mathematics

Review

Exploring Kernel Machines and Support Vector Machines: Principles, Techniques, and Future Directions

Ke-Lin Du [1], Bingchun Jiang [1,*], Jiabin Lu [2], Jingyu Hua [3] and M. N. S. Swamy [4]

[1] School of Mechanical and Electrical Engineering, Guangdong University of Science and Technology, Dongguan 523668, China; kldu@gdust.edu.cn
[2] Faculty of Electromechanical Engineering, Guangdong University of Technology, Guangzhou 510006, China; lujiabin@gdut.edu.cn
[3] School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China; eehjy@mail.zjgsu.edu.cn
[4] Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada; swamy@ece.concordia.ca
* Correspondence: jiangbingchun@gdust.edu.cn

Abstract: The kernel method is a tool that converts data to a kernel space where operation can be performed. When converted to a high-dimensional feature space by using kernel functions, the data samples are more likely to be linearly separable. Traditional machine learning methods can be extended to the kernel space, such as the radial basis function (RBF) network. As a kernel-based method, support vector machine (SVM) is one of the most popular nonparametric classification methods, and is optimal in terms of computational learning theory. Based on statistical learning theory and the maximum margin principle, SVM attempts to determine an optimal hyperplane by addressing a quadratic programming (QP) problem. Using Vapnik–Chervonenkis dimension theory, SVM maximizes generalization performance by finding the widest classification margin within the feature space. In this paper, kernel machines and SVMs are systematically introduced. We first describe how to turn classical methods into kernel machines, and then give a literature review of existing kernel machines. We then introduce the SVM model, its principles, and various SVM training methods for classification, clustering, and regression. Related topics, including optimizing model architecture, are also discussed. We conclude by outlining future directions for kernel machines and SVMs. This article functions both as a state-of-the-art survey and a tutorial.

Keywords: support vector machine; kernel machine; kernel method; support vector regression; support vector clustering

MSC: 68Q32; 68T99; 90-10

check for updates

Citation: Du, K.-L.; Jiang, B.; Lu, J.; Hua, J.; Swamy, M.N.S. Exploring Kernel Machines and Support Vector Machines: Principles, Techniques, and Future Directions. *Mathematics* **2024**, *12*, 3935. https://doi.org/10.3390/math12243935

Academic Editor: Jonathan Blackledge

Received: 12 November 2024
Revised: 29 November 2024
Accepted: 11 December 2024
Published: 13 December 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Kernel method, also referred to as kernel machines and first introduced in [1], involves projecting the training data from a lower-dimensional space to a higher-dimensional feature or kernel space through nonlinear kernel functions (see Figure 1). According to the Cover theorem, data points are more likely to be linearly separable when they are nonlinearly mapped to a higher-dimensional space. Nonlinear kernel functions effectively mitigate the curse of dimensionality.

The kernel method serves as a versatile nonparametric modeling approach that is extensively applied in both machine learning and data analysis. Notable examples of kernel methods include the kernel density estimator (often known as the Parzen window estimator), the RBF network, and SVM.

Kernel method implicitly conducts a nonlinear transformation of the input data into a high-dimensional feature space by substituting the inner product with a suitable positive definite function. This kernel trick, initially introduced with SVM, has been utilized to develop nonlinear extensions of various classical linear statistical and machine learning models. Furthermore, kernel method offers a graceful framework for deriving nonlinear learning algorithms based on their linear counterparts.

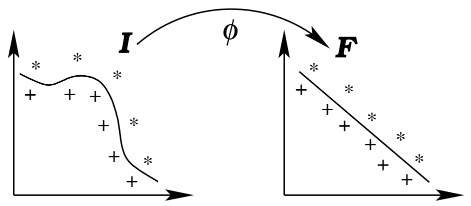

Figure 1. Transforming with kernel methods: maps input space to feature space, achieving linear separation in the feature space. "+" and "*" correspond to points in different classes.

According to Mercer's theorem [2], a kernel function $k(\mathbf{x}, \mathbf{x}')$ is classified as a Mercer kernel if it is continuous, symmetric, and positive definite. Consequently, the corresponding kernel matrix is semidefinite and consists solely of positive eigenvalues. Moreover, any nonnegative linear combination of Mercer kernels, as well as product of Mercer kernels, will produce Mercer kernels. Any symmetric function $k(\mathbf{x}_i, \mathbf{x}_j)$ that satisfies the conditions of Mercer's theorem can serve as a kernel function [3].

Reproducing kernel Hilbert spaces (RKHSs) are frequently employed as hypothesis spaces in the learning theory. When integrated with regularization techniques, RKHSs enhance the generalization capabilities of learned models and ensure that the solutions exhibit smoothness properties [4].

The kernels-as-features approach [5] offers another method for nonlinear feature generation, treating the kernel function directly as features. For L data points in the input space, $\mathbf{x}_1, \ldots, \mathbf{x}_L \in \mathcal{X}$, each $\mathbf{x} \in \mathcal{X}$ is mapped into an L-dimensional φ-space, defined as $\varphi(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_L))^T$. In this φ-space, various algorithms can be applied.

Both the kernel trick and the kernels-as-features concept generate nonlinear feature spaces in which linear algorithms can operate. An equivalence between these two kernel methods has been established for feature extraction algorithms, including principal component analysis (PCA) and linear discriminant analysis (LDA) [6], as well as for PCA, LDA, and canonical correlation analysis (CCA) [7]. Additionally, wavelet-like reproducing kernels are considered refinable kernels, enabling multiresolution analysis within RKHSs [8].

Following the success of SVM, various linear learning techniques have been extended into their respective kernel forms, including kernel PCA [9,10], kernel LDA [11,12], kernel clustering [13], kernel independent component analysis (ICA) [14], kernel blind source separation (BSS) [15], kernel CCA [16], and minimax probability machine [17].

Kernel methods have been expanded beyond RKHSs to incorporate Krein spaces [18] and Banach spaces [19].

SVM [20,21] can be conceptualized as a three-layer feedforward network. It employs the principle of structural risk minimization (SRM), which attempts to achieve the minimal upper bound on generalization error. This error consists of the training error plus a confidence interval that is contingent upon the Vapnik–Chervonenkis (VC) dimension. Notably, the generalization errors of SVMs are not influenced by the dimensionality of the input data, instead by the margin used to separate the data. Rather than solely minimizing the training error, SVM focuses on minimizing the upper bound of the generalization error while maximizing the margin that exists between the separating hyperplane and the training samples.

SVM aims at minimizing the VC dimension by identifying the optimal hyperplane that effectively distinguishes between classes while maximizing the margin. This margin is quantified as the distance between the closest points from each class to the hyperplane. SVM incorporates a general linear learning algorithm alongside a specific kernel designed to compute the inner product of input data points within a feature space. Within this space, the projections of training instances can always be separated linearly. Notably, studies have shown that the hippocampus, a crucial region for learning and memory in the brain, exhibit a pattern separation function that is analogous to that of SVM [22].

Much like biological systems, SVMs focus on borderline cases and outliers while disregarding typical examples. Bio-SVM [23] represents a biologically feasible adaptation of SVM. This model features an unstable associative memory that alternates between support vectors and engages with a feedforward classification pathway. Phenomena such as emotion-based learning, the process of forgetting trivial information, sleep, and brain oscillations align with the principles of the bio-SVM model. Additionally, a mapping to functions within the olfactory system is proposed within this framework.

SVMs serve as universal approximators for a variety of kernels [24], making them applicable to tasks such as classification, regression, and clustering. A notable characteristic of SVM is its avoidance of local minima. The SVM framework is based on a selective subset of the training data known as support vectors, which provides a sparse representation of the training data. This selection allows the model to operate only on essential data points—those represented by the support vectors.

According to the representer theorem, with certain reasonable assumptions, the solution to SVR can be expressed as a linear combination of kernel functions. According to this principle, as seen in (6), only the samples $\mathbf{x}_i$ for which $\alpha_i \neq 0$ influence the function $f(\mathbf{x})$; these samples are referred to as support vectors. Additionally, when using a polynomial kernel $k(\cdot)$, it becomes evident that the representation in (6) lacks uniqueness if the sample size N is excessively large.

A classifier is termed universally consistent if the risk associated with its decision functions converges in probability to the Bayes risk across all underlying distributions. Asymptotic lower bounds for the number of support vectors are derived in [25].

A connection between SVM and a sparse approximation approach similar to the basis pursuit denoising algorithm is discussed in [26]. In the case of noiseless data, a modified form of basis pursuit denoising aligns with SVM. Additionally, SVM can be formulated within the regularization theory framework, highlighting the relationship between sparse approximation, regularization theory, and SVM [26].

This paper offers a comprehensive literature review and tutorial on kernel methods and SVMs, structured with kernel machines first, followed by SVMs. Section 2 introduces the principles of kernel methods and the representer theorem, with representative kernel machines discussed in Section 3 and multiple kernel learning in Section 4. The SVM model and basic methods for solving its QP problem are presented in Sections 5 and 6, while least squares SVM (LS-SVM) is covered in Section 7. Section 8 details various SVM training approaches, and Section 9 addresses architecture optimization. Multi-class SVM categorization is explored in Section 10, followed by support vector regression (SVR), support vector clustering (SVC), and one-class SVM in Sections 11–13. Online learning approaches are outlined in Section 14, and SVMs for active, transductive, and semi-supervised learning are discussed in Section 15. The paper concludes in Section 16 with suggestions for future research directions.

2. Kernel Functions and Representer Theorem

Definition 1 (Kernel Function)**.** *A kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be understood as an inner product in a high-dimensional feature space, expressed as*

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \tag{1}$$

where the function $\phi : \mathcal{I} \to \mathcal{F}$ maps the input space $\mathcal{I}$ to the feature space $\mathcal{F}$.

Equation (1) is commonly referred to as the kernel trick. It involves a Hilbert space $\mathcal{H}$ serving as the feature space for the kernel $k(\cdot)$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$. Both $\mathcal{H}$ and ϕ can take various forms; however, for any given kernel, an RKHS can be established. In kernel methods, data points are characterized as functions or components within RKHSs linked to positive definite kernels. For applications in complex-valued signal processing, input data can be mapped into a complex RKHS utilizing pure complex kernels or real-valued kernels through a technique known as complexification [27].

The frequently utilized kernel functions include the polynomial, sigmoidal, and Gaussian functions [3]:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T \mathbf{x}_j + \theta\right)^m, \tag{2}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(a\mathbf{x}_i^T \mathbf{x}_j + \theta\right), \tag{3}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{e}^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}, \tag{4}$$

where integer $m > 0$, $a, \theta \in \mathbb{R}$, and deviation $\sigma > 0$.

Kernels play a vital role in kernel machines, and the chosen set of kernels must satisfy several criteria: they should permit linear parameterization for ease of use, and be dense enough in the function space to ensure accuracy, and each kernel should be universal to maintain an infinite-dimensional hypothesis space for scalability. While Gaussians fall short in terms of tractability and accuracy, and polynomials are not scalable, tessellated kernels [28] fulfill all three requirements. Consequently, utilizing tessellated kernels for kernel learning can eliminate the need for selecting candidate kernels and their associated parameters.

In kernel-based machine learning, models are expressed as $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$, where $\phi(\mathbf{x})$ represents a transformation from the input space to an RKHS, which can potentially be infinite-dimensional.

In machine learning using kernels, models are formulated as $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$, where $\phi(\mathbf{x})$ represents a transformation from the input space to an RKHS, which may be infinite-dimensional. This framework can simplify the minimization problems associated with the learning process. When algorithms can be represented using dot products within the RKHS, the feature mapping $\phi(x)$ can be replaced by a kernel function defined as $k(x, x') = \langle \phi(x), \phi(x') \rangle$.

Definition 2 (Kernel Definiteness)**.** *Consider a kernel matrix $\mathbf{K} = \left[k(\mathbf{x}_i, \mathbf{x}_j)\right]_{n \times n}$ for n data points. If $\mathbf{v}^T \mathbf{K} \mathbf{v} \geq 0 \quad \forall \mathbf{v} \in \mathbb{R}^n$, then the kernel $k(\cdot)$ is referred to as positive definite. If this condition holds only for those $\mathbf{v}$ satisfying $\mathbf{1}_n^T \mathbf{v} = 0$, then $k(\cdot)$ is termed conditionally positive definite. A kernel is classified as indefinite if there exist $\mathbf{v}$ and $\mathbf{v}'$ such that $\mathbf{v}^T \mathbf{K} \mathbf{v} > 0$ and $\mathbf{v}'^T \mathbf{K} \mathbf{v}' < 0$ for some $\mathbf{K}$.*

The squared Euclidean distance is extended to a high-dimensional space $\mathcal{F}$ through the kernel trick

$$\|\phi(\mathbf{x}) - \phi(\mathbf{y})\|^2 = k(\mathbf{x}, \mathbf{x}) + k(\mathbf{y}, \mathbf{y}) - 2k(\mathbf{x}, \mathbf{y}). \tag{5}$$

This extension is feasible for conditionally positive definite kernels.

Theorem 1 (Representer Theorem)**.** *Any function f that exists in an RKHS can be expressed as a linear combination of Mercer kernels,*

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \tag{6}$$

where $\alpha_i \in \mathbb{R}$ represent appropriate coefficients, and $k(\cdot)$ can be represented as an inner product in a certain high-dimensional feature space.

The representer theorem has been extended to include differentiable loss functions [29] as well as general monotonic loss functions [30]. Additionally, a quantitative version of the representer theorem has been developed in [25] that does not depend on the dual problem, particularly for convex loss functions.

In [31], both finite-sample and infinite-sample representer theorems are introduced for concatenating (linear combinations of) kernel functions within RKHSs. These findings offer a mathematical basis for analyzing machine learning algorithms that depend on the composition of functions.

Mercer's theorem does not extend to indefinite kernels. Nonetheless, the representer theorem remains applicable within reproducing kernel Krein space (RKKS) for non-positive kernels, even when the regularization component is nonconvex [32]. RKKS offers a characterization of both primal and dual problems for SVMs utilizing indefinite kernels [33], where the inner products can be negative.

Additionally, certain reproducing kernel Banach spaces (RKBSs) serve as suitable hypothesis spaces for implementing sparse learning approaches. Explicit representer theorems have been established for learning functions from a finite set of sampled data points in Banach spaces [34].

Example 1. *The XOR problem is linearly inseparable. Define* $k(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}^T \mathbf{x}_i)^2$, *where* $\mathbf{x} = (x_1, x_2)^T$ *and* $\mathbf{x}_i = (x_{i1}, x_{i2})^T$. *The training samples* $\mathbf{x}_1 = (-1, -1)$ *and* $\mathbf{x}_4 = (+1, +1)$ *belong to class* 0, *while* $\mathbf{x}_2 = (-1, +1)$, $\mathbf{x}_3 = (+1, -1)$ *belong to class 1.*

Expanding the kernel function yields $k(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)$, *where* $\phi(\mathbf{x}) = (1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2)^T$, $\phi(\mathbf{x}_i) = (1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2})^T$. *This transformation maps the input into a six-dimensional feature space, where the decision boundary is established by the condition* $x_1x_2 = 0$. *In this feature space, if* $x_1x_2 \geq 0$, *a sample is classified as belonging to class* 0; *otherwise, it is classified as class 1.*

3. Kernel Machines

3.1. Kernel PCA

Kernel PCA [9] integrates kernel functions within the PCA framework. It utilizes the kernel method to project the input data onto a high-dimensional feature space, where PCA is then performed.

Consider an input dataset $\{\mathbf{x}_i \in \mathbb{R}^{J_1} | i = 1, \ldots, N\}$. $\phi : \mathbb{R}^{J_1} \rightarrow \mathbb{R}^{J_2}$ is a nonlinear transformation from a J_1-dimensional input space to a J_2-dimensional feature space. In this transformed feature space, we can construct a $J_2 \times J_2$ correlation matrix, defined as

$$\mathbf{C}_1 = \frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i) \phi^T(\mathbf{x}_i), \tag{7}$$

and the set of feature vectors is restricted to have a zero mean such that $\frac{1}{N}\sum_{i=1}^{N} \phi(\mathbf{x}_i) = \mathbf{0}$. The mapping $\phi(\cdot)$ can be chosen following the procedure outlined in [35].

The principal components are subsequently determined by addressing the eigenvalue problem [3,9]

$$\lambda \mathbf{v} = \mathbf{C}_1 \mathbf{v} = \frac{1}{N} \sum_{j=1}^{N} \left(\phi^T(\mathbf{x}_j) \mathbf{v}\right) \phi(\mathbf{x}_j). \tag{8}$$

Consequently, $\mathbf{v}$ must lie within the span of the transformed data

$$\mathbf{v} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i). \tag{9}$$

By premultiplying both sides of (9) by $\phi(\mathbf{x}_j)$ and carrying out the necessary manipulations, the kernel PCA problem simplifies to

$$\mathbf{K}\boldsymbol{\alpha} = \lambda \boldsymbol{\alpha}, \tag{10}$$

where λ represents an eigenvalue, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^T$ is the corresponding eigenvector of the kernel matrix $\mathbf{K} = [K_{ij}]_{N \times N}$, with

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j). \tag{11}$$

Organize all the eigenvalues in descending order such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{J_2} > 0$, and denote their associated eigenvectors as $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{J_2}$. The eigenvectors are then normalized as

$$\boldsymbol{\alpha}_m^T \boldsymbol{\alpha}_m = \frac{1}{\lambda_m}. \tag{12}$$

The nonlinear principal components of $\mathbf{x}$ are derived by projecting the transformed pattern $\phi(\mathbf{x})$ onto $\mathbf{v}_m$ [3,9]

$$\mathbf{v}_m^T \phi(\mathbf{x}) = \sum_{j=1}^{N} \alpha_{m,j} k(\mathbf{x}_j, \mathbf{x}), \quad m = 1, 2, \ldots, J_2, \tag{13}$$

where $\alpha_{m,j}$ denotes the jth component of $\boldsymbol{\alpha}_m$.

Kernel PCA presents increased complexity and is prone to getting stuck in local minima. The eigendecomposition approach utilized in kernel PCA exhibits a time complexity of $O(N^3)$ for N training samples. Additionally, the kernel principal components are defined implicitly through linear combinations of the training data, necessitating the retention of all data post-training.

An efficient incremental approach introduced in [36] facilitates the rapid computation of the dominant kernel eigenbasis.

In PCA, an eigenvalue problem is solved for a $J_1 \times J_1$ matrix, whereas kernel PCA deals with an $N \times N$ matrix. Sparse approximations can lower computational costs [3], and algorithms like those in [37] efficiently recover principal kernel PCA components for classification tasks.

Kernel PCA's L_2 loss is sensitive to outliers; thus, sparsity is introduced via methods in [38,39]. An alternative approach uses LS-SVM [10], extending kernel PCA to generalized kernel component analysis with ϵ-insensitive loss [40] for improved robustness and sparsity.

The Kernel methods necessitate the precomputation and storage of the kernel matrix. Online versions like the kernel Hebbian algorithm [41], derived from the generalized Hebbian algorithm, compute kernels on demand with a scalar gain that adapts via annealing schedules, enhancing convergence [42]. Subset kernel PCA uses a sample subset to calculate bases, with online variations [43] allowing sample updates during iterations.

Incremental kernel PCA, particularly in Krein space, avoids preimage computation [18], while adaptive kernel PCA [44] tracks principal components using recursive eigendecomposition, ideal for nonstationary data, supporting consistent updates with stable memory usage.

Robust PCA retrieves low-rank matrices amid sparse noise [45], whereas robust kernel PCA [46] decomposes corrupted matrices into sparse and low-rank parts. This nonconvex optimization, tackled via alternating direction method of multipliers (ADMM), which incorporates backtracking line search and adaptive step sizess, handles non-differentiable challenges effectively.

Kernel PCA and kernel singular value decomposition (SVD) may differ if data are not centered. Techniques like reduced set expansions [47] compress the kernel SVD basis, enabling constant-speed incremental updates. An incremental kernel PCA method combining compression with kernel subspace re-orthogonalization [48] achieves linear time complexity, while robust kernel PCA [49] leverages eigenvalue decomposition (EVD) of weighted covariance, enhancing outlier resistance.

Kernel projection pursuit [39] finds sparse directions akin to PCA. Sparse feature extraction methods [50] using orthogonalization operate at $O(N)$ complexity. Kernel entropy component analysis [51] structures Renyi entropy via Parzen-windowed kernel matrices, projecting onto entropy-preserving axes. Widely linear complex kernel PCA [52] efficiently handles small sample sizes with extended linear approaches.

3.2. Kernel LDA

Kernel LDA consists of two main steps: mapping data from $\mathcal{X}$ to an RKHS $\mathcal{F}$ using a kernel function, followed by applying LDA in this space. Similar to standard LDA, kernel LDA can face issues with singular matrices, which are addressed by regularization techniques, such as adding a scalar to the inner product matrix [11], or by incorporating kernels into the transformation $\mathbf{w}$ [3,11].

A hard-margin linear SVM, when all training points act as support vectors, reduces to LDA [53], showing similar performance to kernel LDA. Kernel QP for feature selection aligns with the kernel Fisher vector [54].

For multi-class problems, kernel LDA [55] uses QR decomposition to address singularity, demonstrating greater effectiveness than kernel PCA for tasks like face recognition [56]. Kernel direct discriminant analysis [56] extends direct LDA, while complete kernel LDA [12], which integrates kernel PCA with LDA, outperforms other variants by analyzing data in two discriminant subspaces, utilizing both regular and irregular discriminant information. A kernelized one-class LDA is also introduced in [57].

Kernel quadratic discriminant analysis [58] applies a regularized kernel Mahalanobis distance, effective for datasets with varying class distributions in the kernel space. Meanwhile, kernel uncorrelated discriminant analysis and kernel regularized discriminant analysis [59] focus on projecting samples from the same class onto a common vector, with regularization mitigating overfitting for large samples per class.

Robust kernel fuzzy discriminant analysis [60] employs fuzzy memberships to minimize outliers' influence and applies kernels for nonlinear separability. A class separability criterion adapted for kernel space [61] offers a lower bound for the maximum objective value of kernel LDA.

The common vector method, which represents classes using subspaces, is extended to kernel discriminant common vectors for optimal LDA criterion solutions [62]. The kernel Foley-Sammon method generalizes this to a nonlinear LDA under orthogonality constraints [63], leading to cubic complexity for each discriminant vector due to matrix inversion. A fast variant reduces complexity via rank-one updates and QR decomposition, reducing the complexity to a quadratic level [64]. Using kernel Gram-Schmidt orthogonalization instead of kernel PCA in the preprocessing step further reduces costs [65].

The approach in [66] aims to make the Bayes classifier linear by mapping original class or subclass distributions onto a kernel space, allowing them to be separated by a hyperplane. This criterion is used in methods like LDA, nonparametric discriminant analysis, and subclass discriminant analysis.

3.3. Kernel Clustering

Examples of kernel-based clustering include kernel C-means [9], kernel subtractive clustering [67], a technique focused on minimizing the trace of the within-class scatter matrix [13], and SVC [68].

Kernel C-means detects clusters that are not linearly separable, ensuring monotonic convergence with positive semidefinite kernels, but not with others [9]. Variants include weighted kernel C-means [69], which is associated with graph partitioning, and multilevel kernel C-means [70], which enhances efficiency by avoiding full kernel matrix computation.

Kernelized fuzzy C-means (FCM) [71] replaces Euclidean distance with a kernel function, yielding better performance over FCM. When employing a Gaussian kernel, kernel C-means and kernel FCM demonstrate comparable classification performance [72]. Subtractive clustering has also adopted kernel-induced distances [67].

The kernel self-organizing map (SOM) [73] integrates neighborhood learning and the distance kernel trick, while the self-organizing mixture network [74] employs neurons in the SOM as Gaussian or other types of kernels, effectively modeling a mixture of distributions within the data. By optimizing an energy function, the kernel SOM operates in the feature space [75]. Despite its kernel nature, kernel SOMs do not consistently outperform traditional SOMs [75].

The kernel-based maximum entropy learning rule (kMER) [76] addresses clustering underutilization by updating prototype vectors and kernel radii, but its multi-winner approach can be computationally inefficient. SOM-kMER [77] merges SOM with kMER for improved clustering, and probabilistic SOM-kMER [78] incorporates a probabilistic neural network for classification.

Additional approaches include kernel neural gas [79], an online variant of kernel C-means [9], kernel possibilistic C-means [80], kernel spectral clustering [81], and an SVM-based online clustering algorithm tailored for nonstationary data [82].

Kernel density estimation is a nonparametric method used to create probability density functions (pdfs) from observed data, but it can be computationally demanding. A recursive clustering algorithm utilizing this estimation has been proposed in [83]. Additionally, the cluster kernel approach [84] offers continuously updated kernel density estimators suitable for streaming data.

Distributed kernel C-means algorithms enhance large-scale dataset processing by clustering across machines, often using approximated transformed feature vectors generated by methods like Nystrom approximation [85,86], random feature approximation [87], and kernel PCA. Additionally, leveraging the sparsity of the kernel matrix can enhance the scalability of kernel C-means [88]. These distributed algorithms may need either a kernel matrix generated from the complete dataset of users [88] or a subset of user data to be shared across different the machines [85,86].

Federated kernel C-means [89] employs a federated eigenvector approximation algorithm for distributed clustering, achieving convergence at a rate of $O(1/T)$, with T as the iteration count.

3.4. Kernel Ridge Regression and Gaussian Process Regression

Regularized least-squares (LS) regression, kernel ridge regression (KRR), and Gaussian process regression are key kernel methods in statistics and machine learning, generally exhibiting cubic complexity in relation to the number of data points [90].

In [91], the presented L_q-regularized kernel regression $(0 < q \leq 1)$ approach minimizes an LS loss with an L_q-norm penalty on coefficients within a kernel-induced feature space. For L_1-regularization, they derive convergence rates, highlighting the model's sparsity and efficiency.

Saturation in KRR [92] can be mitigated through iteration-based methods like kernel-based gradient descent [93], conjugate gradient descent [94], and kernel-based partial least squares (PLS) [95], achieving optimal learning rates without saturation [94,95].

Boosted KRR [96] integrates L_2-boosting with KRR. This approach allows for a bias-variance trade-off by adjusting the number of boosting iterations instead of modifying the regularization parameter typical in KRR. Boosted KRR features a (semi-)exponential bias-variance trade-off, maintaining a stable connection between generalization error and iteration count. Additionally, with an adaptive stopping rule, boosted KRR can achieve the optimal learning rate without facing saturation.

Distributed kernel regression using a divide-and-conquer strategy [97] achieves capacity-independent optimal rates, with precise error bounds established through leave-one-out analysis of KRR and bias-corrected KRR. In noise-free scenarios, KRR can exceed rates of $O(N^{-1})$ for N samples. For distributed KRR, optimal learning rates and the ideal number of local processors are derived in [98], along with communication strategies to enhance performance.

A plug-in KRR estimator for nonparametric regression with random design [99] is effective across multi-dimensional support and for arbitrary mixed-partial derivatives, achieving optimal convergence rates with a single tuning parameter for all derivative orders.

Asymptotic convergence for generalized regression models in hyper-RKHS is shown in [100]. Hyper-RKHS can produce indefinite kernels associated with RKHS [32,101], including examples like the hyperbolic tangent kernel [102] and transformers' dot-product attention [103].

In hyper-RKHS, two regularized regression models with bivariate forms are KRR and support vector regression (SVR). To address scalability in large samples, divide-and-conquer techniques can be combined with Nystrom approximation. The kernel is derived from a wide range of positive definite and non-positive definite kernels, and the convergence behavior and learning rates of regularized regression algorithms in hyper-RKHS are explored in [100].

Gaussian process regression requires a correlation function. In [104], upper and lower error bounds are established for this regression, even with potentially misspecified correlation functions. Optimal convergence rates can still be achieved with quasi-uniform sampling, linking the convergence rates of Gaussian process regression and KRR, reflecting the relationship between Gaussian process sample paths and associated RKHS [104]. Gaussian process-based Bayesian learning is connected with with frequentist kernel methods in RKHS [104].

3.5. Kernel Logistic Regression

Kernel logistic regression is a penalized classification technique in an RKHS [105,106], but has higher computational costs than SVMs using decomposition algorithms, with a specific decomposition algorithm for kernel logistic regression detailed in [106].

An indefinite kernel learning framework for kernel logistic regression is established in [107], revealing its nonconvex nature. The model, expressed as the difference of two convex functions, utilizes a constrained concave-convex procedure (CCCP) for nonconvex optimization, with a concave-inexact-convex procedure (CCICP) to speed up iterations. This model outperforms standard kernel logistic regression and other indefinite learning algorithms.

Least-squares prediction is framed as a probabilistic inference problem. The authors of [108] achieve a low-rank approximation of the kernel matrix through projections in a structured Gaussian regression model. Gaussian process regression relates closely to kernel regression and can be viewed as a two-layer neural network. KTBoost [109] integrates kernel boosting with tree boosting, introducing either a regression tree or an RKHS regression function into the ensemble of base learners during each iteration.

