I. Takeuchi, Homotopy continuation approaches for robust SV classification and regression, Mach. Learn. 106 (2017) 1009–1038.
[27] B. Gu, A regularization path algorithm for support vector ordinal regression, Neural Netw. 98 (2018) 114–121.
[28] J. Giesen, S. Laue, P. Wieschollek, Robust and efficient kernel hyperparameter paths with guarantees, in: Proceedings of the International Conference on Machine Learning, 2014, pp. 1296–1304.
[29] Y. Chen, Z. Wen, B. He, J. Chen, Efficient decomposition selection for multi-class classification, IEEE Trans. Knowl. Data Eng. 35 (4) (2021) 3751–3764.

[30] H. Lee, V. Patrangenaru, Extrinsic kernel ridge regression classifier for Kendallâs planar shape space, Pattern Recognit. 161 (2025) 111297.
[31] L. Liu, H. Sun, F. Li, A Lie group kernel learning method for medical image classification, Pattern Recognit. 142 (2023) 109735.
[32] A. Geifman, A. Yadav, Y. Kasten, M. Galun, D. Jacobs, B. Ronen, On the similarity between the laplace and neural tangent kernels, in: Proceedings of the Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 1451–1461.
[33] S. Pesme, N. Flammarion, Online robust regression via SGD on the l1 loss, in: Proceedings of the Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 2540–2552.
[34] L. Yu, S. Li, S. Liu, Fast support vector machine training via three-term conjugate-like SMO algorithm, Pattern Recognit. 139 (2023) 109478.
[35] C. Chang, C. Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 1–27.
[36] G. Beugnot, J. Mairal, A. Rudi, On the benefits of large learning rates for kernel methods, in: Proceedings of the Conference on Learning Theory, 2022, pp. 254–282.
[37] Z. Wen, J. Shi, Q. Li, B. He, J. Chen, ThunderSVM: A fast SVM library on GPUs and CPUs, J. Mach. Learn. Res. 19 (2018) 1–5.
[38] P. Chen, R. Fan, C. Lin, A study on SMO-type decomposition methods for support vector machines, IEEE Trans. Neural Netw. 17 (4) (2024) 893–908.
[39] G. Galvan, M. Lapucci, C. Lin, M. Sciandrone, A two-level decomposition framework exploiting first and second order information for SVM training problems, J. Mach. Learn. Res. 22 (23) (2021) 1–38.
[40] S.S. Keerthi, C. Lin, Asymptotic behaviors of support vector machines with Gaussian kernel, Neural Comput. 15 (7) (2003) 1667–1689.

Yawen Chen is currently a Lecturer in the School of Artificial Intelligence at Henan University. She received her Ph.D. from the School of Software Engineering at South China University of Technology. Her research interests include kernel methods and automated machine learning.

Zeyi Wen is an Assistant Professor at Hong Kong University of Science and Technology, Guangzhou (HKUST-GZ). Before joining HKUST-GZ, he was a Lecturer at The University of Western Australia, Research Fellow at National University of Singapore and The University of Melbourne after Ph.D. completion at The University of Melbourne. His research interests include ML systems and high-performance computing.

Jian Chen is currently a Professor of the School of Software Engineering at South China University of Technology where she started as an Assistant Professor in 2005. She received her B.S. and Ph.D. degrees, both in Computer Science, from Sun Yat-Sen University, China, in 2000 and 2005 respectively. Her research interests can be summarized as developing effective and efficient data analysis techniques for complex data and the related applications.

Jin Huang is currently an Associate Professor at South China Normal University. He received his M.E. and Ph.D. degrees in Computer Science from Sun Yat-Sen University, China, in 2004 and 2010, respectively. His current research interests include social network analysis and graph mining.