Kernel ordinal regression solvers face high complexity due to multiple ordinal thresholds. Doubly stochastic gradient (DSG) integrates random feature approximation with stochastic functional optimization, but is limited to single RKHS optimization, making it unsuitable for ordinal regression. DSGOR [110], a DSG-like algorithm, uses a kernel accommodating decision functions with multiple thresholds and achieves a convergence rate of $O(1/T)$.

Kernel ordinal regression can also use techniques from kernel regression or classification, such as SVR [111] and cost-sensitive classification [112]. Another strategy decomposes ordinal regression into multiple binary classification tasks with ordinal decomposition matrices [113,114], though this may introduce ambiguities. Threshold models are widely used for ordinal regression, capturing global ordinal information through multiple thresholds.

3.6. Various Other Kernel Methods

Traditional online linear algorithms [115,116] are mostly adapted into their kernel equivalents, such as kernel adaline [117], kernel online learning [118,119], kernel least squares (LS) [120], kernelized PLS [121], kernel recursive least squares (RLS) [122,123], kernel least mean squares (LMS) [124], kernel Wiener filter [125], complex kernel LMS [27], kernel adaptive autoregressive moving-average (ARMA) algorithm [126], kernel conjugate gradient (CG) algorithm in online mode [127], and doubly robust Stein-kernelized Monte Carlo estimator [128].

LS-weighted kernel reduced rank regression [129] can be used to formulate component analysis methods, including PCA, LDA, CCA, spectral clustering, locality-preserving projections, and their kernelized and regularized versions.

A prevalent challenge with kernel-based online algorithms, such as stochastic gradient descent (SGD) or online gradient descent, is the curse of kernelization, as the model size increases linearly with the accumulated training data over time [118]. To tackle this issue, dual space stochastic variance-reduced gradient descent (SVRG) for kernel online learning [130] has ε-approximate linear convergence while preserving model sparsity, effectively mitigating the curse of kernelization. SVRG acts as a corrective measure to traditional SGD.

In finite training data scenarios, kernel LMS [124] is well-defined in an RKHS without needing a regularization term. Kernel affine projection algorithm [131] serves as a versatile framework for kernel LMS, kernel adaptive linear neurons (adaline), sliding-window kernel RLS, and regularization networks. In online settings, kernel CG algorithm [127] achieves convergence rates comparable to kernel RLS but with lower computational costs and no need for user-defined parameters. Meanwhile, kernelized PLS [121] matches the performance of kernel CG, albeit with significantly higher computational expenses.

Ivanov regularization [132] in kernel LS estimation enhances control over the estimator's norm compared to Tikhonov regularization. This approach enables the estimator to achieve an optimal convergence rate [132].

The kernel null-space approach is a strong one-class classification method but is vulnerable to training data corruption and lacks the ability to rank observations. The null-space kernel Fisher method [133] improves robustness against training data corruption by regularization, leading to iterative algorithms that refine label confidences.

In [134], two RKBS learning models, namely the minimum norm interpolation (MNI) problem and the regularization problem, are examined, deriving an explicit representer theorem that leads to sparse kernel representations.

In [135], kernel machines replace neurons in feedforward networks, with a greedy layer-wise training scheme. This model performs well compared to both traditional kernel machines and backpropagation-trained connectionist models.

Neural tangent kernel theory [136] studies over-parameterized neural networks by approximating their training dynamics with kernel regression when the network width is large. It is widely utilized to explain the generalization capability of overparameterized neural networks trained using gradient descent and SGD. As shown in [137], two-layer wide ReLU networks exhibit uniform convergence to their corresponding neural tangent kernel regressor. With random initialization, the neural tangent kernel function uniformly converges to a deterministic function across layers as the number of neurons approach infinity [138]. In the streaming setting, the expected prediction error of multi-layer networks trained with SGD converges, and the activation patterns of an L-layer network with width m scale only polynomially in m [138].

Kernel mean embeddings and kernel covariance operators represent the Bochner expectations and covariance operators of random variables embedded in an RKHS [139,140]. Specifically, kernel mean embedding maps a probability distribution into an RKHS, while kernel covariance operators underpin theoretical frameworks for kernel PCA, kernel ICA, and kernel CCA [141–143]. In [144], asymptotic results and finite sample error bounds are provided for autocovariance operators of stationary stochastic processes on a Polish space within an RKHS, particularly for ergodic and strongly mixing cases.

In [144], asymptotic results and finite-sample error bounds are provided for autocovariance operators of stationary stochastic processes in an RKHS over a Polish space.

In [145], constrained covariance and kernel mutual information methods are used for accessing independence by examining the covariance of functions derived from random variables in RKHSs. Meanwhile, kernel-based maximum a posteriori (MAP) classification [146] operates under the assumption of a Gaussian distribution within the feature space.

In [147], the authors introduces a sparsity-driven kernel classifier that seeks to minimize a data-dependent generalization error bound. This classifier integrates hinge loss to address training errors along with a concave penalty on the expansion coefficients. By employing a successive linearization technique, the nonconvex bound minimization problem is formulated as a series of linear programs, resulting in error rates similar to those of SVMs while utilizing much fewer support vectors. In contrast, the kernel classifier developed in [148] aims to minimize the L_2 or integrated squared error between density differences, leading to a sparse solution achieved through QP.

A graph-based framework is established for selecting a subset of kernels from a dictionary, utilizing a graph for guidance. Random feature approximation is employed to facilitate online function learning, and the proposed algorithms demonstrate sub-linear regret bounds [149].

Kernelized matrix factorization is an effective approach for incorporating side information into low-rank approximation models. Current methods typically rely on a regularization or a Markov chain Monte Carlo technique. A hierarchical kernelized sparse Bayesian matrix factorization model that incorporates side information is developed in [150]. This method automatically infers parameters and latent variables, such as reduced rank, by using variational Bayesian inference. This method achieves low-rankness through sparse Bayesian learning while enforcing columnwise sparsity with constraints on the latent factor matrices. Two kernel-based extensions of non-negative matrix factorization (NMF) include polynomial kernel NMF [151] and projected gradient kernel NMF [152].

Kernel thinning [153] is a technique designed to compress a distribution P more efficiently than independent and identically distributed (i.i.d.) sampling or traditional thinning methods. In [154], it is shown that boosting with a kernel-defined weak learner corresponds to an estimation process using a distinctive boosting kernel.

Kernelized inference rules, particularly the kernel Bayes' rule, are used to manipulate high-dimensional embeddings of distributions into RKHSs. However, the computational requirements of the kernel Bayes' rule increase significantly with the sample size, limiting its scalability for large datasets. An approximate solution, the kernel Kalman rule [155], addresses this issue by allowing for the precomputation of costly matrix inversions, thereby substantially lowering the computational complexity.

Kernel forward-backward smoother (KFBS) [155] is developed from a forward and backward kernel Kalman filter, incorporating a smoothing update within Hilbert space. Additionally, the subspace conditional embedding operator [155] serves as a sparsification method that utilizes the complete dataset.

Kernel-based reinforcement learning [156] develops a decision policy that converges to a unique solution and demonstrates statistical consistency. However, the model increases in size as the number of sample transitions increases. In contrast, kernel-based stochastic factorization [157] condenses this information into a fixed-size approximator, maintaining linear computational complexity in the number of sample transitions. A regression-based distributed policy evaluation algorithm within the RKHS framework is analyzed in [158]. This method focuses on a multistage iteration technique that utilizes infinite-dimensional gradient descent in an RKHS, with the Nystrom approximation used to project this space onto a finite-dimensional representation.

Nystrom method [159] is a widely used sampling-based technique for the approximation of large kernel matrices and their eigensystems. Two notable methods for approximating kernel matrices are random kitchen sinks and Fastfood. Random kitchen sinks [160,161] approximate shift-invariant kernels using their Fourier transform. This method approximates a function f by multiplying the input by a Gaussian random matrix followed by a nonlinear transformation. It leverages sampling to estimate the kernel representation of the sum in the inner product according to Mercer's theorem. With n nonlinear basis functions and d-dimensional inputs, this approach necessitates $O(nd)$ time and memory to compute f, making it significantly faster than the kernel trick when the sample size $m \gg n$. Fastfood [162] uses random Fourier features to approximate kernel functions, and employ low-rank representations to expedite the solution of kernel SVMs. By using Fastfood, the $O(nd)$ complexity of random kitchen sinks is reduced to $O(n \log d)$. This method exploits the property that Hadamard matrices, when multiplied by diagonal Gaussian matrices, exhibit similar behavior to Gaussian random matrices. Therefore, Hadamard and diagonal matrices can replace Gaussian matrices in the random kitchen sinks framework. Fastfood offers an unbiased approximation with low variance and converges at a rate comparable to that of random kitchen sinks, requiring $O(n \log d)$ in time and $O(n)$ in storage without compromising accuracy.

Kernel associative memory utilizes the kernel approach in associative memory [163]. It is a special form of kernel auto-associator. Kernel auto-associators function as versatile one-class learning machines applicable to novelty detection and m-class classification tasks. In [164], the recurrent kernel associative memory model is introduced as a kernelized version of recurrent correlation associative memory [165], and the update equation of the dynamic model can be treated as a classification step.

If two variables are nonlinearly correlated, CCA is incapable of correctly correlating this relationship. Kernel CCA [14,16] is able to find nonlinear relationships. The issue of singularity in the Gram matrix is addressed by incorporating a regularization term. The statistical convergence of kernel CCA is proved in [142]. An enhanced kernel CCA algorithm utilizing the EVD approach is presented in [166].

A set of kernel ICA algorithms [14] employs contrast functions derived from canonical correlations within an RKHS, utilizing kernel CCA. In [167], a kernel-based contrast function for ICA is proposed, corresponding to a regularized correlation measure within high-dimensional feature spaces. This formulation extends the LS-SVM framework to a multivariate setting for kernel CCA. In [15], second-order statistics in a kernel-induced feature space is leveraged in the kernel-based nonlinear BSS method. Additionally, kTDSEP [168] integrates kernel feature spaces with second-order temporal decorrelation BSS by utilizing temporal information.

Regularized sparse kernel slow feature analysis creates an orthogonal basis for real-world time series data by combining the kernel trick and sparsification techniques [169]. The approach in [170] involves learning kernels within a LASSO framework by utilizing a generative Bayesian learning and inference method, resulting in a robust algorithm that generates a sparse kernel model.

Besides data-independent approaches [160,171], leveraging low-rank structures of the kernel matrix proves beneficial for extracting informative features [172] and enhancing computational efficiency [173]. Kernelized low-rank representation, along with its robust variant, effectively handles nonlinear data [172]. The robust kernel low-rank representation is solved using alternating minimization method. Meanwhile, kernel matching pursuit [174] serves as a powerful greedy approach for creating a discriminative sparse kernel classifier.

Building on the perceptron algorithm, projectron [175] maps instances into the space defined by prior online hypotheses. Projectron++ extends this approach by incorporating the concept of a large margin. While the projectron algorithm performance is slightly inferior to that of the perceptron, Projectron++ surpasses perceptron, projectron, and forgetron [176] while maintaining a comparable hypothesis size. Moreover, to achieve a given target accuracy, the support sets of projectron and projectron++ are considerably smaller compared to those of forgetron.

In [177], expected bounds for leave-one-out cross-validation errors in kernel methods are derived, along with variance bounds, leading to generalization expectations across various kernel algorithms. Additionally, an RKHS framework is specifically designed for spike trains in [178].

In [179], the sampling theorem is reinterpreted within the context of an RKHS framework. The optimal approximation of functions in an RKHS is explored for both finite sampling points [180] and infinite sampling points [181]. In scenarios with measurement noise, the RKHS estimate aligns with the posterior mean, or minimum variance estimate, of a Gaussian random field, with the field's covariance linked to the RKHS kernel. A broader statistical interpretation, incorporating more general loss functions, is discussed in [182]. For any finite sampling set, the MAP estimate of the signal samples corresponds to the RKHS estimate evaluated at those specific points.

Restricted kernel machines (RKMs) [183] extend kernel methods by incorporating visible and hidden units, bridging the gap between kernel PCA, LS-SVM [184], and restricted Boltzmann machines (RBMs). RKMs share a similar energy function to RBMs. Generative-RKM [185] is a generative model based on RKMs that enables joint multi-view synthesis and independent feature extraction. This model offers both primal and dual formulations, allowing the integration of kernel-based methods and neural network models.

4. Multiple Kernel Learning

Multiple kernel learning (MKL) [186,187] involves combining multiple kernels to create an ensemble suited for applications with multiple, heterogeneous data sources. By integrating multiple views, MKL constructs base kernels from each view, effectively utilizing the complementary information provided by these kernels. MKL models are well-suited for jointly learning the optimal combination of features alongside their associated predictors.

Each convex set of kernel matrices can be represented by a set of semidefinite programs (SDPs). In [186], the initial formulation of MKL is presented as an SDP problem. A convex quadratically constrained quadratic program (QCQP) is formulated through the conic combination of various kernels, $k = \sum_i \alpha_i k_i$, chosen from a set of candidate kernels k_i [186]. For large-scale scenarios, this QCQP is then reformulated into a semi-infinite linear program, allowing the application of standard SVM techniques [188].

For regularized kernel discriminant analysis, the optimal kernel matrix is represented as a linear combination of predefined kernel matrices [189]. The challenge of kernel learning can be recast as a convex semidefinite programming problem. Furthermore, a convex QCQP formulation is suggested for binary-class kernel learning in the framework of regularized kernel discriminant analysis. The multiclass regularized kernel discriminant analysis can be effectively decomposed into several binary-class kernel learning tasks, which naturally leads to QCQP and semi-infinite linear programming (LP) formulations.

In [190], a multiple kernel LDA is presented that encourages sparsity by utilizing an L_1-norm regularization on the kernel weights. This problem of optimal kernel selection can be reformulated as a convex optimization problem that is both tractable and solvable using efficient interior-point methods. In regularized kernel discriminant analysis, the optimal kernel matrix can be expressed as a linear combination of predefined kernel matrices [189].

Sparse MKL [191] focuses on leveraging single kernel algorithms to achieve a sparser result regarding the number of kernels utilized. Most MKL methods enforce L_1-norm simplex constraints on the weights of kernel combinations, resulting in a sparse yet nonsmooth solution for these weights. A more flexible formulation is achieved in a generalized MKL model [192] by implementing an elastic-net-type constraint on the weights of the kernels.

SimpleMKL [193] utilizes a reduced gradient descent approach to optimize kernel weights, while HessianMKL [194] substitutes the gradient descent updates with a Newton update. In nonsparse scenarios, L_p-norm MKL provides superior bounds compared to L_1-norm MKL, and vice versa [195,196]. An analytical solver for nonsparse MKL has been benchmarked against various L_1-norm MKL techniques, including SimpleMKL, HessianMKL, as well as SILP-based wrappers and chunking optimization [188]. Both SimpleMKL and the analytical solver show improved efficiency with an increasing number of kernels, although their performance is still constrained by memory limitations. While HessianMKL operates significantly faster than SimpleMKL, it lags behind nonsparse interleaved methods and SILP in terms of speed. In sum, interleaved analytic along with cutting plane optimization techniques [195] results in speed improvements of one and two orders of magnitude compared to HessianMKL and SimpleMKL, respectively.

SpicyMKL [197] accommodates general convex loss functions along with various forms of regularization. It operates by iteratively addressing smooth minimization problems, eliminating the necessity for internal solutions to SVM, LP, or QP tasks. This method can be regarded as a form of proximal minimization technique and demonstrates superlinear convergence. The computational expense of the inner minimization process is approximately proportional to the number of active kernels.

In [198], kernel slack variables are introduced within a soft margin context for MKL, allowing the incorporation of various loss functions, such as hinge loss, square hinge loss, and square loss. Many MKL methods can be considered special cases within this framework, with L_1-norm MKL being viewed as hard margin MKL. In [199], SMO-MKL employs linear MKL with regularization based on L_p-norm squared or Bregman divergences, utilizing the sequential minimal optimization (SMO) method for training.

Bayesian efficient MKL [200] enables the efficient combination of hundreds or thousands of kernels through a fully conjugate Bayesian formulation paired with a deterministic variational approximation, accommodating both sparse and nonsparse weights. This MKL formulation aligns with maximum entropy discrimination, applying a noninformative prior across multiple views [201]. Additionally, a hierarchical Bayesian model is introduced to simultaneously learn the data-dependent prior alongside the classification model.

Multiple kernel FCM [202] enhances the FCM algorithm to incorporate MKL. Multiple kernel clustering [203–208] attempts to find an optimal kernel from several precomputed basic kernels to enhance clustering performance. This approach improves clustering by deriving a consensus partition or graph from a predefined set of kernels. Additionally, multiple kernel clustering with an adaptive multi-scale partition selection method (MPS) [209] leverages multiple-dimensional representations and pairwise cluster structures for effective clustering.

Moreover, multiple C-means clustering [206] focuses on finding the best combination of kernels for clustering tasks. It employs an alternating minimization method to iteratively refine both the coefficients of the selected kernels and the corresponding cluster memberships. Finally, a simple multiple kernel C-means method based on a min-max framework can achieve a high-quality unified kernel [210].

The scalability of MKL is limited to about ten thousand samples because of significant computational and memory demands. SVRG-MKL [211] represents an MKL solution with intrinsic scalability, capable of efficiently integrating multiple descriptors even when handling millions of samples. This solution exploits an SVRG approach, and operates directly in the primal space, thereby circumventing the computational and memory challenges associated with Gram matrix calculations. Additionally, the proposed optimization algorithm has linear complexity, ensuring computational efficiency.

The multi-kernel RBF network [212] employs a linear convex combination of different primary kernels, utilizing either fixed or adaptive weights adjusted through the orthogonal least squares (OLS) algorithm. This approach yields considerable performance improvements compared to relying on a single kernel.

DOMKL is a comprehensive distributed online learning framework, also known as fully decentralized online federated learning, utilizing multiple kernels [213]. It leverages an online ADMM and a distributed hedge algorithm. Throughout T time slots, DOMKL attains an optimal sublinear regret of $O(\sqrt{T})$, indicating that each learner within the network can converge to a common function, minimizing the gap from the best function identified retrospectively.

A scalable multi-kernel learning scheme known as Raker and an adaptive multi-kernel learning scheme referred to as AdaRaker [214] are designed to derive nonlinear learning functions dynamically. AdaRaker is particularly adept at monitoring nonlinear learning functions in scenarios with unknown dynamics, offering analytical guarantees for its performance.

5. SVM Model

Consider an unknown function $f : \mathbf{x} \to \{-1, 1\}$, with the training set $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, N\} \subset \mathbb{R}^n \times \{-1, 1\}$. The goal of SVM is to identify f by finding the optimal hyperplane that maximizes the separation margin between instances of the two classes, as depicted in Figure 2.

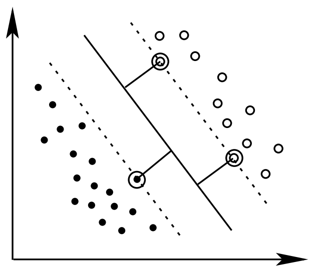

Figure 2. Depiction of the hyperplane within a two-dimensional feature space for SVM. The margin is determined by the distance between the hyperplane and the closest data points from each class. The points encircled represent the support vectors that define the margin. The points of different style represent points belonging to different classes.

To find the optimal hyperplane in a linear SVM, the followinig primal QP problem is solved [21]:

$$\min E_0(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_p \tag{14}$$

subject to

$$y_p\left(\mathbf{w}^T\mathbf{x}_p + b\right) \geq 1 - \xi_p, \quad p = 1, \ldots, N, \tag{15}$$

$$\xi_p \geq 0, \quad p = 1, \ldots, N, \tag{16}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)^T$ represents the slack variables, and $\mathbf{w}$ and b denote the weight vector and bias term of the hyperplane, respectively. The output of the classifier is $y_p \in \{-1, +1\}$, and the regularization parameter C adjusts the balance between maximizing the margin and minimizing classification errors. The optimal value of C can be set by using cross-validation as a statistical model selection technique.

When ξ_p in (14) represents a loss function based on the L_1-norm, the resulting L_1-SVM addresses the unconstrained optimization problem:

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{j=1}^{N}\max(1 - y_j\mathbf{w}^T\mathbf{x}_j, 0). \tag{17}$$

Likewise, L_2-SVM employs the sum of squared losses:

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{j=1}^{N}\left[\max(1 - y_j\mathbf{w}^T\mathbf{x}_j, 0)\right]^2. \tag{18}$$

L_1-SVMs are often preferred over L_2-SVMs in that they typically produce classifiers having significantly fewer support vectors. Additionally, SVMs have a connection to regularized logistic regression, which aims to minimize the objective of

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{j=1}^{N}\log(1 + e^{-y_j\mathbf{w}^T\mathbf{x}_j}). \tag{19}$$

By the employing Lagrange multiplier method and substituting $\mathbf{x}_p^T\mathbf{x}$ with $k(\mathbf{x}_p, \mathbf{x})$, SVM learning is targeted as determining α_p, for $p = 1, \ldots, N$ that minimizes the dual quadratic form [21]

$$E_{\text{SVM}} = \frac{1}{2}\sum_{p=1}^{N}\sum_{i=1}^{N} y_p y_i k(\mathbf{x}_p, \mathbf{x}_i)\alpha_p\alpha_i - \sum_{p=1}^{N}\alpha_p \tag{20}$$

subject to

$$\sum_{p=1}^{N} y_p \alpha_p = 0, \tag{21}$$

$$0 \le \alpha_p \le C, \quad p = 1, \ldots, N, \tag{22}$$

where α_p represents the kernel weight associated with the pth example, defined as $k(\mathbf{x}_p, \mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x}_p)\boldsymbol{\phi}(\mathbf{x})$, with $\boldsymbol{\phi}(\cdot)$ being implicitly defined by the chosen kernel function. When using a linear kernel function, $k(\mathbf{x}_p, \mathbf{x}) = \mathbf{x}_p^T \mathbf{x}$, SVM simplifies to linear SVM [215].

The output of SVM is given by

$$y(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b), \tag{23}$$

which, upon manipulation, yields a classification decision

$$y(\mathbf{x}) = \mathrm{sign}\left(\sum_{p=1}^{N} \alpha_p y_p k(\mathbf{x}_p, \mathbf{x}) + b \right). \tag{24}$$

The QP problem outlined in (20) to (22) will conclude once all Karush-Kuhn-Tucker (KKT) conditions are satisfied:

$$\begin{cases} y_p u_p \ge 1, & \alpha_p = 0 \\ y_p u_p = 1, & 0 < \alpha_p < C \\ y_p u_p \le 1, & \alpha_p = C \end{cases}, \tag{25}$$

where u_p represents the SVM output for the p-th example. Instances with nonzero α_p are identified as support vectors, as they reside on the margin.

In a two-class L_2-SVM, all slack variables ξ_i are assigned the same penalty factor C. However, for imbalanced datasets, it is often beneficial to apply different C values for each class. More generally, each training instance can be assigned its own individual penalty factor C_i.

The generalization capability of SVM is affected by the span of the support vectors [216], which is consistently demonstrated to be smaller than the diameter of the smallest enclosing sphere for these support vectors, as referenced in earlier bounds [217]. This span-based prediction of test error is not only accurate but also aids in determining optimal parameters for SVM. Additionally, the expected error bounds for SVM are established using the leave-one-out estimator, providing an unbiased estimate of the probability of test error [216].

The upper limit for the VC dimension of hyperplanes with a margin of ρ is given by $\frac{D^2}{4\rho^2}$, where D represents the diameter of the smallest sphere that encloses the training dataset [21]. By analyzing the VC dimension associated with the homogeneous polynomial and Gaussian RBF kernels, it is evident that SVM can exhibit a very large, potentially infinite, VC dimension [218]. This suggests that SVM possesses a robust capacity for both classification and regression.

SVM Versus Neural Networks

Generative machine learning methods, including the naive Bayes classifier, LDA, and neural networks, focus on estimating the conditional probability distributions that underlie the data. The effectiveness of these solutions can be significantly influenced by the choice of initialization and the criteria for termination. The assumptions made about the generative processes, along with parameter estimation, help streamline the learning process.

In contrast, discriminative approaches prioritize optimizing prediction accuracy rather than modeling the underlying data distribution. A discriminative classification function evaluates a data point **x** to classify it as one of several classes. Learning a classifier involves determining the equation of a multidimensional surface that effectively distinguishes between the different classes within the feature space.

As a discriminative method, SVM analytically tackles convex optimization problems, ensuring that the optimal parameters for the separating hyperplane remain consistent. The hyperplane identified by SVM for two-class classification corresponds to the outcome achieved through LDA applied to the support vectors [53].

In contrast to neural networks, SVM is not affected by issues such as local minima, the curse of dimensionality, or overfitting. SVM mitigates overfitting by emphasizing the minimization of structural risk, whereas neural networks aim to minimize empirical risk. Additionally, SVM inherently determines model complexity through the selection of support vectors, while neural networks manage complexity by restricting the feature set.

SVM boasts three notable characteristics:

- Sparsity. The complexity associated with classification primarily relies on the count of support vectors instead of the dimensionality of the input space.
- Kernel-based approach.
- Maximum-margin separator. The hyperplane must be positioned to maintain the greatest possible distance from the different classes.

6. Solving the QP Problem

L_1-norm soft-margin SVM, often referred to as QP SVM, was initially introduced with polynomial kernels in [20] and later extended to general kernels in [219]. LP-SVM is known for its efficiency and can outperform QP-SVM due to its linearity and adaptability to large datasets. LP-SVM has a convergence behavior comparable to that of QP-SVM [220]. SVMs utilizing the L_1 or L_∞-norm lead to an LP problem, which generally incurs lower computational costs in comparison to SVMs based on the L_2-norm.

By employing kernel mapping, SVM learning can be formulated as a quadratic optimization problem that has a unique global solution. However, a straightforward application of the QP solver for N training patterns results in a time complexity of $O(N^3)$ and a space requirement of at least $O(N^2)$.

General convex QPs are commonly dealt with using either an interior-point method or an active-set method. An interior-point method is typically employed when the Hessian $\mathbf{Q}$ of the objective function or the constraint matrix of the QP problem is large and sparse. In contrast, moderate-sized problems with dense matrices often benefit from the active-set method. In SVM scenarios, $\mathbf{Q}$ is usually dense, making large SVM challenges for both methods. For certain classes of SVMs with dense but low-rank $\mathbf{Q}$, it is possible to adapt an interior-point method to operate efficiently [221,222]. For high-rank $\mathbf{Q}$, an active-set method may be more appropriate.

The simplex method, which is an active-set approach, is used for solving both LP and QP problems. Active-set methods significantly benefit from warm starts, whereas interior-point methods do not. SVM-QP [223] addresses the convex QP problem utilizing the simplex method. Overall, SVM-QP outperforms SVMlight while maintaining identical generalization properties and extends to incremental mode with minimal changes. In each iteration, SVM-QP fixes all variables in the current dual active set to their respective values (0 or C) before solving the reduced dual problem.

Large SVM problems are often handled using restricted active-set methods, such as chunking [20] or decomposition [224–226], which vary only a limited number of variables in each iteration. While these methods can converge slowly as they are near the optimal solution, their performance can be sensitive to the choice of chunk size, which can be challenging to select. A full active-set method can help address these issues. Active-set methods were utilized in [227] to generate the complete regularization path for the cost parameter for standard L_2-SVM.

6.1. Chunking

Chunking [215,217] is a technique that decomposes a large QP problem into smaller subproblems, ultimately allowing for the identification of all nonzero α_p. This method starts with a randomly selected subset of data, referred to as a chunk, which can fit into memory. A general optimizer is utilized to address the optimization problem and train an initial SVM. Within this chunk, support vectors are retained, while other data points are discarded. These discarded points are then replaced by a new working set composed of those points that show significant violations of the KKT conditions. The SVM undergoes retraining, and this process continues iteratively. One significant drawback of this method is that, at the conclusion of the training process, the complete set of identified support vectors must be retrained.

Chunking exploits the sparsity inherent in the solution of SVM. Throughout the optimization process, the number of active candidate support vectors may significantly exceed those that are eventually chosen, which could surpass the limits of the chunking space. Nonetheless, the kernel matrix might still be too large to be accommodated in memory. If KKT conditions are evaluated to form a new working set without effective kernel caching, it can lead to substantial computational expenses. The standard projected conjugate gradient chunking algorithm exhibits a complexity ranging from $O(N)$ to $O(N^3)$ for a training set containing N samples [226,228].

6.2. Decomposition

A significant QP challenge can be broken down into smaller QP subproblems [224,225,229], where each subproblem is initialized using the outcomes of the preceding one. This decomposition process necessitates a QP algorithm, such as the projected conjugate gradient, and has the potential to lower the $O(N^3)$ time complexity to $O(N^2)$.

Decomposition methods perform two key operations iteratively until an optimality condition is met: selecting q variables to form the working set from a total of l variables and then minimizing the objective function by updating only the chosen q variables. Each iteration focuses solely on optimizing the working set while keeping the remaining variables constant [224]. The SMO algorithm [225] selects two variables for the working set in each iteration, while SVMlight [226] permits any even number q of variables in the working set.

In SMO, each small QP problem has two variables, α_p, and is solved using analytical methods. The first variable is selected from data points that do not satisfy the KKT conditions, while the second variable is selected to achieve a significant improvement in the dual objective function. This procedure continues until all training examples meet the relaxed KKT conditions, avoiding QP optimization in the inner loop. SMO has a memory requirement of $O(N)$, and its computational complexity varies between $O(N)$ and $O(N^2)$, which is more efficient than that of projected conjugate gradient chunking. This complexity primarily arises from SVM evaluations, making SMO particularly efficient for linear SVMs and sparse datasets [225].

By substituting single-threshold parameters with dual-threshold parameters, the two-parameter SMO algorithm largely enhances performance compared to standard SMO [230]. In a similar fashion, the three-parameter SMO [231] optimizes three selected parameters simultaneously, demonstrating substantial improvements over the two-parameter SMO in both execution speed and computational efficiency.

For SMO algorithms, a strict decrease in the objective function is guaranteed only when the working set consists of a violating pair [232]. If not, convergence cannot be assured. Due to the involvement of only two variables in each iteration, convergence tends to be slow. However, the simplicity of each iteration often results in overall speedup. Decomposition methods that utilize α-seeding have proven effective in solving a series of linear SVMs, particularly when the dataset contains more data points than attributes [233].

The choice of the working set plays a pivotal role in decomposition methods. A common approach involves choosing the maximal-violating pair. When maximal-violating pairs are used as working sets, the resulting SMO-type algorithm is known as the maximal-violating-pair algorithm.

Both SMO and an interior-point method are implemented in LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/, accessed on 1 November 2024) [234]. In LIBSVM, working-set selection integrates partial second-order information, boosting convergence rates with a modest increase in computational expense [235]. This strategy achieves linear convergence. A related technique, hybrid maximum-gain working-set selection [236], focuses on minimizing kernel computations per iteration by reducing cache misses within the decomposition process. SMO is also utilized in liquidSVM [237].

Rather than utilizing a working set of two, ThunderSVM training [238] employs a larger working set and solves several SMO subproblems in parallel. The working set is constructed from instances with the highest violations of the optimality condition. The relevant kernel values for each subproblem associated with the working set are calculated, and then SMO is used to solve each subproblem individually. This cycle is repeated until the stopping criterion is met.

In SVMlight [226], the steepest feasible descent direction with q nonzero elements is identified, allowing for an effective selection of q variables as the working set. When $q = 2$, this selection aligns with the optimal pair used in a modified SMO method [234]. To avoid kernel reevaluation, SVMlight caches q rows of the kernel matrix, updated using a least recently used (LRU) strategy. However, for very large training sets, memory limitations require reducing the number of cached rows. A generalized maximal-violating-pair strategy is applied for selecting the working set, along with a solver for inner QP subproblems. For smaller working sets ($q = O(10)$), the SVMlight performance is on par with that of LIBSVM.

Sigmoidal kernels can result in kernel matrices that are not positive-semidefinite. To handle the nonconvex nature of such dual problems, SMO decomposition is applied, enabling LIBSVM software version 2.8x to provide solutions for sigmoidal kernels. The support vector perceptron [239] builds on SVM by using a sigmoidal kernel, providing maximum margin solutions in separable cases and generating compact architectures that compare favorably with multilayer perceptrons (MLPs) and LIBSVM employing sigmoidal kernels.

SimpleSVM [229] is designed for scaling up the method. In each iteration, a point that breaches the KKT conditions is incorporated into the working set, updating the kernel matrix via a rank-one modification. Nonetheless, managing storage demands for large, dense kernel matrices remains an obstacle. SimpleSVM categorizes the dataset into three types: non-bounded support vectors ($0 < \alpha_w < C$), bounded instances that are either misclassified or on the margins ($\alpha_C = C$), and non-support vectors ($\alpha_0 = 0$). The optimization process leads to solving a linear system with α_w as the target variable.

Various kernel techniques can be reinterpreted as minimum enclosing ball (MEB) problems in computational geometry [240]. By utilizing an efficient approximate MEB algorithm, it is possible to obtain solutions close to optimal by employing core sets. In core vector machines, the core set serves a function analogous to the working set in other decomposition techniques. Kernel-based models, including soft-margin one-class and two-class SVMs, can thus be treated as MEB problems, allowing for the efficient acquisition of approximately optimal solutions via core sets. The core vector machine [240] supports nonlinear kernels, operating with a time complexity of $O(N)$ and a space complexity independent of N. This approach matches the accuracy of traditional SVMs but with faster execution and scalability to larger datasets. For smaller datasets, where $N \ll 2/\varepsilon$, SMO may yield better performance. By tuning ε, a balance between computational efficiency and approximation quality can be managed, with $\varepsilon = 10^{-6}$ being suitable for most cases. This MEB setup also corresponds to the hard-margin support vector data description (SVDD) [241].

Core vector machine resembles decomposition algorithms but simplifies the subset selection process significantly. The QP is solved on the core set exclusively using SMO, thereby obtaining α. Its stopping criterion parallels the one used in ν-SVM [242].

Core vector machine, however, does not guarantee convergence to the optimal solution under all hyperparameter settings [243]. For kernel methods to be formulated as MEB problems, their dual objectives must lack a linear term; this condition is satisfied by one-class and two-class SVMs but not by SVR [240]. Generalized core vector machine [244] addresses this by introducing the center-constrained MEB problem, allowing SVR to also be represented as an MEB problem. It maintains the same asymptotic time and space complexity as core vector machine but offers faster performance and yields fewer support vectors on very large datasets.

Example 2. *In this study, we utilize LIBSVM to classify a dataset comprising* 463 *samples distributed across three classes within a two-dimensional space. We employ the RBF kernel defined by* $e^{-\gamma\|\mathbf{u}-\mathbf{v}\|^2}$*, setting the parameters to* $\gamma = 200$ *and* $C = 20$*. This configuration enables the accurate classification of all samples, resulting in a total of* 172 *support vectors. The classification outcomes are presented in Figure 3.*

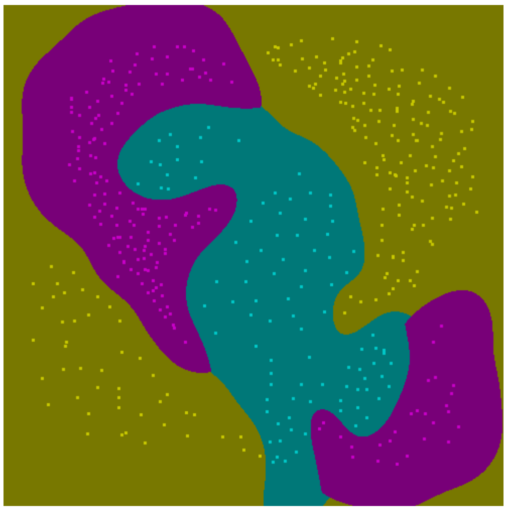

Figure 3. Decision boundary of the SVM for classifying three categories within a two-dimensional space. Points of different color belong to different classes.

The two-variable decomposition method, as applied in [245], offers a practical and efficient solution for handling sub-problems involving up to 50 variables. Various strategies have been investigated for selecting the working set and determining its size, integrating both first-order and second-order working-set selection rules along with a method to utilize cached elements of the Hessian matrix.

A significant bottleneck in SVM training often arises from the computation of kernel values. Since the same kernel values can be reused across multiple iterations during training, caching these values can prove beneficial. Libraries such as LIBSVM [234], SVMlight [226], and liquidSVM [237] implement the least recently used (LRU) replacement policy for caching kernel values. While LRU performs well in situations with strong temporal locality, which is not always present during SVM training, it tends to be most effective in later training stages [246]. The less frequently used (LFU) caching strategy improves upon LRU by replacing kernel values that are infrequently accessed, often achieving a hit ratio that is 20% higher than that of LRU during training with a Gaussian kernel. Hybrid caching for SVM training (HCST) dynamically adjusts the caching strategy at different training stages [246], and this approach is integrated into ThunderSVM, a library designed for many-core processors. ThunderSVM achieves significant speed improvements by employing the HCST strategy while incurring minimal memory overhead.

Additionally, a data decomposition strategy known as localized SVMs [247] enables the solution of SVMs on multiple spatially defined data segments, resulting in enhanced time and space efficiency. A notable feature of this method is the flexibility to choose arbitrary kernels and regularization parameters for each spatial cell. On liquidSVM [237], localized SVMs have been successfully applied to datasets containing up to 32 million training samples [248]. Moreover, global learning rates adapted specifically for localized SVMs that apply Gaussian kernels with hinge loss are derived in [249].

6.3. Convergence of Decomposition Methods

The strategy for updating the working set is theoretically critical, as it ensures that the objective function monotonically decreases in each iteration [232]. The global convergence properties of decomposition methods and SMO algorithms in classification tasks are established in [250–253]. Under appropriate convexity assumptions, the convergence of SVMlight is also demonstrated in [250,252].

When the working set size is restricted to 2, selecting pairs based on the maximal-violating-pair principle ensures asymptotic convergence for the decomposition process [250,251,253]. For larger working sets, an additional condition requiring that the gap between successive approximations reduces to zero guarantees convergence [252]. Generalized SMO has been shown to complete within a finite number of steps, provided specific stopping criteria are met [251]. Moreover, a straightforward proof of the asymptotic linear convergence of SMO-type methods is provided in [253], when selecting the two-element working set through a general criterion. Global convergence properties for both generalized SMO and SVMlight are verified in [254].

SVMs may occasionally involve non-conditionally positive-definite kernels, which can undermine convexity. Despite this, LIBSVM can achieve convergence for indefinite kernels, such as the sigmoidal kernel. A geometric interpretation by [255] suggests that SVMs with such kernels do not maximize the margin but instead minimize distances between convex hulls in pseudo-Euclidean spaces, serving as optimal hyperplane classifiers.

7. Least Squares SVMs

LS-SVM [256,257] offers a significantly simplified approach to SVM training. Instead of the typical inequality constraints, LS-SVM introduces equality constraints, which enables the solution of a set of linear equations rather than solving a QP problem.

The parameters of the decision function $f(\mathbf{x})$, namely, α_p and b, are determined by solving the primal optimization problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{p=1}^{N} \xi_p^2 \tag{26}$$

subject to

$$y_p(\mathbf{w}^T\phi(\mathbf{x}_p) + b) = 1 - \xi_p, \quad p = 1, \ldots, N. \tag{27}$$

The Lagrangian for this formulation is:

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\sum_{p=1}^{N} \xi_p^2 - \sum_{p=1}^{N} \alpha_p \left[y_p(\mathbf{w}^T\phi(\mathbf{x}_p) + b) - 1 + \xi_p\right], \tag{28}$$

where α_p represents the Lagrange multipliers in $f(\mathbf{x})$. The necessary optimality conditions, derived using KKT, are:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Longrightarrow \mathbf{w} = \sum_{p=1}^{N} \alpha_p y_p \phi(\mathbf{x}_p), \tag{29}$$

$$\frac{\partial L}{\partial b} = 0 \Longrightarrow \sum_{p=1}^{N} \alpha_p y_p = 0, \tag{30}$$

$$\frac{\partial L}{\partial \xi_p} = 0 \Longrightarrow \alpha_p = C\xi_p, \quad p = 1, \ldots, N, \tag{31}$$

$$\frac{\partial L}{\partial \alpha_p} = 0 \Longrightarrow y_p(\mathbf{w}^T\phi(\mathbf{x}_p)) + b - 1 + \xi_p = 0, \quad p = 1, \ldots, N. \tag{32}$$

By substituting $\mathbf{w}$ and ξ_p, we obtain:

$$\begin{bmatrix} 0 & -\mathbf{y}^T \\ \mathbf{y} & \mathbf{Q} + C^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \tag{33}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_N)^T$, $\mathbf{Q} = [q_{ij}]_{N\times N}$, $q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{1} = (1, 1, \ldots, 1)^T$, and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_N)^T$.

Solving (33) yields:

$$b = \frac{\mathbf{y}^T\mathbf{A}^{-1}\mathbf{1}}{\mathbf{y}^T\mathbf{A}^{-1}\mathbf{y}}, \quad \boldsymbol{\alpha} = \mathbf{A}^{-1}(\mathbf{1} - b\mathbf{y}), \tag{34}$$

where $\mathbf{A} = \mathbf{Q} + C^{-1}\mathbf{I}$. The LS-SVM output for classification tasks is derived from (24).

LS-SVM has a computational complexity of $O(N^3)$ for a dataset of N samples, and its solution lacks sparsity. Although some fast algorithms for LS-SVM, including the CG method [258,259], SMO algorithm [260], and coordinate-descent algorithm [261], achieve reduced complexity, they still yield non-sparse solutions.

To address this, a rapid sparse approximation for LS-SVM [262] employs a greedy approach that incrementally constructs the decision function by incorporating one basis function at a time from a kernel-based dictionary, guided by an ε-insensitive criterion for stopping. Additionally, a probabilistic variant enhances the speed of the algorithm through a probabilistic acceleration method.

The primal space LS-SVM, derived from the representer theorem, also solves the linear equation set $\mathbf{A}\boldsymbol{\alpha} = \mathbf{h}$ in the primal space. However, if the corresponding kernel matrix is low rank, the symmetric coefficient matrix $\mathbf{A}$ becomes singular, potentially leading to multiple solutions, with some exhibiting sparsity.

Weighted LS-SVM [263] introduces weights for error variables to mitigate biased estimations from LS-SVM and to enhance robustness against noisy data. Initially, LS-SVM is trained on the samples, after which weights for each sample are computed based on their error variables, followed by a retraining of the weighted LS-SVM.

MatLSSVM [264] is specifically designed for matrix patterns, such as images. This method effectively reduces the memory requirement for the weight vector from $d_1 d_2$ to $d_1 + d_2$, where $d_1 \times d_2$ represents the matrix dimensions. Similar to LS-SVM, MatLSSVM encounters regions that cannot be classified in multiclass applications. However, a fuzzy adaptation of MatLSSVM [264] effectively addresses these unclassifiable regions.

The iterative reweighted LS-SVM [265] addresses a series of weighted LS problems that converge to the actual SVM solution and this approach is also suitable for regression tasks. Moreover, it is equivalent to the C-loss kernel classifier that incorporates a Tikhonov regularization term [266], elucidating its robustness from the perspectives of correntropy and density estimation.

Standard SVM [267] and LS-SVM [268] are closely related to the MAP solution of a Gaussian process classifier. In [268], a Bayesian inference framework is introduced to address model bias and weights within the primal space of LS-SVM, along with a procedure for estimating the hyperparameters of the model.

8. SVM Training Methods

8.1. SVM Algorithms with Reduced Kernel Matrix

To tackle the challenges of time and space complexity, low-rank approximations of the kernel (Gram) matrix are typically implemented through various techniques, including the Nystrom method [159], greedy approximation [269], and matrix decomposition [221]. The Gram matrix itself is structured as an $N \times N$ square matrix consisting of kernel products.

In [269], the dataset within the feature space is approximated by leveraging a lower-dimensional subspace formed from a randomly selected subset of data points. The remaining points are expressed as linear combinations of the basis vectors. This basis is iteratively constructed, with each new candidate chosen using a greedy approach aimed at minimizing the bound of the approximation error. The solver for the QP subproblems utilized in this context is loqo, an interior-point method included in the SVMlight package version 6.01.

The reduced SVM for binary classification [270] is formulated based on generalized SVM [271] and smooth SVM [272]. This approach employs a unique SVM cost function, whereby a random segment of the dataset is used to generate a thin rectangular kernel matrix that substitutes the complete kernel matrix in the nonlinear SVM formulation. As there are no constraints, a quadratically converging Newton algorithm can be employed for training, yielding a time complexity of $O(N)$. While reduced SVM may experience higher training errors compared to SVM, it still exhibits comparable generalization performance [273]. In fact, on certain smaller datasets, reduced SVM can surpass the performance of standard SVM. The concept of a reduced kernel matrix has found applications across numerous other kernel-based learning algorithms, including LS-SVM [256], proximal SVM [274], Lagrangian SVM [275], active-set SVR [276], and ε-smooth SVR [273].

Additionally, a direct method [277] that shares similarities with reduced SVM seeks to develop sparse kernel learning algorithms by incorporating an extra constraint into the convex optimization framework.

Proximal SVM [274] mirrors the LS-SVM approach by substituting inequality constraints with equality constraints in its defining structure and applying the LS concept, which changes the error measure from absolute to squared.

Lagrangian SVM [275] offers a rapid and highly efficient iterative algorithm capable of classifying datasets containing millions of instances. This algorithm distinguishes itself by not relying on traditional optimization techniques like LP or QP solvers, ensuring guaranteed convergence. While Lagrangian SVM can accommodate any positive semi-definite kernel, it is restricted to intermediate-sized problems due to its reliance on inverting a single $N \times N$ Hessian matrix from the dual formulation.

8.2. GEPSVM

SVM can encounter challenges in specific scenarios, such as the XOR problem [278]. Generalized eigenvalue proximal SVM (GEPSVM) [279] addresses this by relaxing the parallelism condition inherent in proximal SVM, aiming to derive two optimal nonparallel proximal hyperplanes. This method classifies data points by associating them with the nearest of these two hyperplanes, which are established through generalized eigenvalue problems. The objective of GEPSVM is to identify two nonparallel hyperplanes, each optimally situated near the data points of its respective class while maximizing the distance from the opposing class. However, the reliance of GEPSVM on solving two generalized eigenvalue problems can lead to instability in the final solution.

In contrast to GEPSVM, the multiple view SVM (MVSVM) [280] focuses on determining two optimal projection directions for weight vectors. Here, the objective is to maximize the distance between distinct classes within the projected space while ensuring that points from the same class remain as close as possible.

GEPSVM constrains each class to a single fitting hyperplane, but the work by [281] expands this framework to accommodate multiple hyperplanes per class. This study introduces a pivotal transformation for the optimization problem within GEPSVM, leading to the multiplane convex proximal SVM (MCPSVM). In this approach, a collection of hyperplanes, influenced by the features of the data, is learned for each class, utilizing a strictly geodesically convex objective that leads to a more refined closed-form solution.

8.3. Twin SVM

Twin SVM [282], derived from GEPSVM [279], enhances stability by formulating two hyperplanes that closely correspond to their respective classes while maximizing the distance to the other class. By solving two smaller QP problems, twin SVM is effective for nonparallel datasets and leverages global within-class clustering for improved accuracy, ensuring a unit distance between classes. It operates approximately four times faster than standard SVM and maintains similar or superior generalization capabilities relative to both SVM and GEPSVM.

Various enhanced algorithms have emerged from twin SVM, including nonparallel-plane proximal classifier (NPPC) [283], twin bound SVM [284], improved twin SVM [285], twin support vector hypersphere (TSVH) [286], nonparallel support vector classifiers using various loss functions [287], and twin SVM with pinball loss (pin-twin SVM) [288]. The coordinate-descent margin-based twin SVM [289] processes one point at a time, enabling the rapid training of large datasets without full data loading. The twin bounded SVM [284] introduces a regularization term to minimize structural risk.

The safe screening rule [290] effectively removes redundant features in L_1 sparse contexts, as adapted for twin SVM in [291] to identify and eliminate zero and one components in the dual problem.

Despite its advantages, twin SVM lacks defined geometric properties due to its L_1-norm penalty for slack variables. The ENNHSVM [292] addresses this by integrating an elastic net penalty and constructing two nonparallel hyperplanes. Additionally, a safe screening rule for ENNHSVM has been designed to expedite calculations.

The TSVH classifier [286] resolves inconsistencies in traditional nonparallel SVM, which independently solves two QP problems and ignores new sample points during training. Nonparallel hyperplane SVM (NHSVM) [293] improves this by constructing two nonparallel hyperplanes simultaneously.

The lack of sparsity of twin SVM complicates matrix inversions for large datasets and leads to noise sensitivity due to hinge loss. In nonlinear cases, the authors of [294] indicates a need for reformulating optimization problems with kernel surfaces [295]. A pinball loss function $L_\tau(u)$ [294] mitigates noise sensitivity and enhances stability but can reduce sparsity, prompting the development of the ϵ-insensitive zone pinball loss $L_\tau^\epsilon(u)$ SVM [294].

Nonparallel SVM [296] enhances twin SVM by applying the ϵ-insensitive loss function, avoiding matrix inversions and kernel considerations, with increased ϵ corresponding to higher sparsity.

Standard SVM with hinge loss $L_{\text{hinge}}(u) = \max(0, u)$ offers sparse solutions but is sensitive to noise. The pinball loss [294], defined as $L_\tau(u) = \max(u, -\tau u)$ for $u \in \mathbb{R}$ and $0 \leq \tau \leq 1$, penalizes correctly classified points, allowing pinball loss SVM (pin-SVM) [294] to maximize the quantile distance between classes. While convex and robust, pin-SVM sacrifices sparsity as its sub-gradient is almost never zero.

To enhance sparsity, the ϵ-insensitive pinball loss [294] is defined as $L_\tau^\epsilon(u) = \max(u - \epsilon, 0, -\tau u - \epsilon)$, resembling the ϵ-insensitive loss in SVM regression. This truncated version flattens the negative portion to improve sparsity. Truncated pinball loss SVM ($\overline{\text{pin}}$-SVM) [297] employs coordinate-coupled convex programming to address non-convexity.

Both pin-twin SVM and LS-pinball solve pairs of smaller QP problems, leading to faster training than pin-SVM. Tests in [288] show that pin-twin SVM's training time is nearly equivalent to taht of twin SVM, despite the added pinball loss. The time complexity relationship is: $O(\text{nonparallel SVM}) > O(\text{pin-SVM}) > O(\text{SVM}) > O(\text{LS-pinball}) > O(\text{twin SVM}) \approx O(\text{pin-twin SVM})$.

Twin SVM's need for matrix inversion in the Wolfe-dual formulation and its focus on minimizing empirical risk, which can both lead to overfitting. Pin-SVM [294] solves a single large QP problem, limiting its scalability. Large-scale pinball twin SVM (LPTWSVM) [298] addresses these issues by using pinball loss for better noise insensitivity and reformulating the dual problem to eliminate matrix inversion. LPTWSVM employs the kernel trick for nonlinear cases, minimizing structural risk and enhancing classification accuracy.

A possibilistic classification algorithm using SVMs [299] aims to find a maximal-margin fuzzy hyperplane through fuzzy optimization, providing robustness against outliers. This method introduces a vagueness parameter v to control support vector and error bounds. The entropy-based fuzzy least squares twin SVM [300] computes membership based on entropy values. Unicentric fuzzy twin SVMs assign membership based on proximity to centers, potentially confusing fringe support vectors with outliers. Polycentric intuitionistic fuzzy weighted least squares twin SVMs (PIFW-LSTSVM) [301] assign both membership and non-membership simultaneously, reducing outlier and noise impact while enhancing generalization.

Least-squares (LS) twin SVM achieves faster training by solving linear equations throughout. Large-scale fuzzy LS twin SVM [302] avoids matrix inversion, further speeding up training. Intuitionistic fuzzy weighted twin SVM [303] reduces outlier influence, while [304] uses belief function theory for outlier detection, and [305] employes confidence functions along with rough set theory to identify boundary samples and reduce noise.

A regression-based SVM hyperparameter learning method [306] reframes SVM as piecewise linear regression, addressing hyperparameter learning, prediction probability, and model uncertainty. It introduces model entropy, transforming leave-one-out prediction probabilities into an entropy measure that quantifies uncertainty, considering the distribution of all samples for a more accurate evaluation compared to focusing solely on support vectors.

8.4. ν-SVM

The ν-SVM [307] uses the parameter ν to regulate the number of support vectors and errors, adding complexity to its formulation compared to standard SVM. It can be viewed as a nearest-point problem within reduced convex hulls, with a competitive decomposition method [242]. In ν-SVM with the L_p norm, the solution $\mathbf{w}_p$ is closely related to $\mathbf{w}_2$, with minimal dependency on p [308], applicable to SVR as well.

The par-ν-SVM [309] enhances ν-SVM by employing a parametric-insensitive loss function that adapts the zone to the data. Extended ν-SVM [310] broadens the range of ν for diverse decision functions and can be solved using the RapMinos algorithm [310].

Alternatives to the expected misclassification cost include the minimax and Neyman-Pearson criteria [311]. The 2C-SVM [312] and 2ν-SVM [313] are cost-sensitive extensions, with the former shown to be equivalent to the latter [311].

The ν-twin SVM [314] integrates ν-SVM and twin SVM, adjusting the unit distance to ρ. The Iν-twin SVM [315] identifies two nonparallel hyperplanes, ensuring they remain at least ρ apart. Fast Iν-twin SVM improves efficiency by converting the smaller QP problem into a solvable unimodal function, outperforming twin bounded SVM [284] and ν-twin SVM in speed while maintaining similar generalization ability.

8.5. Cutting-Plane Technique

The cutting-plane algorithm [316] iteratively approximates the risk term in problem (14) using cutting planes, facilitating the addition of basis vectors beyond the training set. A similar method in [277] trains SVMs with kernels that utilize arbitrary basis vectors, not limited to training support vectors. These approaches achieve an ε-accurate solution to the regularized risk minimization problem in $O(1/\varepsilon\lambda)$ iterations, where λ balances regularization and loss [317].

For large-scale L_1-SVM, SVMperf [318] employs a cutting-plane strategy to solve (17) by progressively refining a piecewise linear approximation. An optimized cutting-plane algorithm for linear binary SVM [319] improves efficiency by performing a line search for optimization. This method converges to an ε-accurate solution within $O(N)$ iterations, outperforming SVMlight and SVMperf by over 1200 times and 29 times, respectively, while maintaining precise support vector solutions. This algorithm typically converges faster than gradient descent and Pegasos.

In the L_1-slack reformulation of training linear SVM, the cutting-plane method achgieves a time complexity of $O(N)$ [320], with iterations linear in precision and the regularization parameter, including SVMperf's training algorithm for linear two-class SVMs. It considers both individual data points and their linear combinations as candidate support vectors, resulting in fewer nonzero dual variables and smaller cutting-plane models.

The cutting-plane subspace pursuit method [321] builds the basis set iteratively, similar to basis pursuit methods [45], yielding orders of magnitude sparser classification rules compared to conventional support vector representations for similar accuracy. Its sparse solutions outperform those from Nystrom method [159], incomplete Cholesky factorization [221], core vector machines, LASVM with margin-based active selection and finishing [322], and ball vector machines [323]. Notably, both Nystrom and Cholesky factorization methods are implemented in SVMperf.

8.6. Gradient-Based Methods

Successive overrelaxation [324] is a coordinate-descent approach for training SVMs on large datasets, updating one variable per iteration. It is effective for symmetric linear complementarity problems and QP problems, converging linearly to a solution. On smaller problems, it outperforms SVMlight and matches or exceeds the speed of SMO. Furthermore, LIBLINEAR [325] incorporates successive overrelaxation into the dual coordinate-descent method specifically designed for large-scale linear SVMs.

For L_2-SVM, a coordinate-descent method [326] focuses on updating one variable while fixing others, utilizing a modified Newton method combined with a line-search technique akin to the trust-region method [327]. This method guarantees a strict reduction in function value under convexity conditions and employs full Newton steps when feasible, resulting in faster convergence and greater stability compared to Pegasos and the trust-region Newton method, with proven global convergence to the unique minimum at a linear rate.

The oLBFGS algorithm [328] is a fast second-order learning method, but selecting its global learning gain is challenging [329]. Stochastic gradient-descent quasi-Newton (SGD-QN) algorithm [330] and corrected SGD-QN [329] utilize second-order information in a stochastic gradient-descent framework for linear SVMs, independently scheduling parameter updates. They estimate a diagonal rescaling matrix based on oLBFGS, iterating as fast as first-order methods while requiring fewer iterations for the same level of accuracy. Corrected SGD-QN adapts to skewed feature distributions and sparse datasets.

Modified Newton methods for training large-scale L_2-SVM are discussed in [331,332], minimizing a single-variable piecewise quadratic function (18) that is differentiable but lacks twice differentiability. The generalized Hessian matrix is used to determine the Newton direction. The trust-region Newton method [333] provides an efficient implementation specifically for L_2-SVM.

8.7. Training SVM in the Primal Formulation

Duality theory simplifies constraint handling, allowing the dual optimization problem to be expressed using dot products and kernel functions. While the primal QP problem can be large, its Wolfe dual formulation is comparatively smaller. Typically, primal optimization is framed as an unconstrained problem using the representer theorem, often involving a two-stage process: first approximating the dual QP solution and then mapping it to the primal. Primal optimization is superior in both solution quality and time complexity as it directly minimizes the primal objective function [334], with a simpler implementation that requires no optimization libraries.

Many machine learning algorithms can be formulated as the unconstrained regularized risk minimization problem (14), where $\mathbf{w} \in \mathbb{R}^n$ is the parameter vector, $\frac{1}{2}\|\mathbf{w}\|^2$ serves as the quadratic regularization term, $C > 0$ is a constant value, and the second term represents a convex risk function that approximates empirical risk. This primal formulation proves to be efficient for large N with moderate or sparse input dimensions.

The finite Newton method [332] is a direct primal algorithm designed for linear SVMs with L_2 loss, leveraging sparsity. Modifications in [331] utilize CG techniques for Newton iterations, significantly speeding up the process compared to decomposition methods like

SVMlight and SMO in the case of a large volume of examples. Primal optimization is generally more efficient than dual optimization for linear SVMs [331].

For nonlinear SVMs, smooth SVM [272] implements primal optimization, matching or surpassing the performance of SVMlight, successive overrelaxation [324], and SMO on larger datasets. The recursive finite Newton method [334] solves the primal formulation efficiently for both linear and nonlinear SVMs, maintaining a computational complexity that aligns with that of dual optimization. Algorithms in [335] that allow for an accuracy ε_p for the primal QP guarantee approximate solutions within low-order polynomial time.

An alternative strategy for primal optimization is to decompose the kernel matrix for linearization. Pegasos [336] and SGD (http://leon.bottou.org/projects/sgd, accessed on 1 November 2024) [337] are efficient solvers for linear SVMs. Pegasos is a primal estimator for L_1-SVM, alternating between stochastic gradient-descent and projection steps, outperforming SVMperf.

In [338], a rapid accelerated proximal gradient method is introduced for unified classification models, including SVMs. It employs backtracking line search along with adaptive restarting to enhance convergence speed, often being substantially faster than LIBSVM and LIBLINEAR [325], especially for datasets whose feature dimension $n > 2000$.

8.8. LapSVM for Semi-Supervised Learning

Manifold regularization techniques operate under the premise that data points lie on a low-dimensional manifold. A graph representation is employed to approximate this manifold, where neighboring points connected by strong edges are likely to share the same labels, allowing for label propagation across the graph.

Laplacian SVM (LapSVM) [339] is a semi-supervised SVM (S^{3}VM) that utilizes the graph Laplacian in its framework, incorporating a hinge loss function similar to traditional SVMs.

The primal Laplacian SVM (PLapSVM) can be solved by PLapSVM-Newton and PLapSVM-PCG [340]. By implementing preconditioned CG methods, the training process is expedited using an early stopping criterion based on predictions from either unlabeled or labeled validation datasets. This approach significantly reduces the computational complexity from $O(N^3)$ to $O(kN^2)$, where N denotes the total count of labeled and unlabeled samples and $k \ll N$.

In contrast to LapSVM and PLapSVM, fast Laplacian SVM (FLapSVM) [341] solves a dual optimization problem that aligns with the formulation of conventional SVMs, allowing for the application of the kernel trick. The solution for FLapSVM can be achieved efficiently through the successive overrelaxation method [324], which demonstrates linear convergence. By integrating the random scheduling of subproblems and employing two stopping criteria, FLapSVM offers faster training times and improved average accuracy compared to both LapSVM and PLapSVM.

8.9. Clustering-Based SVM

Clustering-based SVM [342] enhances the efficiency of SVMs on large datasets with constrained resources by using the BIRCH algorithm to create detailed descriptions near the classification boundary and coarser ones farther away. It has a training complexity of $O(n^2)$, where n represents the number of support vectors, making it particularly effective for classifying large datasets with low dimensions, particularly when handling irregular patterns.

Bit-reduction SVM [343] accelerates training and prediction by grouping similar examples and reducing their resolution. Each bin receives a weight according to the number of examples belonging to each class, creating a set of weighted examples for SVM training. Optimal compression parameters are computed once and reused for incremental data, allowing bit-reduction SVM to achieve greater accuracy compared to random sampling if the data are not excessively compressed.

Multi-prototype SVM [344] enhances multiclass SVM by employing multiple prototypes per class, allowing for the combination of several vectors form decision functions with large margins. This approach transforms a non-convex problem into simpler convex ones, outperforming LVQ. The method in [345] integrates SVM with a fast nearest-neighbor condensation rule, reducing the training time by one or two orders of magnitude on large, multidimensional datasets while decreasing the number of support vectors by half, with minimal accuracy loss.

8.10. Other SVM Methods

A rapid SMO method [346] addresses the dual optimization problem in potential SVM by iterating to optimize the Lagrangian concerning one (single SMO) or two (dual SMO) Lagrange multipliers, while fixing the others. Potential SVM can utilize dual SMO, block optimization, and ε-annealing, making it suitable for arbitrary dyadic datasets, which involve relationships between objects. Its computation time is typically on par with or slightly exceeds that of standard SVM methods, while yielding much fewer support vectors for similar generalization performance.

The LASVM algorithm [322] employs SMO to facilitate efficient online and active learning, selectively training on a subset of the data through active selection. As the algorithm iterates through multiple epochs, it converges to the precise solution of the SVM [322], employing the L_1-norm of the slack variables. Improvements in learning speed, accuracy, and sparsity are achieved by substituting the standard working-set selection in SMO with a second-order approach that aims to maximize progress in each iteration [347].

The decision tree SVM method [348] decomposes the data space using a decision tree and trains SVMs on these regions. This method accelerates training by thousands of times for datasets manageable by traditional kernel-based SVMs, while maintaining similar test accuracy.

SVM with Automatic Confidence (SVMAC) [349] computes the confidence label for each training sample, enhancing SVM training by incorporating these values into the QP formulation. SVMAC achieves better generalization performance than standard SVMs, with the primary added cost being the construction of a decision boundary γ for confidence labeling.

Hybrid kernel SVMs [350] are targeted at minimizing the upper bound of the VC dimension, employing a flexible kernel function from common Mercer kernels to realize an SRM and achieve better generalization on test data.

DirectSVM [351] is a simple learning algorithm that identifies support vectors based on the closest training points of opposite classes, provided the points are linearly independent. For soft-margin SVMs with quadratic penalties, DirectSVM reaches a maximal margin hyperplane within $M-2$ iterations, with M being the count of support vectors. It achieves generalization performance on par with other SVMs but with greater speed than QP methods.

Time-adaptive SVM [352] creates classifiers that adapt to changing concepts using a sequence of models over short time windows, penalizing diversity in the cost function to a couple of them. It requires matrix inversion and pseudo-inversion, but an improved version [353] shares a common vector among classifiers, avoiding direct inversion. This leads to an efficient implementation based on the MEB problem using core vector machine techniques [240], achieving asymptotic linear time complexity for large, evolving datasets.

Sparse support vector classification [354] generates sparse solutions by applying L_0-norm regularization to eliminate irrelevant parameters, using an iteratively reweighted learning algorithm. This method has a hierarchical-Bayes interpretation. Meanwhile, the set covering machine [355] seeks to create the sparsest possible classifier with minimal errors on the training set.

Techniques such as core vector machines, LASVM with margin-based active selection and finishing [322], and ball vector machines [323] employ a greedy approach to selectively pick basis vectors exclusively from the training set.

Max-min margin machine [356] enhances SVM by utilizing the Mahalanobis distance to incorporate class structures into the decision boundary.

Support feature machine [357] minimizes the L_0-norm of the separating hyperplane to identify the smallest subspace where two classes are linearly separable. This method effectively handles unbalanced and nonseparable data, making it ideal for feature selection in high-dimensional datasets with limited sample sizes.

Approximate extreme points SVM [358] optimizes SVM by utilizing a chosen subset called the representative set, which is obtained through a linear-time algorithm based on convex hulls in kernel space. This method trains significantly faster than LIBSVM, core vector machines [240], LASVM [322], ball vector machines [323], SVMperf [321], and the random features method [160], while achieving similar classification accuracy and competitive speed.

Field-SVM [359] simultaneously trains and predicts for groups of patterns, learning both the classifier and the style normalization transformation. By leveraging the style consistency within each group, field-SVM improves the classification accuracy significantly.

An SVM framework designed for regression and quaternary classification of complex data [360] utilizes complex kernels and complexified real kernels, effectively addressing two distinct real SVM tasks with an induced real kernel.

In [361], the methodology for L_2-SVM is extended to L_p-norms with $p > 1$ (i.e., L_p-SVM), deriving second-order cone formulations for both dual and primal problems by defining multidimensional kernels. This approach reformulates dual problems in a transformed space of the original data, where dependence appears as homogeneous polynomials.

The L_1-norm nonparallel SVM [362] reformulates the GEPSVM objective function and solves two L_1-norm problems. In a robust twin support vector algorithm for clustering [363], L_1-norm distance is used to reduce the computational time by solving systems of linear equations rather than QP problems. The L_1-norm is effective in mitigating outlier effects [4,45].

In [171], kernel-mapped features are precomputed for training a linear SVM, applicable to specific kernels like string and low-degree polynomial kernels when the kernel space dimension is manageable. Low-rank linearized SVM [364] enhances kernel SVM efficiency by transforming a nonlinear SVM into a linear one using an approximate empirical kernel map derived from low-rank decompositions.

UniSVM [365] trains various SVM models in a unified framework. Most common losses in SVM can be expressed as a least squares type of difference of convex loss (LS-DC) or approximated by LS-DC [365]. UniSVM utilizes the difference of convex algorithm to solve SVM models with both convex and nonconvex LS-DC, yielding a closed-form solution for each iteration and applying low-rank approximation for large-scale nonlinear problems.

Tropical SVM [366] classifies data employing a tropical hyperplane within the tropical metric framework, utilizing max-plus algebra. While generalization error bounds based on VC dimensions depend on dimension, tropical SVM exhibits resilience to the curse of dimensionality [366].

Multi-view learning leverages consensus and complementarity, encompassing co-training, co-regularization, and margin-consistency algorithms. SVM-2K [367] merges two SVM models from distinct perspectives with similarity constraints for consistency. This approach maximizes the potential data information. In [368], safe screening rules analyze dual variables and samples for SVM-2K and multi-view twin SVM, showing the reduced optimization problem retains the original problem's solution.

Relevance vector machine (RVM) [369] operates within a Bayesian automatic relevance determination framework, generating a limited number of basis functions (relevance vectors) with nonzero weights through hierarchical priors. An accelerated RVM strategy maximizes marginal likelihood by efficiently adding and deleting candidate basis functions [370]. Unlike SVM, RVM does not maintain the principle that positive samples should have positive weights, leading to instability and sensitivity to kernel parameters in classification tasks.

Probabilistic Classification Vector Machine (PCVM) [371,372] combines features of SVM and RVM to provide a sparse Bayesian classification solution. It ensures consistency between weights and class labels by using a truncated Gaussian prior on the weights. Multiclass PCVM [373] employs a top-down algorithm for MAP point estimates, leveraging an expectation-maximization method, along with a bottom-up incremental approach designed to maximize the marginal likelihood.

Quantum SVM [374] offers exponential speedup for LS-SVM, with a time complexity that is polynomial in the logarithm of data sizes. The quantum-inspired LS-SVM algorithm [375] is a dequantization of the quantum SVM, achieving similar exponential speedup over classical SVM. It enables classification with any success probability in logarithmic time concerning both the data dimension and the total number of points, using an improved indirect sampling technique for matrices characterized by low rank, low condition number, and high dimensionality.

9. Pruning SVMs

Conventional convex SVM solvers use hinge loss, which allows outliers to influence the model, resulting in the number of support vectors n that grows with the number of training examples [25]. As a result, the computational complexity for predicting new examples is $O(n)$, making it desirable to reduce n. Both ν-SVM [307] and sparse SVMs [376] achieve this while ensuring that their hidden-layer weights are selected from a subset of the data.

The non-convex ramp loss function addresses the scalability issues of convex SVM solvers by facilitating constrained concave–convex procedure (CCCP) optimization and preventing outliers from becoming support vectors. This leads to online learning frameworks for LASVM variants [377], yielding sparser models and fewer discarded instances without compromising generalization performance compared to LASVM [322].

To reduce support vectors post-SVM training, methods like L_1 regularization on the bias b have been proposed [38]. Pruning based on linear dependence is introduced in [217] and refined in [378] to eliminate linearly dependent support vectors after model training. In regression, initial support vectors are produced with SVMTorch [228], followed by the elimination of linearly dependent vectors.

The pruning algorithm involves constructing a new kernel row space $\mathcal{K}$, which is related to feature space $\mathcal{H}$ [379]. By examining the kernel output overlap, a systematic pruning method ensures at most M support vectors in an M-dimensional space, reducing the upper bound from $M+1$ [380] to M while maintaining the separating hyperplane. This method eliminates the necessity of identifying support vectors in the feature space as typically required by SVMs. In [381], the approach from [217,378,379] is extended by replacing linear dependence with orthogonal projection via an LS approximation within the space $\mathcal{K}$, allowing for the batch pruning of support vectors through clustering.

To address excessive support vectors, a primal method [376] decouples basis functions from support vectors, efficiently finding a set of kernel basis functions with a maximum size of $d_{\max}$, roughly scaling as $O(Nd_{\max}^2)$. This method progressively discovers basis functions to enhance accuracy, often yielding classifiers with significantly fewer basis functions than standard SVMs while maintaining similar accuracy.

Even minor reductions in support vectors can severely impact generalization performance. In certain cases, the support vector set from SVM may not be minimal [382]. A reduced set approximation can provide a better approximation of the decision surface using vectors not originally classified as support vectors. For example, in [383] the reduction process iteratively replaces a pair of closest support vectors of the same class with a newly constructed vector.

A neighborhood-based pattern selection algorithm [384] identifies patterns that are close to the decision boundary, leveraging entropy concepts to measure the diversity of class labels among the k nearest neighbors for better proximity estimation.

Based on accurate primal and dual optimal estimations, the approach in [385] simultaneously identifies and removes irrelevant inactive features and samples during training.

Pruning LS-SVMs

LS-SVM lacks sparsity compared to SVMs. Sparse LS-SVM [257] removes training examples with the smallest α_i, which is proportional to the training error e_i. This method recursively prunes support vectors that are farthest from the decision boundary until a decline in performance is observed. A comprehensive method for sparse LS-SVM is proposed in [263], refined in [386] by minimizing output error and selecting support vectors with the least deviation, although this approach can lead to singular matrix inversion without regularization. A regularized pruning method using a selective window algorithm for efficiency is introduced in [387] to eliminate this nonsingularity. In [388], pruning uses an SMO-based pruning method. Pruning methods in [386–388] require solving linear equations for each sample, increasing computational costs.

An adaptive pruning algorithm that avoids solving the primal non-sparse LS-SVM is proposed in [389], offering faster training compared to the SMO method for large-scale problems. Compressive sampling is also applied to derive sparse LS-SVM [390].

Fixed-size LS-SVM [256] quickly finds sparse approximate solutions in the primal space using the Nystrom method. Two algorithms for primal space LS-SVM introduced in [391] find the sparse solution using Cholesky factorization of the kernel matrix with an iterative strategy. In [392], LS-SVM and primal fixed-size LS-SVM models are sparsified iteratively through L_0-norm-based reductions.

Low-rank modifications to LS-SVM [393] enhance variable selection by employing recursive feature elimination (RFE), which assists in identifying the most effective subset r of input dimensions for achieving optimal margins between classes. This approach utilizes a closed-form estimator for leave-one-out error.

Global-representation-based sparse LS-SVM [394] is a noniterative algorithm that constructs a reserved support vector set from globally representative points, achieving a computational complexity of $O(N^2)$ and a storage of $O(N)$, thereby accommodating large datasets effectively.

10. Multiclass SVMs

To tackle multiclass classification using SVM, two primary approaches are used: addressing all classes in a single optimization problem or combining multiple binary classifiers.

In the single optimization approach, multiclass SVMs are formed using the single-machine technique [395,396], resulting in a large optimization problem framed as a constrained quadratic objective. A fixed-point algorithm that ensures convergence for solving this issue is described in [396]. Additionally, multiclass smooth SVM is addressed via an efficient Newton–Armijo algorithm that achieves global convergence to a unique solution in quadratic time [272].

Decomposition strategies include one-against-all (or one-versus-rest) [217,395,397], one-against-one [398], all-against-all [398], and error-correcting output codes (ECOCs) [399,400]. Comparative analyses are presented in [397,401], with one-against-one and one-against-all often favored for their simplicity and reduced computational cost.

SVMs utilizing a binary tree framework [402] minimize the number of classifiers while speeding up decision-making, training $m - 1$ classifiers and requiring $\log_2 m$ tests to reach the final classification outcome. To optimize classifier quality, 2^m group evaluations are needed per node. The binary tree SVM method [403] improves classification efficiency, with a maximum of $m - 1$ binary classifiers during training and an average of $\log_{4/3}((m+3)/4)$ binary tests for decision-making. This approach significantly outperforms directed acyclic graph SVM (DAG-SVM) [398] and ECOC in scenarios with a high number of classes, achieving an average convergent efficiency of $\log_2((m+3)/4)$ while maintaining comparable accuracy.

To address unclassifiable areas within the one-against-all approach using decision tree-based SVMs, a total of $m - 1$ SVMs are trained, with each ith SVM distinguishing class i from classes $i + 1$ to m. Classification proceeds sequentially through these SVMs until a definitive class is assigned.

Figure 4 illustrates class boundaries for four distinct classes using linear kernels, where classes having fewer instances occupy larger regions, impacting the generalization ability. Each node of the decision tree is responsible for separating one set of classes from another.

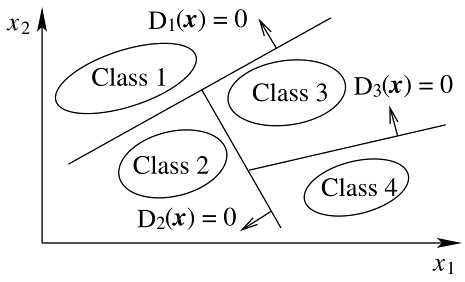
x_2
$D_1(\boldsymbol{x}) = 0$
Class 1
Class 3
$D_3(\boldsymbol{x}) = 0$
Class 2
Class 4
$D_2(\boldsymbol{x}) = 0$
x_1

Figure 4. Decision tree used for classification, featuring discriminant functions $D_i(\mathbf{x})$ that define the boundaries between classes.

Example 3. *Multiclass SVM classification is implemented using STPRtool (http://cmp.felk.cvut.cz/cmp/software/stprtool/, accessed on 10 Sepember 2024) through both one-against-all and one-against-one approaches, with the SMO binary solver utilized for training. A Gaussian kernel with $\sigma = 1$ is chosen, and C is set to 50. Figure 5 displays the training data along with the decision boundary.*

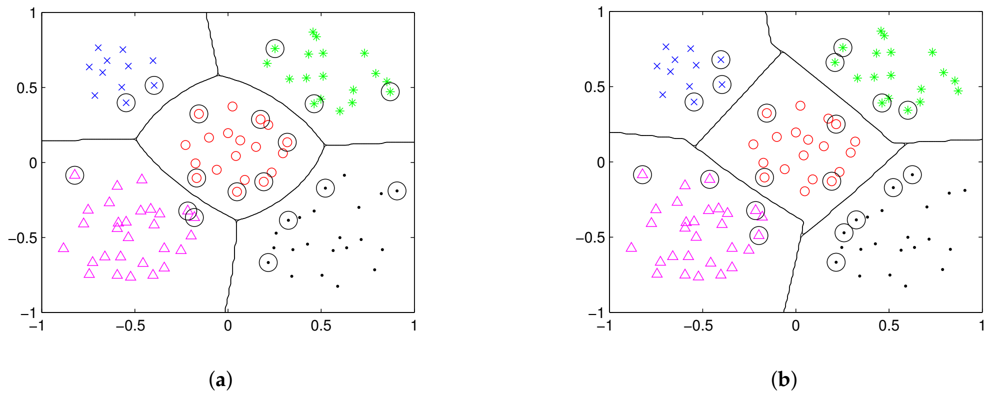

Figure 5. Multiclass SVM classification demonstrating: (**a**) the one-against-all approach; (**b**) the one-against-one approach. Points of different style belongs to different classes.

11. Support Vector Regression

SVM is adapted for regression through SVR, which maps data into a higher-dimensional feature space using a nonlinear function $F(\cdot)$ and subsequently performs linear regression there. It estimates a function for a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathbb{R}, i = 1, \ldots, N\}$. Figure 6 provides a visualization of SVR.

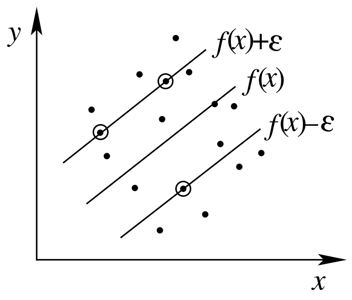

Figure 6. A depiction of hyperplanes within the two-dimensional feature space of SVR, illustrating how the objective function penalizes instances with y values that fall outside the bounds $(f(x) - \varepsilon, f(x) + \varepsilon)$. The instances indicated by circles represent support vectors that the margin seeks to push against.

A linear regression is determined by

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \tag{35}$$

that minimizes the regularized risk function defined by

$$\min R(C) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{p=1}^{N} \|y_p - f(\mathbf{x}_p)\|_{\varepsilon}, \tag{36}$$

where the empirical risk functional is characterized by the ε-insensitive loss function $\|\cdot\|_{\varepsilon}$ as

$$\|\mathbf{x}\|_{\varepsilon} = \max\{0, \|\mathbf{x}\| - \varepsilon\}, \tag{37}$$

with $\varepsilon > 0$ and a predetermined regularization constant $C > 0$. Alternative robust loss functions, like Huber's function, may also be applied. A data point $\mathbf{x}_p$ incurs no loss if it lies within the insensitive region, or ε-tube, defined by the condition $|y_p - f(\mathbf{x}_p)| \le \varepsilon$.

The optimization problem in (36) is reformulated as a QP problem by incorporating slack variables ξ_p and ζ_p [21]:

$$\min R(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\zeta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{p=1}^{N} (\xi_p + \zeta_p) \tag{38}$$

subject to

$$\left(\mathbf{w}^T\mathbf{x}_p + b\right) - y_p \le \varepsilon + \xi_p, \tag{39}$$

$$y_p - \left(\mathbf{w}^T\mathbf{x}_p + b\right) \le \varepsilon + \zeta_p, \tag{40}$$

$$\xi_p \ge 0, \quad \zeta_p \ge 0, \tag{41}$$

for $p = 1, \ldots, N$. If the error falls below the ε threshold, the slack variables ξ_p and ζ_p are zero. The expression $\frac{1}{2}\|\mathbf{w}\|^2$ measures the flatness of the function.

Kernel-based regression is achieved by replacing $\mathbf{x}$ with $\phi(\mathbf{x})$, thereby extending linear regression into the kernel space. A kernel function $k(\mathbf{x}, \mathbf{y}) = \phi^T(\mathbf{x})\phi(\mathbf{y})$, which meets Mercer's condition, is employed. Applying the Lagrange multiplier method, we formulate the following optimization problem [21,217]:

By substituting $\mathbf{x}$ with $\phi(\mathbf{x})$, linear regression extends to kernel-based regression in the kernel space. The kernel function, satisfying Mercer's condition, is defined as $k(\mathbf{x}, \mathbf{y}) = \phi^T(\mathbf{x})\phi(\mathbf{y})$. Using the Lagrange multiplier method, the optimization problem is formulated as [21,217]:

$$\begin{aligned} \min L(\boldsymbol{\alpha}, \boldsymbol{\beta}) \quad &= \quad \frac{1}{2}\sum_{p=1}^{N}\sum_{i=1}^{N}(\alpha_p - \beta_p)(\alpha_i - \beta_i)k\big(\mathbf{x}_p, \mathbf{x}_i\big) \\ &+ \quad \sum_{p=1}^{N}(\alpha_p - \beta_p)y_p + \sum_{p=1}^{N}(\alpha_p + \beta_p)\varepsilon \end{aligned} \tag{42}$$

subject to

$$\sum_{p=1}^{N}(\alpha_p - \beta_p) = 0, \tag{43}$$

$$0 \le \alpha_p, \beta_p \le C, \tag{44}$$

for $p = 1, 2, \ldots, N$, where the Lagrange multipliers α_p and β_p are associated with (39) and (40), respectively.

The output of the SVM produces the regression:

$$u(\mathbf{x}) = f(\mathbf{x}) = \sum_{p=1}^{N} (\beta_p - \alpha_p) k(\mathbf{x}_p, \mathbf{x}) + b, \tag{45}$$

where b is determined based on the boundary conditions. Vectors for which $\alpha_p - \beta_p \neq 0$ are identified as support vectors, highlighting the sparseness characteristic of SVR.

The ε-SVR model is expressed as a convex QP problem, requiring $O(N^2)$ in both memory and computational time. Additionally, the concept of LS-SVM has been adapted for application in SVR [184].

SVR demonstrates robustness thanks to its ε-insensitive loss function, where changes in ε affects the quantity of support vectors and the complexity of the model. Setting ε a priori reduces many weights $\alpha_p - \beta_p$ to zero, resulting in a sparse solution in (44). The choice of kernel is application-specific, and the SVR performance is sensitive to hyperparameter choices.

The least squares support vector regression (LS-SVR) model utilizing an RBF kernel shares theoretical similarities with MAP inference on Bayesian RBF networks that utilize a specific Gaussian prior for regression weights [404]. Additionally, LS-SVR is intricately linked to KRR [405], which can be viewed as the MAP solution derived from a Gaussian process model [406]. Moreover, similar connections are examined within regression contexts involving the SVR model [407].

Solving Support Vector Regression

Bounded influence support vector regression (SVR) [408] employs an adaptive weighting approach to downweight outliers in all regression variables, utilizing a robust scale estimator for large residuals.

In the work of [409], various leave-one-out bounds for SVR are established and compared to classification bounds, with the new bounds showing competitiveness with Bayesian SVR when it comes to parameter selection.

Regularization path algorithms [410] provide an efficient exploration of potential solutions with respect to regularization hyperparameters. The ε-path algorithm offers piecewise linearity and advantages over the λ-path algorithm [411], efficiently finding a sparse regression function. It begins with the tube width ε set to infinity, gradually reducing it to increase the number of support vectors.

By applying the smoothing technique introduced for smooth SVM [272], ε-smooth SVR [273] replaces the ε-insensitive loss function with a smooth approximation, allowing for unconstrained minimization via the Newton–Armijo method. It only requires solving a system of linear equations through iterative methods. In the linear case, it significantly outperforms LIBSVM and SVMlight in speed while having similar accuracy.

SVR is framed as a convex QP problem involving pairs of variables. Certain SMO algorithms leverage the connection between α_i and α_i^*. The approach presented in [111] selects two variable pairs in each iteration, solving the QP subproblem analytically. While the bias update method is inefficient, improvements in [412] enhance SMO for classification. Nodelib [413] improves upon SMO by simultaneously selecting α_i and α_i^*, transforming QP problems that involve $2l$ variables into nonsmooth optimization problems with l variables represented as $\beta_i = \alpha_i - \hat{\alpha}_i$, solvable via an SMO algorithm.

SVMTorch [228] is a decomposition algorithm designed for regression, akin to SVMlight for classification, and it includes a proof of convergence. It independently selects α_i and α_i^*, typically being significantly faster than Nodelib, with a training time scaling marginally below $O(N^2)$. It solves subproblems of size 2 analytically, and incorporates a caching mechanism for the kernel matrix, allowing the resolution of large problems without excessive memory use.

The global convergence of a general SMO algorithm for SVR is established in [414], proving optimal solutions can be reached in a finite number of iterations given certain conditions [414].

In line with twin SVM [282], twin SVR [415] generates two nonparallel ε-insensitive bounding functions—one for the lower bound and one for the upper bound—by solving two smaller QP problems rather than one large one, enhancing speed with comparable generalization. Primal twin-SVR [416] optimizes these QP problems directly in the primal space through sets of linear equations, improving learning speed without sacrificing generalization.

To address ε selection, ν-SVR [242,307] modifies ε-SVR to minimize ε automatically, ensuring no more than a specified fraction ν of the data points falls outside the defined tube. As a batch learning algorithm, ν-SVR allows dropping parameter C by relating it to other hyperparameters, specifically γ and ε, incorporating a decomposition algorithm akin to that of ν-SVM, implemented in LIBSVM.

Pairing ν-SVR [417] integrates the benefits of twin SVR and ε-SVR. In line with the twin SVR approach, it solves two smaller QP problems for faster learning than ε-SVR. It enhances the prediction speed and generalization by incorporating insensitive zones and regularization terms, with parameter ν regulating bounds on support vectors and errors.

SVR with parameterized lncosh loss [418] learns a loss function yielding a maximum likelihood optimal regression model for particular input-output data. This lncosh loss approach retains the beneficial properties of various loss functions, including Vapnik's, squared, and Huber's loss, adapting to the value of the parameter λ.

Unlike fixed global margins in standard SVR, localized SVR [419] flexibly adapts margins locally, functioning as a regression extension for the max-min margin machine [356]. The optimization process can be reformulated as a second-order cone programming problem, allowing for global optimality to be reached in polynomial time, with the option for kernelization.

In [420], a recursive finite Newton approach is introduced for nonlinear SVR in the primal form, showing similar performance to dual optimization techniques such as LIBSVM 2.82.

Support Vector Ordinal Regression

Support vector ordinal regression (SVOR) [421] extends binary SVM to ordinal regression, learning an optimal direction vector $\mathbf{w}$ along with $r-1$ thresholds, $b_1, \ldots, b_{r-1}$, which delineate $r-1$ parallel hyperplanes representing r distinct ranks. This approach was refined in [422] by enforcing ordinal constraints $b_1 \leq b_2 \leq \ldots \leq b_{r-1}$, yielding improved generalization using an SMO-type algorithm. The methods in [422] focus on optimizing multiple thresholds to create parallel hyperplanes, with problem size scaling as $O(N)$. Meanwhile, linear rankSVM [423] remains a widely used method for pairwise ranking tasks.

12. Support Vector Clustering

SVC [424] employs a Gaussian kernel to map data points into a high-dimensional feature space, where clustering is performed and then projected back into the original space. The goal is to find the smallest sphere with radius R that encapsulates all the N data points $\{\mathbf{x}_p\}$ within the feature space, described by minimizing:

$$E(R, \mathbf{c}, \boldsymbol{\xi}) = R^2 + C \sum_{p=1}^{N} \xi_p \tag{46}$$

subject to

$$\left\| \boldsymbol{\phi}(\mathbf{x}_p) - \mathbf{c} \right\|^2 \leq R^2 + \xi_p, \tag{47}$$

$$\xi_p \geq 0, \tag{48}$$

$p = 1, 2, \ldots, N$, where $\boldsymbol{\phi}(\cdot)$ transforms each data point into the feature space, ξ_p denotes the slack variable associated with the pth observation, $\mathbf{c}$ represents the center of the enclosing sphere, and C acts as a penalty constant to manage noise.

Using the Lagrange multiplier method, the problem is reformulated as:

$$\min E_{\mathrm{SVC}} = \sum_{p=1}^{N}\sum_{i=1}^{N} \alpha_p \alpha_i k(\mathbf{x}_p, \mathbf{x}_i) - \sum_{p=1}^{N} \alpha_p k(\mathbf{x}_p, \mathbf{x}_i) \tag{49}$$

subject to

$$\sum_{p=1}^{N} \alpha_p = 1, \tag{50}$$

$$0 \le \alpha_p \le C, \quad p = 1, 2, \ldots, N, \tag{51}$$

where $k(\cdot)$ represents the Gaussian kernel, and α_p denotes the Lagrange multiplier for the pth data point. The kernel width σ determines the cluster scale, while the slack variable ξ_p manages outliers and overlaps among clusters. By fine-tuning α_p and ξ_p, SVC aims to reduce the number of support vectors, thereby creating smooth, flexible cluster boundaries.

The distance from the input pattern in the mapped space to the center of the sphere can be expressed as

$$\begin{aligned} d^2(\mathbf{x}, \mathbf{c}) &= \|\boldsymbol{\phi}(\mathbf{x}) - \mathbf{c}\|^2 \\ &= k(\mathbf{x}, \mathbf{x}) - 2\sum_{p=1}^{N} \alpha_p k(\mathbf{x}_p, \mathbf{x}) + \sum_{p=1}^{N}\sum_{i=1}^{N} \alpha_p \alpha_i k(\mathbf{x}_p, \mathbf{x}_i). \end{aligned} \tag{52}$$

Support vectors are defined as the data points that reside on the edges of the contours.

A support function is defined as a positive scalar function $f : \mathbb{R}^n \to \mathbb{R}^+$ that approximates the support of a data distribution through its level set. Typically, the level set of f can be expressed as a union of multiple disjoint connected sets, represented by

$$L_f(r) = \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \le r\} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_m, \tag{53}$$

where each set $\mathcal{C}_i$ is both disjoint and connected, representing distinct clusters, and m indicates the total number of clusters identified by f.

A support function is constructed using the SVDD approach (or one-class SVM) [241,425], which transforms data points into a high-dimensional feature space to identify the smallest enclosing sphere enclosing most of the points. When this sphere is projected back to the original space, it can be decomposed into distinct components, each corresponding to distinct clusters.

SVC involves two main steps: estimating a support function via SVM training and assigning cluster labels. The labeling step has a time complexity of $O(N^2 m)$, where N is the number of data points and m is the number of sampling points per edge. SVC effectively forms arbitrary cluster boundaries and handles outliers. Using dynamical consistency, the labeling time can be improved to $O(\log N)$ [426].

Width parameters for Gaussian kernels and soft-margin constants are determined by a heuristic rule. SVC can terminate when the proportion of support vectors exceeds a threshold (about 10%) [424]. Multisphere SVC [427] employs multiple spheres to flexibly model clustersby detecting dense areas in the data space through the identification of minimal-radius spheres in the feature space, thus providing cluster prototypes and memberships.

Example 4. *This example, sourced from [424], utilizes a dataset comprising 183 samples. With parameters set to $\sigma = 0.1443$ and $C = 1$, the clusters can be effectively separated. Figure 7 displays the SVC results, where support vectors are marked with small circles, and different colors (gray scales) represent the clusters of the samples.*

Twin SVC [428] is a plane-based clustering method that employs a twin SVM approach to determine k cluster center planes through a sequence of QP problems, utilizing an initialization algorithm derived from the nearest-neighbor graph.

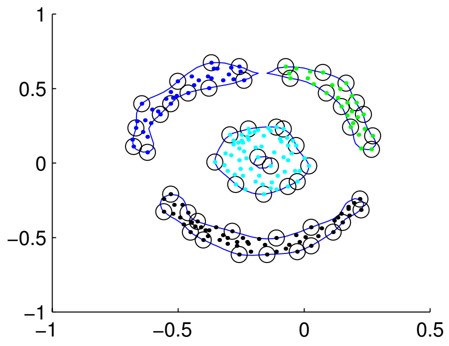

Figure 7. Clustering of a dataset via SVC with parameters $C = 1$ and $\sigma = 0.1443$. Points of different color beling to different classes. Adapted from [424], Figure 1d. ©Ben-Hur.

Maximum-margin clustering [429] seeks to identify the hyperplane with the largest margin that separates different clusters, reformulating the non-convex integer programming problem as a semi-definite programming problem. In [430], alternating optimization is applied, which modifies the loss function from hinge loss to Laplacian or square loss to enhance accuracy and speed by two to four orders of magnitude. Additionally, generalized maximum-margin clustering [431] further simplifies the complexity of the problem, reducing it from $O(N^2)$ to $O(N)$.

A cutting-plane maximum-margin clustering algorithm [432] addresses the non-convex problem by decomposing it into convex subproblems using CCCP, with each subproblem solved via a cutting-plane algorithm to enhance efficiency and accuracy. This approach converges in $O(sN)$ time, where N is the number of samples and s denotes dataset sparsity. Additionally, a multiclass extension of the algorithm is introduced.

Maximum volume clustering [433] is a discriminative model rooted in the large volume principle, featuring two approximation schemes: a soft-label method using sequential QP and a hard-label method based on semi-definite programming. This approach generalizes various clustering techniques, including spectral clustering, relaxed C-means, and information-maximization clustering, as specific cases under specific regularization conditions.

13. SVMs for One-Class Classification

Information retrieval that relies exclusively on positive training examples is essential for various applications, such as classifying sites of interest based solely on a user's activity history. Novelty detection identifies new patterns with minimal training data, often within imbalanced datasets. When both novel and normal samples are available, this task resembles a binary classification problem. Data description, or *one-class classification*, aims to compactly represent the characteristics of target data.

One-class SVM [241,424,425] is a kernel method that describes datasets with only positive examples. The main approaches are separating data points from the origin [425] and enclosing them with a sphere of minimum volume [241,434]. The first strategy identifies a hyperplane within a kernel feature space, permitting some training points to lie beyond it while maximizing the distance from the origin [425]. This allows the origin to represent the second class, enabling the application of standard two-class SVMs. The second method computes the smallest enclosing sphere for inliers. Both methods yield similar, and often identical, dual optimization formulations, particularly with the Gaussian kernel. In cases where all data are inliers, one-class SVM calculates the smallest enclosing sphere for the data.

The hard (soft) margin SVDD produces solutions that are equivalent to those of the hard (soft) margin one-class SVM, with the weight vector $\mathbf{w}$ corresponding to the center $\mathbf{c}$ of the SVDD solution [425]. The process of determining the soft-margin one-class SVM resembles fitting an MEB that accounts for outliers. In SVDD [434,435], the target data is compactly described by a hypersphere $(\mathbf{a}, R)$ determined by support vectors. However, SVDD can struggle to capture the overall density distribution of a dataset, as only support vectors influence the solution. The kernel trick enhances flexibility by operating in high-dimensional feature space [434].

To improve SVDD, density-induced SVDD [436] incorporates relative density degrees for each data point, allowing the hypersphere's center to shift towards denser regions. This method performs comparably to k-NN and SVM when negative data information is available.

Outlier-SVM [437] identifies outliers as representatives of a second class. In the context of information retrieval, both one-class SVM [425] and outlier-SVM [437] surpass traditional methods like prototypes and naive Bayes. While one-class SVM performs well for smaller categories, one-class neural networks and outlier-SVM excel in larger categories.

LS one-class SVM [438] reformulates the conventional one-class SVM [425], resembling LS-SVM. It extracts a hyperplane that optimally describes training objects using a quadratic loss function and equality constraints, minimizing distances in a regularized LS sense. However, LS one-class SVM sacrifices the sparsity characteristic typical of standard one-class SVMs, which can be mitigated through training sample pruning.

14. Incremental SVMs

Incremental learning methods for SVM are more computationally efficient than batch SVMs, as outlined in various studies [225,229,240,439–442].

ALMA$_p$ [443] approximates the maximal margin hyperplane for linearly separable data without relying on QP methods for speed akin to the perceptron algorithm. ALMA$_2$, as a perceptron-like algorithm, is faster and produces a sparser solution than SVM, but the accuracy is superior to than of perceptron learning but slight inferior to the accuracy of SVM. In tasks involving sparse target vectors, commonly found in text processing, ALMA$_p$ with $p > 2$ significantly outperforms ALMA$_2$. While ALMA$_2$ focuses on approximating the primal maximal margin problem, ALMA$_p$ represents a large-margin variant of the L_p-norm perceptron algorithm.

The exact incremental SVM [439] updates the SVM solution with each added or removed training example, requiring substantial memory for support vectors. Improvements to this method by utilizing gaxpy-type updates for the sensitivity vector can accelerate training by a factor of 5–20 [444].

Condensed SVM [445] keeps the number of support vectors to a minimum by integrating vector combinations. Incremental asymmetric proximal SVM (IAPSVM) [446] improves classifier accuracy through a greedy search for basis vectors and automatic parameter tuning. This method largely improves the accuracy compared to reduced-set methods proposed for proximal SVM, and outperforms SVMTorch and core vector machine while maintaining lower complexity.

Kernel Adatron [440] modifies the Adatron algorithm for maximum-margin classification using kernels. Active-set incremental SVM [447] employs a warm-start approach, while online algorithms for L_1-SVM [322] approximate solutions well, and scales well to tens of thousands of examples, but it does not achieve the accuracy of the exact solution.

Core vector machines [240] scale to millions of examples and approximate L_2-SVM solutions but may incur higher test errors. The method applies to both linear and nonlinear kernels. Metaheuristic kernel core vector machine [448] exploits a metaheuristic method [116,449] to optimizate a nonlinear combination of conventional kernel functions.

An incremental one-class SVM method [450] generates hyperspheres for each class, retaining the boundary candidates as support vectors. Online independent SVM [451] optimizes the model by projecting new observations onto a set of linearly independent ones, reducing time and space at a minor accuracy cost.

Online independent SVM [451] converges to an approximate SVM solution with each new observation, with the degree of approximation governed by a user-defined parameter. This approach utilizes a set of linearly independent observations and aims to project each new observation onto the current set, significantly lowering time and space demands while incurring only a minimal accuracy loss. Compared to standard SVM, online independent

SVM yields a more compact model, featuring a training complexity of asymptotically $O(N^2)$ and employing the L_2-norm of the slack variables.

Accurate online ν-SVM (AONSVM) [452] builds on exact incremental learning to adapt ν-SVM, managing equality constraint conflicts. Following the framework of exact incremental SVM [439], both accurate incremental SVR [442] and accurate online SVR [441] efficiently update SVR functions with new samples. An incremental ν-SVR [453] combines the initial adjustments with AONSVM steps, while an incremental SVOR algorithm [454] extends AONSVM for a modified ordinal regression formulation, converging to optimal solutions efficiently.

15. SVMs for Active, Transductive, and Semi-Supervised Learning

15.1. SVMs for Active Learning

An active learning algorithm utilizing SVM identifies positive examples within a dataset [455]. It selects the next point to label using two heuristics based on an SVM classifier trained on labeled data. The largest positive heuristic chooses the unlabeled point with the highest classification score, while the near boundary heuristic focuses on the point with the smallest absolute classification score. Both methods require retraining the SVM after each selection, making incremental learning a more efficient alternative.

In pool-based active learning [456], the learner has the ability to request labels for instances from a pool of unlabeled data. An SVM-based active learning algorithm determines which instances to query, significantly reducing the number of labeled training instances required in both inductive and transductive settings.

15.2. SVMs for Transductive Learning

SVM can be used for transduction, finding a hyperplane that maximizes the margin concerning both labeled and unlabeled data (see Figure 8). Transductive SVM has shown improved precision/recall performance in text classification compared to traditional inductive SVM [456].

Transduction leverages prior knowledge about unlabeled test patterns [215], simplifying the process compared to first establishing a general rule and then implementing it. It focuses on performance for specific test patterns, which is beneficial when the training and test data distributions differ. Transductive SVM treats unknown labels as optimization variables, aiming to maximize the margin while considering unlabeled data to identify a decision boundary that navigates low-density areas in the input space. This method embodies the cluster assumption in semi-supervised learning [457].

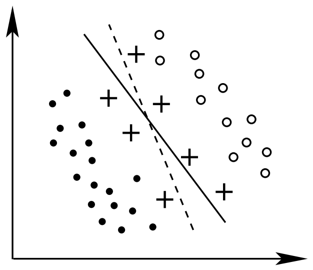

Figure 8. Comparison of SVM (solid line) and transductive SVM (dotted line), with $+$ indicating unlabeled instances. Points of different style beling to different classes.

Transductive SVM [21] enhances the SVM generalization accuracy by utilizing unlabeled data, finding a large margin hyperplane that is distant from unlabeled data. It is especially effective when labeled points are scarce and unlabeled points are abundant. Transduction, which pertains to labeling a test set, is fundamentally simpler than induction, which focuses on learning a general rule [21].

The transductive SVM problem involves L labeled examples $\{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, L\}$ and U unlabeled examples $\{\mathbf{x}_i \mid i = L+1, \ldots, N\}$, where $N = L + U$. The goal is to determine a binary vector $y = (y_{L+1}, \ldots, y_{L+U})$ that allows an SVM trained on $\mathcal{L} \cup (\mathcal{U} \times \mathcal{Y})$ to maximize the margin. This combinatorial problem can be approximated by forcing unlabeled examples to remain distant from the margin [21].

The primal method has a computational complexity of $O((L+U)^3)$ and requires storing the complete $(L+U) \times (L+U)$ kernel matrix [457]. A linear transductive SVM with L_1-norm regularization [458] decomposes its loss function into a linear component and a concave component, and then is solve using an integer programming method, approximating a combinatorial approach like CS^{3}VM [459]. SVMLight-twin SVM [460] is suitable for datasets with a few thousand examples, while ∇-twin SVM [457] optimizes using gradient descent.

A challenge with transductive SVM arises in high-dimensional spaces with limited training examples, potentially resulting in all unlabeled examples being classified into a single class. To mitigate this, a balancing constraint can be introduced to ensure unlabeled data are allocated to both classes, maintaining the same positive-negative ratio as in the labeled data [460].

Ultimately, transductive SVM learning establishes a decision boundary across the entire input space. This approach can be viewed as a form of inductive, semi-supervised learning.

15.3. SVMs for Semi-Supervised Learning

S^{3}VM [458,461,462] effectively utilizes large volumes of unlabeled data to develop high-quality classifiers. It applies the principle of margin maximization for both labeled and unlabeled instances, making S^{3}VM an inductive semi-supervised method rather than strictly transductive [458]. Various techniques exist for addressing the associated non-convex optimization challenge, including local combinatorial search [460], gradient descent [457], and convex-concave procedures [459,461], as well as semi-definite programming [429].

In linear S^{3}VM, the optimization problem involves minimizing the hyperplane parameters $(\mathbf{w}, b)$ along with the label vector $\mathbf{y}_U = (y_{L+1}, \ldots, y_N)^T$, and is given by

$$\min_{(\mathbf{w},b),\mathbf{y}_U} I(\mathbf{w}, b, \mathbf{y}_U) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} V(y_i, o_i) + C^* \sum_{i=L+1}^{N} V(y_i, o_i), \tag{54}$$

where $o_i = \mathbf{w}^T\mathbf{x}_i + b$ and $V(y_i, o_i)$ is typically the hinge loss:

$$V(y_i, o_i) = [\max(0, 1 - y_i o_i)]^p. \tag{55}$$

Common choices for p are 1 or 2. The kernel trick can be employed to create nonlinear decision boundaries.

The first two terms of (54) outline the SVM framework, while the third accounts for unlabeled data. The losses for labeled and unlabeled examples are weighted by hyperparameters C and C^*, reflecting label confidence and the cluster assumption, respectively. The class-balancing constraint [460] ensures

$$\frac{1}{U}\sum_{i=L+1}^{N} \max(y_i, 0) = r \quad \Longleftrightarrow \quad \frac{1}{U}\sum_{i=L+1}^{N} y_i = 2r - 1, \tag{56}$$

to mitigate unbalanced solutions by assigning a user-defined proportion r of unlabeled data to the positive class.

CCCP method is applied to S^{3}VM [459,461], solving a sequence of SVM optimization challenges with $L + 2U$ variables through iterative dual QP problems, providing a scalable approach for nonlinear scenarios. Successive convex optimization serves as an implementation of SMO. CCCP-twin SVM outperforms SVMlight-twin SVM and ∇-twin SVM [461], achieving a speed improvement by orders of magnitude.

An inductive semi-supervised learning approach from [463] uses a Gaussian kernel support function to estimate the support of the data distribution through an SVDD process with both labeled and unlabeled data. This method segments the data space into clustered regions for classification, resulting in a non-convex optimization problem.

S^3VMlight [460], which implements S^3VM, employs local combinatorial search along with a label-switching technique. ∇S^3VM [457] minimizes the objective directly using gradient descent, achieving $O(N^3)$ complexity, while S^3VMlight generally operates with a complexity of $O(n^3 + N^2)$ for n support vectors. Newton S^3VM method is proposed in [458], reducing the complexity of ∇S^3VM. Other methods like semi-supervised LS-SVM [464] and fully weighted semi-supervised LS-SVM [465] extend these ideas through various combinatorial optimization approaches.

Finally, while traditional kernel path algorithms are confined to convex problems, a kernel path algorithm for S^3VM [466] tracks non-convex solutions concerning kernel parameters by employing incremental and decremental learning strategies to manage Karush-Kuhn-Tucker violations, proving finite convergence.

16. Concluding Remarks

SVM was the first machine learning model founded on computational learning theory. Unlike connectionist models, which form the basis of neural networks and deep learning, SVM is a deterministic model that consistently produces optimal results through a precise, mathematical approach.

As a validation of computational learning theory, the SVM approach has been extensively studied in the 1990s and 2000s and is now considered a mature method, especially after deep learning became predominant around 2012. In this paper, we have summarized all major contributions of SVM research, presenting a comprehensive literature survey, and systematically introducing the principles and methods for the SVM approach. This paper also functions as a tutorial. A few possible directions for future investigation are presented here.

16.1. Integrating SVM into Deep Learning Architecture

SVM is essentially a deterministic, discriminative method based on computational learning theory, in contrast to the stochastic, generative method of neural networks. While deep learning is not a panacea [467], it is very powerful for classification of complicated classification and inference tasks. Future research on SVM could involve integrating it as a component within deep learning architectures [115,467].

Deep learning models are primarily composed of conventional models like MLP, long short-term memory (LSTM), and convolutional neural network, along with some unsupervised neural network models. Few deep learning frameworks utilize SVM as a building block, likely due to the considerable computational complexity related to the large number of support vectors, non-iterative learning algorithms, and the storage requirements for the $\mathbf{K}$ matrix. In contrast, deep learning models typically rely on gradient descent.

Lately, the kernel method has been used to explain the mechanism of deep neural networks [136–138], and to develop kernelized feedforward deep neural network models [135]. Nonetheless, further investigation into kernel and SVM methods in deep learning is called for. Future work can focus on customizing the SVM or kernel module for deep learning by developing iterative training algorithms that integrate smoothly within deep learning frameworks.

16.2. Solving SVM with Indefinite Matrices

LP can address the SVM problem without requiring positive definiteness, allowing LP-SVM to handle indefinite matrices [468]. As a kernel-as-features approach, LP-SVM utilizes kernel entries as additional features.

When the kernel matrix is positive semi-definite, SVM can effectively employ a convex optimization algorithm. However, to convert an indefinite kernel matrix $\mathbf{K}$ into a positive semi-definite form, several methods can be employed [469]:

- Clip: Negative eigenvalues are set to zero. IndefiniteSVM [470] implements this by modifying the spectrum.
- Shift: The spectrum is adjusted so that the smallest eigenvalue becomes zero.
- Flip: The eigenvalues are replaced with their absolute values. IKFD [471] applies this method in LDA.
- Square: Eigenvalues are squared, effectively using $\mathbf{K}\mathbf{K}^T$ in place of $\mathbf{K}$. This method is employed by potential SVM [346] and relevance vector machines [369], where the rows of the kernel are considered as features.

Krein spaces, as indefinite inner product spaces with a Hilbertian topology, allow SVMs to operate without requiring positive semi-definiteness. In reproducing kernel Krein spaces [32], the representer theorem applies, framing the SVM problem with indefinite kernels as a cost-stabilizing projection. Expanding on [32], Krein-space SVM [33] decomposes Krein spaces into Hilbert spaces, transforming the dual maximization into a convex minimization, though with a complexity of quadratic to cubic and a non-sparse decision function. The Krein-space indefinite core vector machine (iCVM) [472] manages indefinite kernels with a sparse model, achieving linear runtime via a low-rank approach, offering efficiency comparable to Krein-space SVM but with reduced computational demands. Theoretical breakthroughs of SVM with indefinite matrices are anticipated.

16.3. Expanding to Tensor Data

Tensors are increasingly common in prediction tasks, prompting research into methods for high-dimensional, multi-way data. Traditional approaches often reshape matrix data into vectors, which disrupts their structure. To address this, kernel methods have been extended to tensor data [473–475], using multi-linear or higher-order SVD. However, flattening tensors increases the dimensions of vectors or matrices, raising overfitting risks and losing structural information. Further advancements in unified, high-efficiency kernel and SVM-like tensor models, along with theoretical developments, are anticipated.

SVMs have been extended to matrix and tensor inputs to leverage structured data for classification. Support matrix machine [476] classifies matrix data by combining hinge loss with a spectral elastic net and nuclear norm penalties, solved with ADMM; statistical properties and a communication-efficient estimator for nuclear-norm-regularized SVMs are analyzed in [477]. For tensor data, support tensor machine [478] applies the kernel trick on tensors, with LIBSVM handling kernel functions, and achieves convergence rates similar to those of LS tensor regression under Gaussian assumptions [479]. Dual structure-preserving kernel (DuSK) for SVMs operates on canonical polyadic (CP) format using rank-one decomposition, supporting MMK methods like kernelized CP and Tucker models [480–484], with tensor train (TT)-format kernel approximations introduced in [485]. Support Tucker machine (STuM) [486] optimizes SVM weights via Tucker decomposition, with TT decomposition enhancing robustness over CP decomposition; TT-MMK [487] serves as a tailored kernel model for tensor classification in SVMs.

Author Contributions: All authors contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

ADMM	alternating direction method of multipliers	MVSVM	multiple view SVM
AONSVM	accurate online ν-SVM	NHSVM	nonparallel hyperplane SVM
ARMA	adaptive autoregressive moving-average	NMF	non-negative matrix factorization
CCA	canonical correlation analysis	NPPC	nonparallel-plane proximal classifier
CCCP	constrained concave-convex procedure	OLS	orthogonal least squares
CCICP	concave-inexact-convex procedure	PCA	principal component analysis
CG	conjugate gradient	PCVM	probabilistic classification vector machine
CP	canonical polyadic	pdf	probability density functions
DAG-SVM	directed acyclic graph SVM	pin-twin SVM	twin SVM with pinball loss
DSG	doubly stochastic gradient	PLapSVM	primal Laplacian SVM
DuSK	dual structure-preserving kernel	PLS	partial least squares
ECOC	error-correcting output codes	QCQP	quadratically constrained quadratic program
EVD	eigenvalue decomposition	QP	quadratic programming
FCM	fuzzy *C*-means	RBF	radial basis function
FLapSVM	fast Laplacian SVM	RBM	restricted Boltzmann machines
GEPSVM	generalized eigenvalue proximal SVM	RFE	recursive feature elimination
HCST	hybrid caching for SVM training	RKBS	reproducing kernel Banach spaces
ICA	independent component analysis	RKHS	reproducing kernel Hilbert spaces
iCVM	indefinite core vector machine	RKKS	reproducing kernel Krein space
KFBS	kernel forward-backward smoother	RKM	restricted kernel machines
KKT	Karush-Kuhn-Tucker	RLS	recursive least squares
kMER	kernel-based maximum entropy learning rule	RVM	relevance vector machine
KRR	ridge regression	S^3VM	semi-supervised SVM
LapSVM	Laplacian SVM	SDP	semidefinite programs
LDA	linear discriminant analysis	SGD	stochastic gradient descent
LFU	less frequently used	SMO	sequential minimal optimization
LMS	least mean squares	SOM	kernel self-organizing map
LP	linear programming	SRM	structural risk minimization
LRU	least recently used	STuM	support Tucker machine
LS	least squares	SVC	support vector clustering
LS-DC	least squares type of difference of convex loss	SVD	singular value decomposition
LS-SVM	least squares SVM	SVDD	support vector data description
LSTM	long short-term memory	SVM	support vector machine
MCPSVM	multiplane convex proximal SVM	SVMAC	SVM with Automatic Confidence
MEB	minimum enclosing ball	SVOR	support vector ordinal regression
MKL	multiple kernel learning	SVR	support vector regression
MLP	multilayer perceptron	SVRG	stochastic variance-reduced gradient descent
MNI	minimum norm interpolation	TSVH	twin support vector hypersphere
MPS	partition selection method	TT	tensor train
		VC	Vapnik–Chervonenkis

References

1. Aizerman, M.; Braverman, E.; Rozonoer, L. Theoretical foundations of the potential function method in pattern recognition learning. *Automat. Remote Contr.* **1964**, *25*, 821–837.
2. Mercer, T. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Royal Soc. Lond. Ser. A* **1909**, *209*, 415–446.
3. Muller, K.R.; Mika, S.; Ratsch, G.; Tsuda, K.; Scholkopf, B. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.* **2001**, *12*, 181–201. [CrossRef]
4. Du, K.-L.; Swamy, M.N.S. *Neural Networks and Statistical Learning*; Springer: London, UK, 2019.
5. Balcan, M.-F.; Blum, A.; Vempala, S. Kernels as features: On kernels, margins, and low-dimensional mappings. In Proceedings of the Algorithmic Learning Theory: 15th International Conference, Padova, Italy, 2–5 October 2004; pp. 194–205.
6. Ma, J. Function replacement vs. kernel trick. *Neurocomputing* **2003**, *50*, 479–483. [CrossRef]
7. Yang, C.; Wang, L.; Feng, J. On feature extraction via kernels. *IEEE Trans. Syst. Man. Cybern. B* **2008**, *38*, 553–557. [CrossRef]
8. Xu, Y.; Zhang, H. Refinable kernels. *J. Mach. Learn. Res.* **2007**, *8*, 2083–2120.
9. Scholkopf, B.; Smola, A.; Muller, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.* **1998**, *10*, 1299–1319. [CrossRef]

10. Suykens, J.A.K.; Van Gestel, T.; Vandewalle, J.; De Moor, B. A support vector machine formulation to PCA analysis and its kernel version. *IEEE Trans. Neural Netw.* **2003**, *14*, 447–450. [CrossRef]
11. Mika, S.; Ratsch, G.; Weston, J.; Scholkopf, B.; Muller, K.-R. Fisher discriminant analysis with kernels. In Proceedings of the Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop, Madison, WI, USA, 25 August 1999; pp. 41–48.
12. Yang, J.; Frangi, A.F.; Yang, J.-Y.; Zhang, D.; Jin, Z. KPCA plus LDA: A complete kernel Fisher discriminant framework for feature extraction and recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 230–244. [CrossRef]
13. Girolami, M. Mercer Kernel-based clustering in feature space. *IEEE Trans. Neural Netw.* **2002**, *13*, 780–784. [CrossRef]
14. Bach, F.R.; Jordan, M.I. Kernel independent component analysis. *J. Mach. Learn. Res.* **2002**, *3*, 1–48.
15. Martinez, D.; Bray, A. Nonlinear blind source separation using kernels. *IEEE Trans. Neural Netw.* **2003**, *14*, 228–235. [CrossRef] [PubMed]
16. Lai, P.L.; Fyfe, C. Kernel and nonlinear canonical correlation analysis. *Int. J. Neural Syst.* **2000**, *10*, 365–377. [CrossRef]
17. Lanckriet, G.R.G.; Ghaoui, L.E.; Bhattacharyya, C.; Jordan, M.I. A robust minimax approach to classification. *J. Mach. Learn. Res.* **2002**, *3*, 555–582.
18. Liwicki, S.; Zafeiriou, S.; Tzimiropoulos, G.; Pantic, M. Efficient online subspace learning with an indefinite kernel for visual tracking and recognition. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 1624–1636. [CrossRef]
19. Song, G.; Zhang, H. Reproducing kernel Banach spaces with the l_1 Norm II: Error analysis for regularized least square regression. *Neural Comput.* **2001**, *23*, 2713–2729. [CrossRef]
20. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proceedings of the COLT92: 5th Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; pp. 144–152.
21. Vapnik, V.N. *The Nature of Statistical Learning Theory*; Springer: New York, NY, USA, 1995.
22. Baker, J.L. Is there a support vector machine hiding in the dentate gyrus? *Neurocomputing* **2003**, *52–54*, 199–207. [CrossRef]
23. Jandel, M. A neural support vector machine. *Neural Netw.* **2010**, *23*, 607–613. [CrossRef]
24. Hammer, B.; Gersmann, K. A note on the universal approximation capability of support vector machines. *Neural Process. Lett.* **2003**, *17*, 43–53. [CrossRef]
25. Steinwart, I. Sparseness of support vector machines. *J. Mach. Learn. Res.* **2003**, *4*, 1071–1105.
26. Girosi, F. An equivalence between sparse approximation and support vector machines. *Neural Comput.* **1998**, *10*, 1455–1480. [CrossRef] [PubMed]
27. Bouboulis, P.; Theodoridis, S. Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS. *IEEE Trans. Signal Process.* **2011**, *59*, 964–978. [CrossRef]
28. Colbert, B.K.; Peet, M.M. A convex parametrization of a new class of universal kernel functions. *J. Mach. Learn. Res.* **2020**, *21*, 1–29.
29. Cox, D.; O'Sullivan, F. Asymptotic analysis of penalized likelihood and related estimators. *Ann. Statist.* **1990**, *18*, 1676–1695. [CrossRef]
30. Scholkopf, B.; Herbrich, R.; Smola, A.J. A generalized representer theorem. In Proceedings of the 14th Annual Conference on Computational Learning Theory, LNCS, Amsterdam, The Netherlands, 16–19 July 2001; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2111, pp. 416–426.
31. Bohn, B.; Rieger, C.; Griebel, M. A Representer Theorem for Deep Kernel Learning. *J. Mach. Learn. Res.* **2019**, *20*, 1–32.
32. Ong, C.S.; Mary, X.; Canu, S.; Smola, A.J. Learning with non-positive kernels. In Proceedings of the 21th International Conference on Machine Learning, Banff, AB, Canada, 4–8 July 2004; pp. 81–89.
33. Loosli, G.; Canu, S.; Ong, C.S. Learning SVM in Krein spaces. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 1204–1216. [CrossRef]
34. Wang, R.; Xu, Y. Representer theorems in Banach spaces: Minimum norm interpolation, regularized learning and semi-discrete inverse problems. *J. Mach. Learn. Res.* **2021**, *22*, 1–65.
35. Scholkopf, B. *Support Vector Learning*; R Oldenbourg Verlag: Munich, Germany, 1997.
36. Hoegaerts, L.; De Lathauwer, L.; Goethals, I.; Suykens, J.A.K.; Vandewalle, J.; De Moor, B. Efficiently updating and tracking the dominant kernel principal components. *Neural Netw.* **2007**, *20*, 220–229. [CrossRef]
37. Braun, M.L.; Buhmann, J.M.; Muller, K.-R. On relevant dimensions in kernel feature spaces. *J. Mach. Learn. Res.* **2008**, *9*, 1875–1908.
38. Scholkopf, B.; Mika, S.; Burges, C.J.C.; Knirsch, P.; Muller, K.R.; Ratsch, G.; Smola, A.J. Input space versus feature space in kernel-based methods. *IEEE Trans. Neural Netw.* **1999**, *10*, 1000–1017. [CrossRef]
39. Smola, A.J.; Mangasarian, O.; Scholkopf, B. *Sparse Kernel Feature Analysis*; Technical Report 99-03; Data Mining Institute, University of Wisconsin: Madison, WI, USA, 1999.
40. Alzate, C.; Suykens, J.A.K. Kernel component analysis using an epsilon-insensitive robust loss function. *IEEE Trans. Neural Netw.* **2008**, *19*, 1583–1598. [CrossRef] [PubMed]
41. Kim, K.I.; Franz, M.O.; Scholkopf, B. Iterative kernel principal component analysis for image modeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1351–1366. [PubMed]
42. Gunter, S.; Schraudolph, N.N.; Vishwanathan, S.V.N. Fast iterative kernel principal component analysis. *J. Mach. Learn. Res.* **2007**, *8*, 1893–1918.

43. Washizawa, Y. Adaptive subset kernel principal component analysis for time-varying patterns. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 1961–1973. [CrossRef] [PubMed]
44. Ding, M.; Tian, Z.; Xu, H. Adaptive kernel principal component analysis. *Signal Process.* **2010**, *90*, 1542–1553. [CrossRef]
45. Du, K.-L.; Swamy, M.N.S.; Wang, Z.-Q.; Mow, W.H. Matrix factorization techniques in machine learning, signal processing and statistics. *Mathematics* **2023**, *11*, 2674. [CrossRef]
46. Fan, J.; Chow, T.W.S. Exactly robust kernel principal component analysis. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 749–761. [CrossRef]
47. Chin, T.-J.; Schindler, K.; Suter, D. Incremental kernel SVD for face recognition with image sets. In Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition, Southampton, UK, 10–12 April 2006; pp. 461–466.
48. Chin, T.-J.; Suter, D. Incremental kernel principal component analysis. *IEEE Trans. Image Process.* **2007**, *16*, 1662–1674. [CrossRef]
49. Huang, S.-Y.; Yeh, Y.-R.; Eguchi, S. Robust kernel principal component analysis. *Neural Comput.* **2009**, *21*, 3179–3213. [CrossRef]
50. Dhanjal, C.; Gunn, S.R.; Shawe-Taylor, J. Efficient sparse kernel feature extraction based on partial least squares. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 1347–1361. [CrossRef]
51. Jenssen, R. Kernel entropy component analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 847–860. [CrossRef]
52. Papaioannou, A.; Zafeiriou, S. Principal component analysis with complex kernel: The widely linear model. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 1719–1726. [CrossRef]
53. Shashua, A. On the relationship between the support vector machine for classification and sparsified Fisher's linear discriminant. *Neural Process. Lett.* **1999**, *9*, 129–139. [CrossRef]
54. Rodriguez-Lujan, I.; Santa Cruz, C.; Huerta, R. On the equivalence of kernel Fisher discriminant analysis and kernel quadratic programming feature selection. *Pattern Recogn. Lett.* **2011**, *32*, 1567–1571. [CrossRef]
55. Baudat, G.; Anouar, F. Generalized discriminant analysis using a kernel approach. *Neural Comput.* **2000**, *12*, 2385–2404. [CrossRef]
56. Lu, J.; Plataniotis, K.N.; Venetsanopoulos, A.N. Face recognition using kernel direct discriminant analysis algorithms. *IEEE Trans. Neural Netw.* **2003**, *14*, 117–126.
57. Dufrenois, F. A one-class kernel Fisher criterion for outlier detection. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 982–994. [CrossRef]
58. Pekalska, E.; Haasdonk, B. Kernel discriminant analysis for positive definite and indefinite kernels. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 1017–1031. [CrossRef]
59. Ji, S.; Ye, J. Kernel uncorrelated and regularized discriminant analysis: A theoretical and computational study. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 1311–1321.
60. Heo, G.; Gader, P. Robust kernel discriminant analysis using fuzzy memberships. *Pattern Recogn.* **2011**, *44*, 716–723. [CrossRef]
61. Wang, L. Feature selection with kernel class separability. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *30*, 1534–1546. [CrossRef]
62. Cevikalp, H.; Neamtu, M.; Wilkes, M. Discriminative common vector method with kernels. *IEEE Trans. Neural Netw.* **2006**, *17*, 1550–1565. [CrossRef]
63. Zheng, W.; Zhao, L.; Zou, C. Foley-Sammon optimal discriminant vectors using kernel approach. *IEEE Trans. Neural Netw.* **2005**, *16*, 1–9. [CrossRef]
64. Zheng, W.; Lin, Z.; Tang, X. A rank-one update algorithm for fast solving kernel Foley-Sammon optimal discriminant vectors. *IEEE Trans. Neural Netw.* **2010**, *21*, 393–403. [CrossRef]
65. Wolf, L.; Shashua, A. Learning over sets using kernel principal angles. *J. Mach. Learn. Res.* **2003**, *4*, 913–931.
66. You, D.; Hamsici, O.C.; Martinez, A.M. Kernel optimization in discriminant analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 631–638. [CrossRef]
67. Kim, D.W.; Lee, K.Y.; Lee, D.; Lee, K.H. A kernel-based subtractive clustering method. *Pattern Recogn. Lett.* **2005**, *26*, 879–891. [CrossRef]
68. Du, K.-L. Clustering: A neural network approach. *Neural Netw.* **2010**, *23*, 89–107. [CrossRef]
69. Dhillon, I.S.; Guan, Y.; Kulis, B. Kernel *k*-means, spectral clustering and normalized cuts. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 22–25 August 2004; pp. 551–556.
70. Dhillon, I.S.; Guan, Y.; Kulis, B. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1944–1957. [CrossRef]
71. Zhang, D.Q.; Chen, S.C. Clustering incomplete data using kernel-based fuzzy C-means algorithm. *Neural Process. Lett.* **2003**, *18*, 155–162. [CrossRef]
72. Kim, D.W.; Lee, K.Y.; Lee, D.; Lee, K.H. Evaluation of the performance of clustering algorithms kernel-induced feature space. *Pattern Recogn.* **2005**, *38*, 607–611. [CrossRef]
73. MacDonald, D.; Fyfe, C. The kernel self organising map. In Proceedings of the 4th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, Brighton, UK, 30 August–1 September 2000; Volume 1, pp. 317–320.
74. Yin, H.; Allinson, N. Self-organising mixture networks for probability density estimation. *IEEE Trans. Neural Netw.* **2001**, *12*, 405–411. [CrossRef] [PubMed]
75. Lau, K.W.; Yin, H.; Hubbard, S. Kernel self-organising maps for classification. *Neurocomputing* **2006**, *69*, 2033–2040. [CrossRef]
76. van Hulle, M.M. Kernel-based equiprobabilistic topographic map formation. *Neural Comput.* **1998**, *10*, 1847–1871. [CrossRef]
77. Teh, C.S.; Lim, C.P. Monitoring the formation of kernel-based topographic maps in a hybrid SOM-kMER model. *IEEE Trans. Neural Netw.* **2006**, *17*, 1336–1341.

78. Teh, C.S.; Lim, C.P. An artificial neural network classifier design based-on variable kernel and non-parametric density estimation. *Neural Process. Lett.* **2008**, *27*, 137–151. [CrossRef]
79. Qin, A.K.; Suganthan, P.N. Kernel neural gas algorithms with application to cluster analysis. In Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, 26 August 2004; Volume 4, pp. 617–620.
80. Filippone, M.; Masulli, F.; Rovetta, S. Applying the possibilistic *c*-means algorithm in kernel-induced spaces. *IEEE Trans. Fuzzy Syst.* **2010**, *18*, 572–584. [CrossRef]
81. Alzate, C.; Suykens, J.A.K. Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 335–347. [CrossRef]
82. Boubacar, H.A.; Lecoeuche, S.; Maouche, S. SAKM: Self-adaptive kernel machine. A kernel-based algorithm for online clustering. *Neural Netw.* **2008**, *21*, 1287–1301. [CrossRef]
83. Steinwart, I.; Sriperumbudur, B.K.; Thomann, P. Adaptive clustering using kernel density estimators. *J. Mach. Learn. Res.* **2023**, *24*, 1–56.
84. Heinz, C.; Seeger, B. Cluster kernels: Resource-aware kernel density estimators over streaming data. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 880–893. [CrossRef]
85. Chitta, R.; Jin, R.; Havens, T.C.; Jain, A.K. Approximate kernel k-means: Solution to large scale kernel clustering. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011; pp. 895–903.
86. Wang, S.; Gittens, A.; Mahoney, M.W. Scalable kernel k-means clustering with Nystrom approximation: Relative-error bounds. *J. Mach. Learn. Res.* **2019**, *20*, 431–479.
87. Chitta, R.; Jin, R.; Jain, A.K. Efficient kernel clustering using random Fourier features. In Proceedings of the IEEE 12th International Conference on Data Mining, Brussels, Belgium, 10–13 December 2012; pp. 161–170.
88. Tsapanos, N.; Tefas, A.; Nikolaidis, N.; Pitas, I. A distributed framework for trimmed kernel K-means clustering. *Pattern Recognit.* **2015**, *48*, 2685–2698. [CrossRef]
89. Zhou, X.; Wang, X. Memory and communication efficient federated kernel *k*-means. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 7114–7125. [CrossRef]
90. Evgeniou, T.; Pontil, M.; Poggio, T. Regularization networks and support vector machines. *Adv. Comput. Math.* **2000**, *13*, 1–50. [CrossRef]
91. Shi, L.; Huang, X.; Feng, Y.; Suykens, J.A.K. Sparse kernel regression with coefficient-based ℓ_q regularization. *J. Mach. Learn. Res.* **2019**, *20*, 1–44.
92. Gerfo, L.L.; Rosasco, L.; Odone, F.; De Vito, E.; Verri, A. Spectral algorithms for supervised learning. *Neural Comput.* **2008**, *20*, 1873–1897. [CrossRef]
93. Yao, Y.; Rosasco, L.; Caponnetto, A. On early stopping in gradient descent learning. *Constr. Approx.* **2007**, *26*, 289–315. [CrossRef]
94. Blanchard, G.; Kramer, N. Convergence rates of kernel conjugate gradient for random design regression. *Anal. Appl.* **2016**, *14*, 763–794. [CrossRef]
95. Lin, S.B.; Zhou, D.X. Optimal learning rates for kernel partial least squares. *J. Fourier Anal. Appl.* **2018**, *24*, 908–933. [CrossRef]
96. Lin, S.-B.; Lei, Y.; Zhou, D.-X. Boosted kernel ridge regression: Optimal learning rates and early stopping. *J. Mach. Learn. Res.* **2019**, *20*, 1–36.
97. Sun, H.; Wu, Q. Optimal rates of distributed regression with imperfect kernels. *J. Mach. Learn. Res.* **2021**, *22*, 1–34.
98. Lin, S.-B.; Wang, D.; Zhou, D.-X. Distributed kernel ridge regression with communications. *J. Mach. Learn. Res.* **2020**, *21*, 1–38.
99. Liu, Z.; Li, M. On the estimation of derivatives using plug-in kernel ridge regression estimators. *J. Mach. Learn. Res.* **2023**, *24*, 1–37.
100. Liu, F.; Shi, L.; Huang, X.; Yang, J.; Suykens, J.A.K. Generalization properties of hyper-RKHS and its applications. *J. Mach. Learn. Res.* **2021**, *22*, 1–38.
101. Bognar, J. *Indefinite Inner Product Spaces*; Springer: Berlin/Heidelberg, Germany, 1974.
102. Smola, A.J.; Ovari, Z.L.; Williamson, R.C. Regularization with dot-product kernels. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2001; pp. 308–314.
103. Wright, M.A.; Gonzalez, J.E. Transformers are deep infinite-dimensional non-mercer binary kernel machines. *arXiv* **2021**, arXiv:2106.01506.
104. Wang, W.; Jing, B.-Y. Gaussian process regression: Optimality, robustness, and relationship with kernel ridge regression. *J. Mach. Learn. Res.* **2022**, *23*, 1–67.
105. Jaakkola, T.; Haussler, D. Probabilistic kernel regression models. In Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 3–6 January 1999; Morgan Kaufmann: San Francisco, CA, USA, 1999.
106. Zhu, J.; Hastie, T. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2002; Volume 14.
107. Liu, F.; Huang, X.; Gong, C.; Yang, J.; Suykens, J.A.K. Indefinite kernel logistic regression with concave-inexact-convex procedure. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 765–776. [CrossRef]
108. Bartels, S.; Hennig, P. Conjugate gradients for kernel machines. *J. Mach. Learn. Res.* **2020**, *21*, 1–42.
109. Sigrist, F. KTBoost: Combined Kernel and Tree Boosting. *Neural Process. Lett.* **2021**, *53*, 1147–1160. [CrossRef]

110. Gu, B.; Geng, X.; Li, X.; Shi, W.; Zheng, G.; Deng, C.; Huang, H. Scalable kernel ordinal regression via doubly stochastic gradients. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 3677–3689. [CrossRef] [PubMed]
111. Smola, A.J.; Scholkopf, B. A tutorial on support vector regression. *Statist. Comput.* **2004**, *14*, 199–222. [CrossRef]
112. Tu, H.-H.; Lin, H.-T. One-sided support vector regression for multiclass cost-sensitive classification. In Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; pp. 1095–1102.
113. Frank, E.; Hall, M. A simple approach to ordinal classification. In Proceedings of the 12th European Conference on Machine Learning, Freiburg, Germany, 5–7 September 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 145–156.
114. Waegeman, W.; Boullart, L. An ensemble of weighted support vector machines for ordinal regression. *Int. J. Comput. Syst. Sci. Eng.* **2009**, *3*, 47–51.
115. Du, K.-L.; Leung, C.-S.; Mow, W.H.; Swamy, M.N.S. Perceptron: Learning, generalization, model Selection, fault tolerance, and role in the deep learning era. *Mathematics* **2022**, *10*, 4730. [CrossRef]
116. Du, K.-L.; Swamy, M.N.S. *Neural Networks in a Softcomputing Framework*; Springer: London, UK, 2006.
117. Frieb, T.-T.; Harrison, R.F. A kernel-based ADALINE. In Proceedings of the European Symposium on Artificial Neural Networks, Bruges, Belgium, 21–23 April 1999; pp. 245–250.
118. Freund, Y.; Schapire, R.E. Large margin classifcation using the perceptron algorithm. *Mach. Learn.* **1999**, *37*, 277–296. [CrossRef]
119. Kivinen, J.; Smola, A.; Williamson, R.C. Online learning with kernels. *IEEE Trans. Signal Process.* **2004**, *52*, 2165–2176. [CrossRef]
120. Ruiz, A.; Lopez-de-Teruel, P.E. Nonlinear kernel-based statistical pattern analysis. *IEEE Trans. Neural Netw.* **2001**, *12*, 16–32. [CrossRef]
121. Rosipal, R.; Trejo, L.J. Kernel partial least squares regression in reproducing kernel Hilbert spaces. *J. Mach. Learn. Res.* **2001**, *2*, 97–123.
122. Engel, Y.; Mannor, S.; Meir, R. The kernel recursive least-squares algorithm. *IEEE Trans. Signal Process.* **2004**, *52*, 2275–2285. [CrossRef]
123. Lee, J.; Nikolopoulos, D.S.; Vandierendonck, H. Mixed-precision kernel recursive least squares. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 1284–1298. [CrossRef]
124. Liu, W.; Pokharel, P.P.; Principe, J.C. The kernel least-mean-square algorithm. *IEEE Trans. Signal Process.* **2008**, *56*, 543–554. [CrossRef]
125. Yoshino, H.; Dong, C.; Washizawa, Y.; Yamashita, Y. Kernel Wiener filter and its application to pattern recognition. *IEEE Trans. Neural Netw.* **2010**, *21*, 1719–1730. [CrossRef]
126. Li, K.; Principe, J.C. The kernel adaptive autoregressive-moving-average algorithm. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 334–346. [CrossRef]
127. Zhang, M.; Wang, X.; Chen, X.; Zhang, A. The kernel conjugate gradient algorithms. *IEEE Trans. Signal Process.* **2018**, *66*, 4377–4387. [CrossRef]
128. Lam, H.; Zhang, H. Doubly robust Stein-kernelized Monte Carlo estimator: Simultaneous bias-variance reduction and supercanonical convergence. *J. Mach. Learn. Res.* **2023**, *24*, 1–58.
129. De la Torre, F. A least-squares framework for component analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 1041–1055. [CrossRef]
130. Le, T.; Nguyen, K.; Phung, D. Improving kernel online learning with a snapshot memory. *Mach. Learn.* **2022**, *111*, 997–1018. [CrossRef]
131. Liu, W.; Principe, J.C. Kernel affine projection algorithms. *EURASIP J. Adv. Signal Process.* **2008**, *2008*, 784292. [CrossRef]
132. Page, S.; Grunewalder, S. Ivanov-regularised least-squares estimators over large RKHSs and their interpolation Spaces. *J. Mach. Learn. Res.* **2019**, *20*, 1–49.
133. Arashloo, S.R.; Kittler, J. Robust One-Class Kernel Spectral Regression. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 999–1013. [CrossRef]
134. Wang, R.; Xu, Y.; Yan, M. Sparse representer theorems for learning in reproducing kernel Banach spaces. *J. Mach. Learn. Res.* **2024**, *25*, 1–45.
135. Duan, S.; Yu, S.; Chen, Y.; Principe, J.C. On kernel method-based connectionist models and supervised deep learning without backpropagation. *Neural Comput.* **2020**, *32*, 97–135. [CrossRef] [PubMed]
136. Jacot, A.; Gabriel, F.; Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Newry, UK, 2018; Volume 31.
137. Lai, J.; Xu, M.; Chen, R.; Lin, Q. Generalization ability of wide neural networks on R. *arXiv* **2023**, arXiv:2302.05933.
138. Xu, J.; Zhu, H. Overparametrized multi-layer neural networks: Uniform concentration of neural tangent kernel and convergence of stochastic gradient descent. *J. Mach. Learn. Res.* **2024**, *25*, 1–83.
139. Berlinet, A.; Thomas-Agnan, C. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2004.
140. Smola, A.; Gretton, A.; Song, L.; Scholkopf, B. A Hilbert space embedding for distributions. In Proceedings of the 18th International Conference on Algorithmic Learning Theory, Sendai, Japan, 1–4 October 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 13–31.

141. Blanchard, G.; Bousquet, O.; Zwald, L. Statistical properties of kernel principal component analysis. *Mach. Learn.* **2007**, *66*, 259–294. [CrossRef]
142. Fukumizu, K.; Bach, F.R.; Gretton, A. Statistical consistency of kernel canonical correlation analysis. *J. Mach. Learn. Res.* **2007**, *8*, 361–383.
143. Rosasco, L.; Belkin, M.; Vito, E.D. On learning with integral operators. *J. Mach. Learn. Res.* **2010**, *11*, 905–934.
144. Mollenhauer, M.; Klus, S.; Schutte, C.; Koltai, P. Kernel autocovariance operators of stationary processes: Estimation and convergence. *J. Mach. Learn. Res.* **2022**, *23*, 1–34.
145. Gretton, A.; Herbrich, R.; Smola, A.; Bousquet, O.; Scholkopf, B. Kernel methods for measuring independence. *J. Mach. Learn. Res.* **2005**, *6*, 2075–2129.
146. Xu, Z.; Huang, K.; Zhu, J.; King, I.; Lyua, M.R. A novel kernel-based maximum a posteriori classification method. *Neural Netw.* **2009**, *22*, 977–987. [CrossRef]
147. Peleg, D.; Meir, R. A sparsity driven kernel machine based on minimizing a generalization error bound. *Pattern Recogn.* **2009**, *42*, 2607–2614. [CrossRef]
148. Kim, J.; Scott, C.D. L_2 kernel classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1822–1831. [PubMed]
149. Ghari, P.M.; Shen, Y. Graph-aided online multi-kernel learning. *J. Mach. Learn. Res.* **2023**, *24*, 1–44.
150. Li, C.; Xie, H.-B.; Fan, X.; Xu, R.Y.D.; Van Huffel, S.; Mengersen, K. Kernelized sparse Bayesian matrix factorization. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 391–404. [CrossRef]
151. Buciu, I.; Nikolaidis, N.; Pitas, I. Nonnegative matrix factorization in polynomial feature space. *IEEE Trans. Neural Netw.* **2008**, *19*, 1090–1100. [CrossRef]
152. Zafeiriou, S.; Petrou, M. Nonlinear nonnegative component analysis algorithms. *IEEE Trans. Image Process.* **2010**, *19*, 1050–1066. [CrossRef]
153. Dwivedi, R.; Mackey, L. Kernel Thinning. *J. Mach. Learn. Res.* **2024**, *25*, 1–77.
154. Aravkin, A.Y.; Bottegal, G.; Pillonetto, G. Boosting as a kernel-based method. *Mach. Learn.* **2019**, *108*, 1951–1974. [CrossRef]
155. Gebhardt, G.H.W.; Kupcsik, A.; Neumann, G. The kernel Kalman rule. *Mach. Learn.* **2019**, *108*, 2113–2157. [CrossRef]
156. Ormoneit, D.; Sen, S. Kernel-based reinforcement learning. *Mach. Learn.* **2002**, *49*, 161–178. [CrossRef]
157. Barreto, A.M.S.; Precup, D.; Pineau, J. Practical kernel-based reinforcement learning. *J. Mach. Learn. Res.* **2016**, *17*, 1–70.
158. Liu, J.; Lian, H. Kernel-based decentralized policy evaluation for reinforcement learning. *IEEE Trans. Neural Netw. Learn. Syst.* 2024, *in press*.
159. Williams, C.K.I.; Seeger, M. Using the Nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems*; Leen, T., Dietterich, T., Tresp, V., Eds.; MIT Press: Cambridge, MA, USA, 2001; Volume 13; pp. 682–688.
160. Rahimi, A.; Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2007; Volume 20, pp. 1177–1184.
161. Rahimi, A.; Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2008; Volume 21, pp. 1313–1320.
162. Le, Q.; Sarlos, T.; Smola, A. Fastfood—Approximating kernel expansions in loglinear time. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 17–19 June 2013; Volume 28, pp. 244–252.
163. Zhang, B.; Zhang, H.; Ge, S.S. Face recognition by applying wavelet subband representation and kernel associative memory. *IEEE Trans. Neural Netw.* **2004**, *15*, 166–177. [CrossRef] [PubMed]
164. Garcia, C.; Moreno, J.A. The Hopfield associative memory network: Improving performance with the kernel "trick". In *Advances in Artificial Intelligence—IBERAMIA 2004, Proceedings of the 9th Ibero-American Conference on AI, Puebla, Mexico, 22–26 November 2004, Proceedings, LNCS*; Springer: Berlin/Heidelberg, Germany, 2024; Volume 3315, pp. 871–880.
165. Perfetti, R.; Ricci, E. Recurrent correlation associative memories: A feature space perspective. *IEEE Trans. Neural Netw.* **2008**, *19*, 333–345. [CrossRef] [PubMed]
166. Zheng, W.; Zhou, X.; Zou, C.; Zhao, L. Facial expression recognition using kernel canonical correlation analysis (KCCA). *IEEE Trans. Neural Netw.* **2006**, *17*, 233–238. [CrossRef] [PubMed]
167. Alzate, C.; Suykens, J.A.K. A regularized kernel CCA contrast function for ICA. *Neural Netw.* **2008**, *21*, 170–181. [CrossRef]
168. Harmeling, S.; Ziehe, A.; Kawanabe, M.; Muller, K.-R. Kernel-based nonlinear blind source separation. *Neural Comput.* **2003**, *15*, 1089–1124. [CrossRef]
169. Bohmer, W.; Grunewalder, S.; Nickisch, H.; Obermayer, K. Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. *Mach. Learn.* **2012**, *89*, 67–86. [CrossRef]
170. Gao, J.; Kwan, P.W.; Shi, D. Sparse kernel learning with LASSO and Bayesian inference algorithm. *Neural Netw.* **2010**, *23*, 257–264. [CrossRef]
171. Chang, Y.-W.; Hsieh, C.-J.; Chang, K.-W.; Ringgaard, M.; Lin, C.-J. Training and testing low-degree polynomial data mappings via linear SVM. *J. Mach. Learn. Res.* **2010**, *11*, 1471–1490.
172. Xiao, S.; Tan, M.; Xu, D.; Dong, Z.Y. Robust kernel low-rank representation. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 2268–2281. [CrossRef]
173. Yang, T.; Li, Y.-F.; Mahdavi, M.; Jin, R.; Zhou, Z.-H. Nystrom method vs random Fourier features: A theoretical and empirical comparison. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2010; pp. 476–484.
174. Vincent, P.; Bengio, Y. Kernel matching pursuit. *Mach. Learn.* **2002**, *48*, 165–187. [CrossRef]

175. Orabona, F.; Keshet, J.; Caputo, B. Bounded kernel-based online learning. *J. Mach. Learn. Res.* **2009**, *10*, 2643–2666.
176. Dekel, O.; Shalev-Shwartz, S.; Singer, Y. The Forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.* **2007**, *37*, 1342–1372. [CrossRef]
177. Zhang, T. Leave-one-out bounds for kernel methods. *Neural Comput.* **2003**, *15*, 1397–1437. [CrossRef]
178. Paiva, A.R.C.; Park, I.; Principe, J.C. A reproducing kernel Hilbert space framework for spike train signal processing. *Neural Comput.* **2009**, *21*, 424–449. [CrossRef]
179. Nashed, M.Z.; Walter, G.G. General sampling theorem for functions in reproducing kernel Hilbert space. *Math. Contr. Signals Syst.* **1991**, *4*, 363–390. [CrossRef]
180. Ogawa, H. What can we see behind sampling theorems? *IEICE Trans. Fund.* **2009**, *E92-A*, 688–707. [CrossRef]
181. Tanaka, A.; Imai, H.; Miyakoshi, M. Kernel-induced sampling theorem. *IEEE Trans. Signal Process.* **2010**, *58*, 3569–3577. [CrossRef]
182. Aravkin, A.Y.; Bell, B.M.; Burke, J.V.; Pillonetto, G. The connection between Bayesian estimation of a Gaussian random field and RKHS. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1518–1524. [CrossRef]
183. Suykens, J.A.K. Deep restricted kernel machines using conjugate feature duality. *Neural Comput.* **2017**, *29*, 2123–2163. [CrossRef]
184. Suykens, J.A.K.; Van Gestel, T.; De Brabanter, J.; De Moor, B.; Vandewalle, J. *Least Squares Support Vector Machines*; World Scientific: Singapore, 2002.
185. Pandey, A.; Schreurs, J.; Suykens, J.A.K. Generative restricted kernel machines: A framework for multi-view generation and disentangled feature learning. *Neural Netw.* **2021**, *135*, 177–191. [CrossRef] [PubMed]
186. Lanckriet, G.R.G.; Cristianini, N.; Bartlett, P.; Ghaoui, L.E.; Jordan, M.I. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.* **2004**, *5*, 27–72.
187. Ong, C.S.; Smola, A.J.; Williamson, R.C. Learning the kernel with hyperkernels. *J. Mach. Learn. Res.* **2005**, *6*, 1043–1071.
188. Sonnenburg, S.; Ratsch, G.; Schafer, C.; Scholkopf, B. Large scale multiple kernel learning. *J. Mach. Learn. Res.* **2006**, *7*, 1531–1565.
189. Ye, J.; Ji, S.; Chen, J. Multi-class discriminant kernel learning via convex programming. *J. Mach. Learn. Res.* **2008**, *9*, 719–758.
190. Kim, S.-J.; Magnani, A.; Boyd, S. Optimal kernel selection in kernel Fisher discriminant analysis. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 465–472.
191. Subrahmanya, N.; Shin, Y.C. Sparse multiple kernel learning for signal processing applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 788–798. [CrossRef]
192. Yang, H.; Xu, Z.; Ye, J.; King, I.; Lyu, M.R. Efficient sparse generalized multiple kernel learning. *IEEE Trans. Neural Netw.* **2011**, *22*, 433–446. [CrossRef]
193. Rakotomamonjy, A.; Bach, F.; Canu, S.; Grandvalet, Y. SimpleMKL. *J. Mach. Learn. Res.* **2008**, *9*, 2491–2521.
194. Chapelle, O.; Rakotomamonjy, A. Second order optimization of kernel parameters. In Proceedings of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels, Whistler, BC, Canada, 12 December 2008.
195. Kloft, M.; Brefeld, U.; Sonnenburg, S.; Zien, A. l_p-norm multiple kernel learning. *J. Mach. Learn. Res.* **2011**, *12*, 953–997.
196. Aflalo, J.; Ben-Tal, A.; Bhattacharyya, C.; Nath, J.S.; Raman, S. Variable sparsity kernel learning. *J. Mach. Learn. Res.* **2011**, *12*, 565–592.
197. Suzuki, T.; Tomioka, R. SpicyMKL: A fast algorithm for multiple kernel learning with thousands of kernels. *Mach. Learn.* **2011**, *85*, 77–108. [CrossRef]
198. Xu, X.; Tsang, I.W.; Xu, D. Soft margin multiple kernel learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 749–761. [PubMed]
199. Vishwanathan, S.V.N.; Sun, Z.; Ampornpunt, N.; Varma, M. Multiple kernel learning and the SMO algorithm. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2010.
200. Gonen, M. Bayesian efficient multiple kernel learning. In Proceedings of the 29th International Conference on Machine Learning, Edinburgh, UK, 26 June–1 July 2012; Volume 1, pp. 1–8.
201. Mao, Q.; Tsang, I.W.; Gao, S.; Wang, L. Generalized multiple kernel learning with data-dependent priors. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1134–1148. [CrossRef]
202. Huang, H.-C.; Chuang, Y.-Y.; Chen, C.-S. Multiple kernel fuzzy clustering. *IEEE Trans. Fuzzy Syst.* **2012**, *20*, 120–134. [CrossRef]
203. Bickel, S.; Scheffer, T. Multi-view clustering. In Proceedings of the 4th IEEE International Conference on Data Mining (ICDM'04), Brighton, UK, 1–4 November 2004; pp. 19–26.
204. Liu, X.; Dou, Y.; Yin, J.; Wang, L.; Zhu, E. Multiple kernel K-means clustering with matrix-induced regularization. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30, pp. 1–7.
205. Zhou, S.; Ou, Q.; Liu, X.; Wang, S.; Liu, L.; Wang, S.; Zhu, E.; Yin, J.; Xu, X. Multiple kernel clustering with compressed subspace alignment. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 252–263. [CrossRef]
206. Yao, Y.; Li, Y.; Jiang, B.; Chen, H. Multiple kernel *k*-means clustering by selecting representative kernels. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4983–4996. [CrossRef]
207. Han, Y.; Yang, K.; Yang, Y.; Ma, Y. Localized multiple kernel learning with dynamical clustering and matrix regularization. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 486–499. [CrossRef]
208. Wang, C.; Chen, M.; Huang, L.; Lai, J.; Yu, P.S. Smoothness regularized multiview subspace clustering with kernel learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 5047–5060. [CrossRef]
209. Wang, J.; Li, Z.; Tang, C.; Liu, S.; Wan, X.; Liu, X. Multiple kernel clustering with adaptive multi-scale partition selection. *IEEE Trans. Know. Data Eng.* **2024**, *36*, 6641–6652. [CrossRef]

210. Li, M.; Zhang, Y.; Ma, C.; Liu, S.; Liu, Z.; Yin, J.; Liu, X.; Liao, Q. Regularized simple multiple kernel *k*-means with kernel average alignment. *IEEE Trans. Neural Netw. Learn. Syst.* 2024, *in press*.
211. Alioscha-Perez, M.; Oveneke, M.C.; Sahli, H. SVRG-MKL: A fast and scalable multiple kernel learning solution for features combination in multi-class classification problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 1710–1723. [CrossRef]
212. Fu, L.; Zhang, M.; Li, H. Sparse RBF Networks with Multi-kernels. *Neural Process. Lett.* **2010**, *32*, 235–247. [CrossRef]
213. Hong, S.; Chae, J. Distributed online learning with multiple kernels. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 1263–1277 [CrossRef] [PubMed]
214. Shen, Y.; Chen, T.; Giannakis, G.B. Random feature-based online multi-kernel learning in environments with unknown dynamics. *J. Mach. Learn. Res.* **2019**, *20*, 1–36.
215. Vapnik, V.N. *Estimation of Dependences Based on Empirical Data*; Springer: New York, NY, USA, 1982.
216. Vapnik, V.; Chapelle, O. Bounds on error expectation for support vector machines. *Neural Comput.* **2000**, *12*, 2013–2036. [CrossRef] [PubMed]
217. Vapnik, V.N. *Statistical Learning Theory*; Wiley: New York, NY, USA, 1998.
218. Burges, C.J.C. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **1998**, *2*, 121–167. [CrossRef]
219. Cortes, C.; Vapnik, V. Support vector networks. *Mach. Learn.* **1995**, *20*, 1–25. [CrossRef]
220. Wu, Q.; Zhou, D.-X. SVM soft margin classifiers: Linear programming versus quadratic programming. *Neural Comput.* **2005**, *17*, 1160–1187. [CrossRef]
221. Fine, S.; Scheinberg, K. Efficient SVM training using low-rank kernel representations. *J. Mach. Learn. Res.* **2001**, *2*, 243–264.
222. Ferris, M.C.; Munson, T.S. *Interior Point Methods for Massive Support Vector Machines*; Technical Report 00-05; Computer Sciences Department, University of Wisconsin: Madison, WI, USA, 2000.
223. Scheinberg, K. An efficient implementation of an active set method for SVMs. *J. Mach. Learn. Res.* **2006**, *7*, 2237–2257.
224. Osuna, E.; Freund, R.; Girosi, F. An improved training algorithm for support vector machines. In Proceedings of the IEEE Workshop on Neural Networks for Signal Processing, Amelia Island, FL, USA, 24–26 September 1997; pp. 276–285.
225. Platt, J. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods—Support Vector Learning*; Scholkopf, B., Burges, C., Smola, A., Eds.; MIT Press: Cambridge, MA, USA, 1999; pp. 185–208.
226. Joachims, T. Making large-scale SVM learning practical. In *Advances in Kernel Methods—Support Vector Learning*; Scholkopf, B., Burges, C.J.C, Smola, A.J., Eds.; MIT Press: Cambridge, MA, USA, 1999; pp. 169–184.
227. Hastie, T.; Rosset, S.; Tibshirani, R.; Zhu, J. The entire regularization path for the support vector machine. *J. Mach. Learn. Res.* **2004**, *5*, 1391–1415.
228. Collobert, R.; Bengio, S. SVMTorch: Support vector machines for large-scale regression problems. *J. Mach. Learn. Res.* **2001**, *1*, 143–160.
229. Vishwanathan, S.V.N.; Smola, A.J.; Murty, M.N. SimpleSVM. In Proceedings of the 20th International Conference on Machine Learning, Washington, DC, USA, 21–24 August 2003; pp. 760–767.
230. Keerthi, S.S.; Shevade, S.K.; Bhattacharyya, C.; Murthy, K.R.K. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Comput.* **2001**, *13*, 637–649. [CrossRef]
231. Lin, Y.-L.; Hsieh, J.-G.; Wu, H.-K.; Jeng, J.-H. Three-parameter sequential minimal optimization for support vector machines. *Neurocomputing* **2011**, *74*, 3467–3475. [CrossRef]
232. Hush, D.; Scovel, C. Polynomial-time decomposition algorithms for support vector machines. *Mach. Learn.* **2003**, *51*, 51–71. [CrossRef]
233. Kao, W.-C.; Chung, K.-M.; Sun, C.-L.; Lin, C.-J. Decomposition methods for linear support vector machines. *Neural Comput.* **2004**, *16*, 1689–1704. [CrossRef]
234. Chang, C.-C.; Lin, C.-J. *LIBSVM: A Library for Support Vector Machines*; Technical Report; Department of Computer Science and Information Engineering, National Taiwan University: Taipei City, Taiwan, 2001.
235. Fan, R.-E.; Chen, P.-H.; Lin, C.-J. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.* **2005**, *6*, 1889–1918.
236. Glasmachers, T.; Igel, C. Maximum-gain working set selection for SVMs. *J. Mach. Learn. Res.* **2006**, *7*, 1437–1466.
237. Steinwart, I.; Thomann, P. liquidSVM: A fast and versatile SVM package. *arXiv* **2017**, arXiv:1702.06899.
238. Wen, Z.; Shi, J.; Li, Q.; He, B.; Chen, J. ThunderSVM: A fast SVM library on GPUs and CPUs. *J. Mach. Learn. Res.* **2018**, *19*, 797–801.
239. Navia-Vazquez, A. Support vector perceptrons. *Neurocomputing* **2007**, *70*, 1089–1095. [CrossRef]
240. Tsang, I.W.; Kwok, J.T.; Cheung, P.-M. Core vector machines: Fast SVM training on very large data sets. *J. Mach. Learn. Res.* **2005**, *6*, 363–392.
241. Tax, D.M.J.; Duin, R.P.W. Support vector domain description. *Pattern Recogn. Lett.* **1999**, *20*, 1191–1199. [CrossRef]
242. Chang, C.C.; Lin, C.J. Training ν-support vector regression: Theory and algorithms. *Neural Comput.* **2002**, *14*, 1959–1977. [CrossRef] [PubMed]
243. Loosli, G.; Canu, S. Comments on the core vector machines: Fast SVM training on very large data sets. *J. Mach. Learn. Res.* **2007**, *8*, 291–301.

244. Tsang, I.W.-H.; Kwok, J.T.-Y.; Zurada, J.M. Generalized core vector machines. *IEEE Trans. Neural Netw.* **2006**, *17*, 1126–1140. [CrossRef]
245. Galvan, G.; Lapucci, M.; Lin, C.-J.; Sciandrone, M. A two-level decomposition framework exploiting First and second order information for SVM training problems. *J. Mach. Learn. Res.* **2021**, *22*, 1–38.
246. Li, Q.; Wen, Z.; He, B. Adaptive kernel value caching for SVM training. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 2376–2386. [CrossRef]
247. Meister, M.; Steinwart, I. Optimal Learning Rates for Localized SVMs. *J. Mach. Learn. Res.* **2016**, *17*, 1–44.
248. Thomann, P.; Blaschzyk, I.; Meister, M.; Steinwart, I. Spatial decompositions for large scale SVMs. *Int. Conf. Artif. Intell. Statist. (AISTATS)* **2017**, *54*, 1329–1337.
249. Blaschzyk, I.; Steinwart, I. Improved classification rates for localized SVMs. *J. Mach. Learn. Res.* **2020**, *23*, 1–59
250. Lin, C.-J. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Trans. Neural Netw.* **2022**, *13*, 248–250.
251. Keerthi, S.S.; Gilbert, E.G. Convergence of a generalized SMO algorithm for SVM classifier design. *Mach. Learn.* **2002**, *46*, 351–360. [CrossRef]
252. Lin, C.-J. On the convergence of the decomposition method for support vector machines. *IEEE Trans. Neural Netw.* **2001**, *12*, 1288–1298. [PubMed]
253. Chen, P.-H.; Fan, R.-E.; Lin, C.-J. A study on SMO-type decomposition methods for support vector machines. *IEEE Trans. Neural Netw.* **2006**, *17*, 893–908. [CrossRef]
254. Takahashi, N.; Nishi, T. Global convergence of decomposition learning methods for support vector machines. *IEEE Trans. Neural Netw.* **2006**, *17*, 1362–1369. [CrossRef]
255. Haasdonk, B. Feature space interpretation of SVMs with indefinite kernels. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 482–492. [CrossRef]
256. Suykens, J.A.K.; Vandewalle, J. Least squares support vector machine classifiers. *Neural Process. Lett.* **1999**, *9*, 293–300. [CrossRef]
257. Suykens, J.A.K.; Lukas, L.; Vandewalle, J. Sparse approximation using least squares support vector machines. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Genvea, Switzerland, 28–31 May 2000; Volume 2, pp. 757–760.
258. Chu, W.; Ong, C.J.; Keerthy, S.S. An improved conjugate gradient method scheme to the solution of least squares SVM. *IEEE Trans. Neural Netw.* **2005**, *16*, 498–501. [CrossRef]
259. Suykens, J.A.K.; Lukas, L.; Van Dooren, P.; De Moor, B.; Vandewalle, J. Least squares support vector machine classifiers: A large scale algorithm. In Proceedings of the European Conference on Circuit Theory and Design, Stresa, Italy, 29 August–2 September 1999; pp. 839–842.
260. Keerthi, S.S.; Shevade, S.K. SMO for least squares SVM formulations. *Neural Comput.* **2003**, *15*, 487–507. [CrossRef]
261. Li, B.; Song, S.; Li, K. A fast iterative single data approach to training unconstrained least squares support vector machines. *Neurocomputing* **2013**, *115*, 31–38. [CrossRef]
262. Jiao, L.; Bo, L.; Wang, L. Fast sparse approximation for least squares support vector machine. *IEEE Trans. Neural Netw.* **2007**, *18*, 685–697. [CrossRef] [PubMed]
263. Suykens, J.A.K.; De Brabanter, J.; Lukas, L.; Vandewalle, J. Weighted least squares support vector machines: Robustness and sparse approximation. *Neurocomputing* **2002**, *48*, 85–105. [CrossRef]
264. Wang, Z.; Chen, S. New least squares support vector machines based on matrix patterns. *Neural Process. Lett.* **2007**, *26*, 41–56. [CrossRef]
265. Perez-Cruz, F.; Navia-Vazquez, A.; Rojo-Alvarez, J.L.; Artes-Rodriguez, A. A new training algorithm for support vector machines. In Proceedings of the Fifth Bayona Workshop on Emerging Technologies in Telecommunications, Baiona, Spain, 6–8 September 1999; pp. 116–120.
266. Xu, G.; Hu, B.-G.; Principe, J.C. Robust C-loss kernel classifiers. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 510–522. [CrossRef]
267. Seeger, M. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In Proceedings of the 12th International Conference on Neural Information Processing Systems, Denver, CO, USA, 29 November–4 December 1999; pp. 603–609.
268. Gestel, T.V.; Sukens, J.A.; Lanckriet, G.; Lambrechts, A.; De Moor, B.; Vandewalle, J. Bayesian framework for least-squares support vector machine classifiers, Gaussian processes, and kernel Fisher discriminant analysis. *Neural Comput.* **2002**, *14*, 1115–1147. [CrossRef]
269. Smola, A.J.; Scholkopf, B. Sparse greedy matrix approximation for machine learning. In Proceedings of the the 17th International Conference on Machine Learning, San Francisco, CA, USA, 29 June–2 July 2000; pp. 911–918.
270. Lee, Y.J.; Mangasarian, O.L. RSVM: Reduced support vector machines. In Proceedings of the 1st SIAM International Conference on Data Mining, Chicago, IL, USA, 5–7 April 2001; pp. 1–17.
271. Mangasarian, O.L. Generalized support vector machines. In *Advances in Large Margin Classifiers*; Smola, A., Bartlett, P., Scholkopf, B., Schuurmans, D., Eds.; MIT Press: Cambridge, MA, USA, 2000; pp. 135–146.
272. Lee, Y.J.; Mangasarian, O.L. SSVM: A smooth support vector machine. *Comput. Optim. Applic.* **2001**, *20*, 5–22. [CrossRef]
273. Lee, Y.-J.; Hsieh, W.-F.; Huang, C.-M. ε-SSVR: A smooth support vector machine for ε-insensitive regression. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 678–685.

274. Fung, G.; Mangasarian, O. Proximal support vector machines. In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 26–29 August 2001; pp. 77–86.
275. Mangasarian, O.L.; Musicant, D.R. Lagrangian support vector machines. *J. Mach. Learn. Res.* **2001**, *1*, 161–177.
276. Musicant, D.R.; Feinberg, A. Active set support vector regression. *IEEE Trans. Neural Netw.* **2004**, *15*, 268–275. [CrossRef]
277. Wu, M.; Scholkopf, B.; Bakir, G. A direct method for building sparse kernel learning algorithms. *J. Mach. Learn. Res.* **2006**, *7*, 603–624.
278. Bennett, K.P.; Mangasarian, O.L. Robust linear programming discrimination of two linearly inseparable sets. *Optim. Methods Softw.* **1992**, *1*, 23–34. [CrossRef]
279. Mangasarian, O.L.; Wild, E.W. Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 69–74. [CrossRef] [PubMed]
280. Ye, Q.; Zhao, C.; Ye, N.; Chen, Y. Multi-weight vector projection support vector machines. *Pattern Recognit. Lett.* **2010**, *31*, 2006–2011. [CrossRef]
281. Geng, C.; Chen, S. Multiplane convex proximal support vector machine. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 4918–4931. [CrossRef]
282. Jayadeva; Khemchandani, R.; Chandra, S. Twin support vector machines for pattern classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 905–910. [CrossRef]
283. Ghorai, S.; Mukherjee, A.; Dutta, P.K. Nonparallel plane proximal classifier. *Signal Process.* **2009**, *89*, 510–522. [CrossRef]
284. Shao, Y.-H.; Zhang, C.-H.; Wang, X.-B.; Deng, N.-Y. Improvements on twin support vector machines. *IEEE Trans. Neural Netw.* **2011**, *22*, 962–968. [CrossRef]
285. Tian, Y.; Ju, X.; Qi, Z.; Shi, Y. Improved twin support vector machine. *Sci. China Math.* **2014**, *57*, 417–432. [CrossRef]
286. Peng, X.; Xu, D. Twin support vector hypersphere (TSVH) classifier for pattern recognition. *Neural Comput. Appl.* **2014**, *24*, 1207–1220. [CrossRef]
287. Mehrkanoon, S.; Huang, X.; Suykens, J.A.K. Non-parallel support vector classifiers with different loss functions. *Neurocomputing* **2014**, *143*, 294–301. [CrossRef]
288. Xu, Y.; Yang, Z.; Pan, X. A novel twin support-vector machine with pinball loss. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 359–370. [CrossRef]
289. Shao, Y.-H.; Deng, N.-Y. A coordinate descent margin based-twin support vector machine for classification. *Neural Netw.* **2012**, *25*, 114–121. [CrossRef] [PubMed]
290. Ghaoui, L.E.; Viallon, V.; Rabbani, T. Safe feature elimination for the LASSO and sparse supervised learning problems. *Pacific J. Optim.* **2010**, *8*, 667–698.
291. Pan, X.; Yang, Z.; Xu, Y.; Wang, L. Safe screening rules for accelerating twin support vector machine classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 1876–1887. [CrossRef]
292. Qi, K.; Yang, H. Elastic net nonparallel hyperplane support vector machine and its geometrical rationality. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 7199–7209. [CrossRef]
293. Shao, Y.-H.; Chen, W.-J.; Deng, N.-Y. Nonparallel hyperplane support vector machine for binary classification problems. *Inf. Sci.* **2014**, *263*, 22–35. [CrossRef]
294. Huang, X.; Shi, L.; Suykens, J.A.K. Support vector machine classifier with pinball loss. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 984–997. [CrossRef]
295. Deng, N.Y.; Tian, Y.J.; Zhang, C.H. *Support Vector Machines: Theory Algorithms and Extensions*; CRC Press: Boca Raton, FL, USA, 2012.
296. Tian, Y.; Qi, Z.; Ju, X.; Shi, Y.; Liu, X. Nonparallel support vector machines for pattern classification. *IEEE Trans. Cybern.* **2014**, *44*, 1067–1079. [CrossRef]
297. Shen, X.; Niu, L.; Qi, Z.; Tian, Y. Support vector machine classifier with truncated pinball loss. *Pattern Recogn.* **2017**, *68*, 199–210. [CrossRef]
298. Tanveer, M.; Tiwari, A.; Choudhary, R.; Ganaie, M.A. Large-scale pinball twin support vector machines. *Mach. Learn.* **2022**, *111*, 3525–3548. [CrossRef]
299. Hao, P.-Y.; Chiang, J.-H.; Chen, Y.-D. Possibilistic classification by support vector networks. *Neural Netw.* **2022**, *149*, 40–56. [CrossRef] [PubMed]
300. Chen, S.; Cao, J.; Chen, F.; Liu, B. Entropy-based fuzzy least squares twin support vector machine for pattern classification. *Neural Process. Lett.* **2020**, *51*, 41–66. [CrossRef]
301. Liu, L.; Li, S.; Zhang, X.; Dai, Z.; Zhu, Y. Polycentric intuitionistic fuzzy weighted least squares twin SVMs. *Neurocomputing* **2024**, *609*, 128475. [CrossRef]
302. Ganaie, M.A.; Tanveer, M.; Lin, C.-T. Large-scale fuzzy least squares twin SVMs for class imbalance learning. *IEEE Trans. Fuzzy Syst.* **2022**, *30*, 4815–4827. [CrossRef]
303. Tanveer, M.; Ganaie, M.A.; Bhattacharjee, A.; Lin, C.T. Intuitionistic fuzzy weighted least squares twin SVMs. *IEEE Trans. Cybern.* **2023**, *53*, 4400–4409. [CrossRef]
304. Moslemnejad, S.; Hamidzadeh, J. A hybrid method for increasing the speed of SVM training using belief function theory and boundary region. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 3557–3574. [CrossRef]

305. Hamidzadeh, J.; Moslemnejad, S. Identification of uncertainty and decision boundary for SVM classification training using belief function. *Appl. Intell.* **2019**, *49*, 2030–2045. [CrossRef]
306. Peng, S.; Wang, W.; Chen, Y.; Zhong, X.; Hu, Q. Regression-based hyperparameter learning for support vector machines. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 18799–18813.. [CrossRef]
307. Scholkopf, B.; Smola, A.J.; Williamson, R.C.; Bartlett, P.L. New support vector algorithm. *Neural Comput.* **2000**, *12*, 1207–1245. [CrossRef]
308. Ikeda, K.; Murata, N. Geometrical properties of Nu support vector machines with different norms. *Neural Comput.* **2005**, *17*, 2508–2529. [CrossRef]
309. Hao, P.-Y. New support vector algorithms with parametric insensitive/margin model. *Neural Netw.* **2010**, *23*, 60–73. [CrossRef] [PubMed]
310. Barbero, A.; Takeda, A.; Lopez, J. Geometric intuition and algorithms for Eν-SVM. *J. Mach. Learn. Res.* **2015**, *16*, 323–369.
311. Davenport, M.A.; Baraniuk, R.G.; Scott, C.D. Tuning support vector machines for minimax and Neyman-Pearson classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1888–1898. [CrossRef]
312. Osuna, E.; Freund, R.; Girosi, F. *Support Vector Machines: Training and Applications*; Technical Report A.I. Memo No. 1602; MIT Artificial Intelligence Laboratory: Cambridge, MA, USA, 1997.
313. Chew, H.G.; Bogner, R.E.; Lim, C.C. Dual-ν support vector machine with error rate and training size biasing. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Salt Lake City, UT, USA, 7–11 May 2001; pp. 1269–1272.
314. Peng, X. A ν-twin support vector machine (ν-TSVM) classifier and its geometric algorithms. *Inf. Sci.* **2010**, *180*, 3863–3875. [CrossRef]
315. Khemchandan, R.; Saigal, P.; Chandra, S. Improvements on ν-twin support vector machine. *Neural Netw.* **2016**, *79*, 97–107. [CrossRef]
316. Teo, C.H.; Smola, A.; Vishwanathan, S.V.; Le, Q.V. A scalable modular convex solver for regularized risk minimization. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), San Jose, CA, USA, 12–15 August 2007; pp. 727–736.
317. Teo, C.H.; Vishwanthan, S.V.N.; Smola, A.; Le, Q. Bundle methods for regularized risk minimization. *J. Mach. Learn. Res.* **2010**, *11*, 311–365.
318. Joachims, T. Training linear SVMs in linear time. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006; pp. 217–226.
319. Franc, V.; Sonnenburg, S. Optimized cutting plane algorithm for large-scale risk minimization. *J. Mach. Learn. Res.* **2009**, *10*, 2157–2192.
320. Joachims, T.; Finley, T.; Yu, C.-N.J. Cutting-plane training of structural SVMs. *Mach. Learn.* **2009**, *77*, 27–59. [CrossRef]
321. Joachims, T.; Yu, C.-N.J. Sparse kernel SVMs via cutting-plane training. *Mach. Learn.* **2009**, *76*, 179–193. [CrossRef]
322. Bordes, A.; Ertekin, S.; Wesdon, J.; Bottou, L. Fast kernel classifiers for online and active learning. *J. Mach. Learn. Res.* **2005**, *6*, 1579–1619.
323. Tsang, I.W.; Kocsor, A.; Kwok, J.T. Simpler core vector machines with enclosing balls. In Proceedings of the the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 911–918.
324. Mangasarian, O.L.; Musicant, D.R. Successive overrelaxation for support vector machines. *IEEE Trans. Neural Netw.* **1999**, *10*, 1032–1037. [CrossRef] [PubMed]
325. Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; Lin, C.-J. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.* **2008**, *9*, 1871–1874.
326. Zhang, T.; Oles, F.J. Text categorization based on regularized linear classification methods. *Inf. Retr.* **2001**, *4*, 5–31. [CrossRef]
327. Chang, K.-W.; Hsieh, C.-J.; Lin, C.-J. Coordinate descent method for large-scale L2-loss linear support vector machines. *J. Mach. Learn. Res.* **2008**, *9*, 1369–1398.
328. Schraudolph, N.; Yu, J.; Gunter, S. A stochastic quasi-Newton method for online convex optimization. In Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AIstats), San Juan, Puerto Rico, 21–24 March 2007; Society for AIstats: Indio, CA, USA, 2007; pp. 433–440.
329. Bordes, A.; Bottou, L.; Gallinari, P.; Chang, J.; Smith, S.A. Erratum: SGDQN is less careful than expected. *J. Mach. Learn. Res.* **2010**, *11*, 2229–2240.
330. Bordes, A.; Bottou, L.; Gallinari, P. SGD-QN: Careful quasi-Newton stochastic gradient descent. *J. Mach. Learn. Res.* **2009**, *10*, 1737–1754.
331. Keerthi, S.S.; DeCoste, D. A modified finite Newton method for fast solution of large scale linear SVMs. *J. Mach. Learn. Res.* **2005**, *6*, 341–361.
332. Mangasarian, O.L. A finite Newton method for classification. *Optim. Methods Softw.* **2002**, *17*, 913–929. [CrossRef]
333. Lin, C.-J.; Weng, R.C.; Keerthi, S.S. Trust region Newton method for logistic regression. *J. Mach. Learn. Res.* **2008**, *9*, 627–650.
334. Chapelle, O. Training a support vector machine in the primal. *Neural Comput.* **2007**, *19*, 1155–1178. [CrossRef]
335. Hush, D.; Kelly, P.; Scovel, C.; Steinwart, I. QP algorithms with guaranteed accuracy and run time for support vector machines. *J. Mach. Learn. Res.* **2006**, *7*, 733–769.

336. Shalev-Shwartz, S.; Singer, Y.; Srebro, N. Pegasos: Primal estimated sub-gradient solver for SVM. In Proceedings of the 24th International Conference on Machine Learning (ICML), Corvalis, OR, USA, 20–24 June 2007; ACM Press: New York, NY, USA, 2007; pp. 807–814.
337. Bottou, L.; Bousquet, O. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2007; Volume 20, pp. 161–168.
338. Ito, N.; Takeda, A.; Toh, K.-C. A unified formulation and fast accelerated proximal gradient method for classification. *J. Mach. Learn. Res.* **2017**, *18*, 1–49.
339. Belkin, M.; Niyogi, P.; Sindhwani, V. Manifold regularization: A geometric framework for learning from examples. *J. Mach. Learn. Res.* **2006**, *7*, 2399–2434.
340. Melacci, S.; Belkin, M. Laplacian support vector machines trained in the primal. *J. Mach. Learn. Res.* **2011**, *12*, 1149–1184.
341. Qi, Z.; Tian, Y.; Shi, Y. Successive overrelaxation for Laplacian support vector machine. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 674–683. [CrossRef]
342. Yu, H.; Yang, J.; Han, J.; Li, X. Making SVMs scalable to large data sets using hierarchical cluster indexing. *Data Mining Knowl. Discov.* **2005**, *11*, 295–321. [CrossRef]
343. Kramer, K.A.; Hall, L.O.; Goldgof, D.B.; Remsen, A.; Luo, T. Fast support vector machines for continuous data. *IEEE Trans. Syst. Man Cybern. B* **2009**, *39*, 989–1001. [CrossRef]
344. Aiolli, F.; Sperduti, A. Multiclass classification with multi-prototype support vector machines. *J. Mach. Learn. Res.* **2005**, *6*, 817–850.
345. Angiulli, F.; Astorino, A. Scaling up support vector machines using nearest neighbor condensation. *IEEE Trans. Neural Netw.* **2010**, *21*, 351–357. [CrossRef]
346. Knebel, T.; Hochreiter, S.; Obermayer, K. An SMO algorithm for the potential support vector machine. *Neural Comput.* **2008**, *20*, 271–287. [CrossRef]
347. Glasmachers, T.; Igel, C. Second-order SMO improves SVM online and active learning. *Neural Comput.* **2008**, *20*, 374–382. [CrossRef] [PubMed]
348. Chang, F.; Guo, C.-Y.; Lin, X.-R.; Lu, C.-J. Tree decomposition for large-scale SVM problems. *J. Mach. Learn. Res.* **2010**, *11*, 2935–2972.
349. Zheng, J.; Lu, B.-L. A support vector machine classifier with automatic confidence and its application to gender classification. *Neurocomputing* **2011**, *74*, 1926–1935. [CrossRef]
350. Tan, Y.; Wang, J. A support vector machine with a hybrid kernel and minimal Vapnik-Chervonenkis dimension. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 385–395.
351. Roobaert, D. DirectSVM: A simple support vector machine perceptron. *J. VLSI Signal Process.* **2002**, *32*, 147–156. [CrossRef]
352. Grinblat, G.L.; Uzal, L.C.; Ceccatto, H.A.; Granitto, P.M. Solving nonstationary classification problems with coupled support vector machines. *IEEE Trans. Neural Netw.* **2011**, *22*, 37–51. [CrossRef]
353. Shi, Y.; Chung, F.-L.; Wang, S. An improved TA-SVM method without matrix inversion and its fast implementation for nonstationary datasets. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2005–2018. [CrossRef]
354. Huang, K.; Zheng, D.; Sun, J.; Hotta, Y.; Fujimoto, K.; Naoi, S. Sparse learning for support vector classification. *Pattern Recogn. Lett.* **2010**, *31*, 1944–1951. [CrossRef]
355. Marchand, M.; Shawe-Taylor, J. The set covering machine. *J. Mach. Learn. Res.* **2022**, *3*, 723–746.
356. Huang, K.; Yang, H.; King, I.; Lyu, M.R. Maxi-min margin machine: Learning large margin classifiers locally and globally. *IEEE Trans. Neural Netw.* **2008**, *19*, 260–272. [CrossRef]
357. Klement, S.; Anders, S.; Martinetz, T. The support feature machine: Classification with the least number of features and application to neuroimaging data. *Neural Netw.* **2013**, *25*, 1548–1584. [CrossRef] [PubMed]
358. Nandan, M.; Khargonekar, P.P.; Talathi, S.S. Fast SVM training using approximate extreme points. *J. Mach. Learn. Res.* **2014**, *15*, 59–98.
359. Huang, K.; Jiang, H.; Zhang, X.-Y. Field support vector machines. *IEEE Trans. Emerg. Top. Comput. Intell.* **2017**, *1*, 454 –463. [CrossRef]
360. Bouboulis, P.; Theodoridis, S.; Mavroforakis, C.; Evaggelatou-Dalla, L. Complex support vector machines for regression and quaternary classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1260–1274. [CrossRef]
361. Blanco, V.; Puerto, J.; Rodriguez-Chia, A.M. On ℓ_p-support vector machines and multidimensional kernels. *J. Mach. Learn. Res.* **2020**, *21*, 1–29.
362. Li, C.N.; Shao, Y.H.; Deng, N.Y. Robust L1-norm non-parallel proximal support vector machine. *Optimization* **2016**, *65*, 169–183. [CrossRef]
363. Ye, Q.; Zhao, H.; Li, Z.; Yang, X.; Gao, S.; Yin, T.; Ye, N. L1-Norm distance minimization-based fast robusttwin support vector *k*-plane clustering. *lEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 4494–4503. [CrossRef]
364. Lan, L.; Wang, Z.; Zhe, S.; Cheng, W.; Wang, J.; Zhang, K. Scaling Up Kernel SVM on Limited Resources: A Low-Rank Linearization Approach. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 369–378 [CrossRef]
365. Zhou, S.; Zhou, W. Unifed SVM algorithm based on LS-DC loss. *Mach. Learn.* **2023**, *112*, 2975–3002. [CrossRef]
366. Yoshida, R.; Takamori, M.; Matsumoto, H.; Miura, K. Tropical support vector machines: Evaluations and extension to function spaces. *Neural Netw.* **2023**, *157*, 77–89. [CrossRef]

367. Farquhar, J.; Hardoon, D.; Meng, H.; Shawe-taylor, J.; Szedmak, S. Two view learning: SVM-2K, theory and practice. In Proceedings of the Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 5–8 December 2005; pp. 355–362.
368. Wang, H.; Zhu, J.; Zhang, S. Safe screening rules for multi-view support vector machines. *Neural Netw.* **2023**, *166*, 326–343. [CrossRef] [PubMed]
369. Tipping, M.E. Sparse Bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.* **2001**, *1*, 211–244.
370. Tipping, M.E.; Faul, A.C. Fast marginal likelihood maximisation for sparse Bayesian models. In Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics, Key West, FL, USA, 3–6 January 2003; pp. 3–6.
371. Chen, H.; Tino, P.; Yao, X. Probabilistic classification vector machines. *IEEE Trans. Neural Netw.* **2009**, *20*, 901–914. [CrossRef]
372. Chen, H.; Tino, P.; Yao, X. Efficient probabilistic classification vector machine with incremental basis function selection. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 356–369. [CrossRef]
373. Lyu, S.; Tian, X.; Li, Y.; Jiang, B.; Chen, H. Multiclass probabilistic classification vector machine. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 3906–3919. [CrossRef]
374. Rebentrost, P.; Mohseni, M.; Lloyd, S. Quantum support vector machine for big data classification. *Phys. Rev. Lett.* **2014**, *113*, 130503. [CrossRef]
375. Ding, C.; Bao, T.-Y.; Huang, H.-L. Quantum-Inspired Support Vector Machine. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 7210–7222. [CrossRef]
376. Keerthi, S.S.; Chapelle, O.; DeCoste, D. Building support vector machines with reduced classifier complexity. *J. Mach. Learn. Res.* **2006**, *7*, 1493–1515.
377. Ertekin, S.; Bottou, L.; Giles, C.L. Nonconvex online support vector machines. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 368–381. [CrossRef]
378. Downs, T.; Gates, K.E.; Masters, A. Exact simplification of support vector solutions. *J. Mach. Learn. Res.* **2001**, *2*, 293–297.
379. Liang, X.; Chen, R.-C.; Guo, X. Pruning support vector machines without altering performances. *IEEE Trans. Neural Netw.* **2008**, *19*, 1792–1803. [CrossRef] [PubMed]
380. Pontil, M.; Verri, A. Properties of support vector machines. *Neural Comput.* **1998**, *10*, 955–974. [CrossRef]
381. Liang, X. An effective method of pruning support vector machine classifiers. *IEEE Trans. Neural Netw.* **2010**, *21*, 26–38. [CrossRef] [PubMed]
382. Burges, C.J.C. Simplified support vector decision rules. In Proceedings of the 13th International Conference on Machine Learning, Bari, Italy, 3–6 July 1996; pp. 71–77.
383. Nguyen, D.; Ho, T. A bottom-up method for simplifying support vector solutions. *IEEE Trans. Neural Netw.* **2006**, *17*, 792–796. [CrossRef]
384. Shin, H.; Cho, S. Neighborhood property–based pattern selection for support vector machines. *Neural Comput.* **2007**, *19*, 816–855. [CrossRef]
385. Hong, B.; Zhang, W.; Liu, W.; Ye, J.; Cai, D.; Wang, J. Scaling up sparse support vector machines by simultaneous feature and sample reduction. *J. Mach. Learn. Res.* **2019**, *20*, 1–39.
386. de Kruif, B.J.; de Vries, T.J.A. Pruning error minimization in least squares support vector machines. *IEEE Trans. Neural Netw.* **2003**, *14*, 696–702. [CrossRef]
387. Kuh, A.; De Wilde, P. Comments on pruning error minimization in least squares support vector machines. *IEEE Trans. Neural Netw.* **2007**, *18*, 606–609. [CrossRef]
388. Zeng, X.Y.; Chen, X.W. SMO-based pruning methods for sparse least squares support vector machines. *IEEE Trans. Neural Netw.* **2005**, *16*, 1541–1546. [CrossRef]
389. Yang, X.; Lu, J.; Zhang, G. Adaptive pruning algorithm for least squares support vector machine classifier. *Soft Comput.* **2010**, *14*, 667–680. [CrossRef]
390. Yang, J.; Bouzerdoum, A.; Phung, S.L. A training algorithm for sparse LS-SVM using compressive sampling. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, TX, USA, 14–19 March 2010; pp. 2054–2057.
391. Zhou, S. Sparse LSSVM in primal using Cholesky factorization for large-scale problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 783–795. [CrossRef] [PubMed]
392. Mall, R.; Suykens, J.A.K. Very sparse LSSVM reductions for large-scale data. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1086–1097. [CrossRef] [PubMed]
393. Ojeda, F.; Suykens, J.A.K.; Moor, B.D. Low rank updated LS-SVM classifiers for fast variable selection. *Neural Netw.* **2008**, *21*, 437–449. [CrossRef]
394. Ma, Y.; Liang, X.; Sheng, G.; Kwok, J.T.; Wang, M.; Li, G. Noniterative sparse LS-SVM based on globally Representative Point Selection. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 788–798. [CrossRef]
395. Weston, J.; Watkins, C. Multi-class support vector machines.In Proceedings of the European Symposium on Artificial Neural Networks, Bruges, Belgium, 21–23 April 1999; Verleysen, M., Ed.; D. Facto Press: Brussels, Belgium, 1999.
396. Crammer, K.; Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.* **2001**, *2*, 265–292.
397. Rifkin, R.; Klautau, A. In defense of one-vs-all classification. *J. Mach. Learn. Res.* **2004**, *5*, 101–141.

398. Kressel, U.H.-G. Pairwise classification and support vector machines. In *Advances in Kernel Methods—Support Vector Learning*; Scholkopf, B., Burges, C.J.C., Smola, A.J., Eds.; MIT Press: Cambridge, MA, USA, 1999; pp. 255–268.
399. Dietterich, T.; Bakiri, G. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Intell. Res.* **1995**, *2*, 263–286. [CrossRef]
400. Allwein, E.L.; Schapire, R.E.; Singer, Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *J. Mach. Learn. Res.* **2000**, *1*, 113–141.
401. Hsu, C.-W.; Lin, C.-J. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Netw.* **2002**, *13*, 415–425.
402. Cheong, S.; Oh, S.H.; Lee, S.-Y. Support vector machines with binary tree architecture for multi-class classification. *Neural Inf. Process.–Lett. Rev.* **2004**, *2*, 47–51.
403. Fei, B.; Liu, J. Binary tree of SVM: A new fast multiclass training and classification algorithm. *IEEE Trans. Neural Netw.* **2006**, *17*, 696–704. [CrossRef] [PubMed]
404. Mesquita, D.P.P.; Freitas, L.A.; Gomes, J.P.P.; Mattos, C.L.C. LS-SVR as a Bayesian RBF network. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4389–4393. [CrossRef] [PubMed]
405. Saunders, C.; Gammerman, A.; Vovk, V. Ridge regression learning algorithm in dual variables. In Proceedings of the 15th International Conference on Machine Learning (ICML), Madison, WI, USA, 24–27 July 1998; pp. 515–521.
406. Rasmussen, C.E.; Williams, C.K.I. *Gaussian Processes for Machine Learning*; MIT Press: Cambridge, MA, USA, 2006.
407. Gao, J.B.; Gunn, S.R.; Harris, C.J.; Brown, M. A probabilistic framework for SVM regression and error bar estimation. *Mach. Learn.* **2002**, *46*, 71–89. [CrossRef]
408. Dufrenois, F.; Colliez, J.; Hamad, D. Bounded influence support vector regression for robust single-model estimation. *IEEE Trans. Neural Netw.* **2009**, *20*, 1689–1706. [CrossRef]
409. Chang, M.-W.; Lin, C.-J. Leave-one-out bounds for support vector regression model selection. *Neural Comput.* **2005**, *17*, 1188–1222. [CrossRef]
410. Wang, G.; Yeung, D.-Y.; Lochovsky, F.H. A new solution path algorithm in support vector regression. *IEEE Trans. Neural Netw.* **2008**, *19*, 1753–1767. [CrossRef]
411. Gunter, L.; Zhu, J. Efficient computation and model selection for the support vector regression. *Neural Comput.* **2007**, *19*, 1633–1655. [CrossRef]
412. Shevade, S.K.; Keerthi, S.S.; Bhattacharyya, C.; Murthy, K.R.K. Improvements to the SMO algorithm for SVM regression. *IEEE Trans. Neural Netw.* **2000**, *11*, 1188–1193. [CrossRef]
413. Flake, G.W.; Lawrence, S. Efficient SVM regression training with SMO. *Mach. Learn.* **2002**, *46*, 271–290. [CrossRef]
414. Takahashi, N.; Guo, J.; Nishi, T. Global convergence of SMO algorithm for support vector regression. *IEEE Trans. Neural Netw.* **2008**, *19*, 971–982. [CrossRef]
415. Peng, X. TSVR: An efficient twin support vector machine for regression. *Neural Netw.* **2010**, *23*, 365–372. [CrossRef]
416. Peng, X. Primal twin support vector regression and its sparse approximation. *Neurocomputing* **2010**, *73*, 2846–2858. [CrossRef]
417. Hao, P.-Y. Pair-v-SVR: A novel and efficient pairing v-support vector regression algorithm. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2503–2515. [CrossRef]
418. Karal, O. Maximum likelihood optimal and robust support vector regression with lncosh loss function. *Neural Netw.* **2017**, *94*, 1–12. [CrossRef]
419. Yang, H.; Huang, K.; King, I.; Lyu, M.R. Localized support vector regression for time series prediction. *Neurocomputing* **2009**, *72*, 2659–2669. [CrossRef]
420. Bo, L.; Wang, L.; Jiao, L. Recursive finite Newton algorithm for support vector regression in the primal. *Neural Comput.* **2007**, *19*, 1082–1096. [CrossRef]
421. Shashua, A.; Levin, A. Ranking with large margin principle: Two approaches. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2002; Volume 15, pp. 937–944.
422. Chu, W.; Keerthi, S.S. Support vector ordinal regression. *Neural Comput.* **2007**, *19*, 792–815. [CrossRef] [PubMed]
423. Lee, C.-P.; Lin, C.-J. Large-scale linear RankSVM. *Neural Comput.* **2014**, *26*, 781–817 [CrossRef]
424. Ben-Hur, A.; Horn, D.; Siegelmann, H.; Vapnik, V. Support vector clustering. *J. Mach. Learn. Res.* **2001**, *2*, 125–137. [CrossRef]
425. Scholkopf, B.; Platt, J.; Shawe-Taylor, J.; Smola, A.; Williamson, R. Estimating the support of a high-dimensional distribution. *Neural Comput.* **2001**, *13*, 1443–1471. [CrossRef]
426. Jung, K.-H.; Lee, D.; Lee, J. Fast support-based clustering method for large-scale problems. *Pattern Recogn.* **2010**, *43*, 1975–1983. [CrossRef]
427. Chiang, J.-H.; Hao, P.-Y. A new kernel-based fuzzy clustering approach: Support vector clustering with cell growing. *IEEE Trans. Fuzzy Syst.* **2003**, *11*, 518–527. [CrossRef]
428. Wang, Z.; Shao, Y.-H.; Bai, L.; Deng, N.-Y. Twin support vector machine for clustering. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 2583–2588. [CrossRef] [PubMed]
429. Xu, L.; Neufeld, J.; Larson, B.; Schuurmans, D. Maximum margin clustering. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2004; Volume 17.
430. Zhang, K.; Tsang, I.W.; Kwok, J.T. Maximum margin clustering made practical. *IEEE Trans. Neural Netw.* **2009**, *20*, 583–596. [CrossRef] [PubMed]

431. Valizadegan, H.; Jin, R. Generalized maximum margin clustering and unsupervised kernel learning. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2007; Volume 19, pp. 1417–1424.
432. Wang, F.; Zhao, B.; Zhang, C. Linear time maximum margin clustering. *IEEE Trans. Neural Netw.* **2010**, *21*, 319–332. [CrossRef]
433. Niu, G.; Dai, B.; Shang, L.; Sugiyama, M. Maximum volume clustering: A new discriminative clustering approach. *J. Mach. Learn. Res.* **2013**, *14*, 2641–2687.
434. Tax, D.M.J.; Duin, R.P.W. Support vector data description. *Mach. Learn.* **2004**, *54*, 45–66. [CrossRef]
435. Tax, D.M.J. One-Class Classification: Concept-Learning in the Absence of Counter-Examples. Ph.D. Dissertation, Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands, 2001.
436. Lee, K.Y.; Kim, D.-W.; Lee, K.H.; Lee, D. Density-induced support vector data description. *IEEE Trans. Neural Netw.* **2007**, *18*, 284–289. [CrossRef]
437. Manevitz, L.M.; Yousef, M. One-class SVMs for document classification. *J. Mach. Learn. Res.* **2001**, *2*, 139–154.
438. Choi, Y.-S. Least squares one-class support vector machine. *Pattern Recogn. Lett.* **2009**, *30*, 1236–1240. [CrossRef]
439. Cauwenberghs, G.; Poggio, T. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*; Leen, T.K., Dietterich, T.G., Tresp, V., Eds.; MIT Press: Cambridge, MA, USA, 2001; Volume 13, pp. 409–415
440. Friess, T.; Cristianini, N.; Campbell, C. The kernel-adatron algorithm: A fast and simple learning procedure for support vector machines. In Proceedings of the 15th International Conference on Machine Learning, Madison, WI, USA, 24–27 July 1998; pp. 188–196.
441. Ma, J.; Theiler, J.; Perkins, S. Accurate online support vector regression. *Neural Comput.* **2003**, *15*, 2683–2703. [CrossRef] [PubMed]
442. Martin, M. On-line support vector machine regression. In Proceedings of the 13th European Conference on Machine Learning, LNAI, Helsinki, Finland, 19–23 August 2002; Springer: Berlin, Germany, 2002; Volume 2430, pp. 282–294.
443. Gentile, C. A new approximate maximal margin classification algorithm. *J. Mach. Learn. Res.* **2001**, *2*, 213–242.
444. Laskov, P.; Gehl, C.; Kruger, S.; Muller, K.-R. Incremental support vector learning: Analysis, implementation and applications. *J. Mach. Learn. Res.* **2006**, *7*, 1909–1936.
445. Nguyen, D.D.; Matsumoto, K.; Takishima, Y.; Hashimoto, K. Condensed vector machines: Learning fast machine for large data. *IEEE Trans. Neural Netw.* **2010**, *21*, 1903–1914. [CrossRef]
446. Renjifo, C.; Barsic, D.; Carmen, C.; Norman, K.; Peacock, G.S. Improving radial basis function kernel classification through incremental learning and automatic parameter selection. *Neurocomputing* **2008**, *72*, 3–14. [CrossRef]
447. Shilton, A.; Palamiswami, M.; Ralph, D.; Tsoi, A. Incremental training of support vector machines. *IEEE Trans. Neural Netw.* **2005**, *16*, 114–131. [CrossRef]
448. Afshin, B.; Shiri, M.E.; Layeghi, K.; Javadi, H.H. Kernel optimization for reducing core vector machine classification error. *Neural Process. Lett.* **2023**, *55*, 10011–10036. [CrossRef]
449. Du, K.-L.; Swamy, M.N.S. *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*; Springer: New York, NY, USA, 2016.
450. Katagiri, S.; Abe, S. Incremental training of support vector machines using hyperspheres. *Pattern Recogn. Lett.* **2006**, *27*, 1495–1507. [CrossRef]
451. Orabona, F.; Castellini, C.; Caputo, B.; Jie, L.; Sandini, G. On-line independent support vector machines. *Pattern Recogn.* **2010**, *43*, 1402–1412. [CrossRef]
452. Gu, B.; Wang, J.-D.; Yu, Y.-C.; Zheng, G.-S.; Huang, Y.F.; Xu, T. Accurate on-line ν-support vector learning. *Neural Netw.* **2012**, *27*, 51–59. [CrossRef]
453. Gu, B.; Sheng, V.S.; Wang, Z.; Ho, D.; Osman, S.; Li, S. Incremental learning for ν-support vector regression. *Neural Netw.* **2015**, *67*, 140–150. [CrossRef] [PubMed]
454. Gu, B.; Sheng, V.S.; Tay, K.Y.; Romano, W.; Li, S. Incremental support vector learning for ordinal regression. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 1403–1416. [CrossRef]
455. Warmuth, M.K.; Liao, J.; Ratsch, G.; Mathieson, M.; Putta, S.; Lemmem, C. Support vector machines for active learning in the drug discovery process. *J. Chem. Inf. Sci.* **2003**, *43*, 667–673. [CrossRef]
456. Tong, S.; Koller, D. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.* **2001**, *2*, 45–66.
457. Chapelle, O.; Zien, A. Semi-supervised classification by low density separation. In Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, Bridgetown, Barbados, 6–8 January 2005; pp. 57–64.
458. Chapelle, O.; Sindhwani, V.; Keerthi, S.S. Optimization techniques for semi-supervised support vector machines. *J. Mach. Learn. Res.* **2008**, *9*, 203–233.
459. Fung, G.; Mangasarian, O. Semi-supervised support vector machines for unlabeled data classification. *Optim. Meth. Softw.* **2001**, *15*, 29–44. [CrossRef]
460. Joachims, T. Transductive inference for text classification using support vector machines. In Proceedings of the 16th International Conference on Machine Learning, Bled, Slovenia, 27–30 June 1999; Morgan Kaufmann: San Mateo, CA, USA, 1999; pp. 200–209.
461. Collobert, R.; Sinz, F.; Weston, J.; Bottou, L. Large scale transductive SVMs. *J. Mach. Learn. Res.* **2006**, *7*, 1687–1712.
462. Wang, J.; Shen, X.; Pan, W. On transductive support vector machines. *Contemp. Math.* **2007**, *443*, 7–20.
463. Lee, D.; Lee, J. Equilibrium-based support vector machine for semisupervised classification. *IEEE Trans. Neural Netw.* **2007**, *18*, 578–583. [CrossRef]

464. Adankon, M.M.; Cheriet, M.; Biem, A. Semisupervised least squares support vector machine. *IEEE Trans. Neural Netw.* **2009**, *20*, 1858–1870. [CrossRef]
465. Ma, Y.; Liang, X.; Kwok, J.T.; Li, J.; Zhou, X.; Zhang, H. Fast-solving quasi-optimal LS-S^{3}VM based on an extended candidate set. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 1120–1131. [CrossRef] [PubMed]
466. Zhai, Z.; Huang, H.; Gu, B. Kernel path for semisupervised support vector machine. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 1512–1522. [CrossRef] [PubMed]
467. Du, K.-L. Several misconceptions and misuses of deep neural networks and deep learning. In Proceedings of the 2023 International Congress on Communications, Networking, and Information Systems (CNIS 2023), Guilin, China, 25–27 March 2023; Springer: Berlin/Heidelberg, Germany, 2023; CCIS 1893, pp. 155–171.
468. Alabdulmohsin, I.; Zhang, X.; Gao, X. Support vector machines with indefinite kernels. In *JMLR Workshop and Conference Proceedings: The Asian Conference on Machine Learning*; JMLR, Inc.: Norfolk, MA, USA, 2014; Volume 39, pp. 32–47.
469. Munoz, A.; de Diego, I.M. From indefinite to positive semidefinite matrices. In Proceedings of the Joint IAPR International Workshops, Structural, Syntactic, and Statistical Pattern Recognition, Hong Kong, China, 17–19 August 2006; pp. 764–772.
470. Luss, R.; d'Aspremont, A. Support vector machine classification with indefinite kernels. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2007; Volume 20, pp. 953–960.
471. Haasdonk, B.; Pekalska, E. Indefinite kernel Fisher discriminant. In Proceedings of the 19th International Conference on Pattern Recognition, Tampa, FL, USA, 8–11 December 2008; pp. 1–4.
472. Schleif, F.-M.; Tino, P. Indefinite core vector machine. *Pattern Recogn.* **2017**, *71*, 187–195. [CrossRef]
473. Signoretto, M.; Olivetti, E.; De Lathauwer, L.; Suykens, J.A.K. A kernel-based framework to tensorial data analysis. *Neural Netw.* **2011**, *24*, 861–874. [CrossRef] [PubMed]
474. Signoretto, M.; Olivetti, E.; De Lathauwer, L.; Suykens, J.A.K. Classification of multichannel signals with cumulant-based kernels. *IEEE Trans. Signal Process.* **2012**, *60*, 2304–2314. [CrossRef]
475. Zhao, Q.; Zhou, G.; Adali, T.; Zhang, L.; Cichocki, A. Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. *IEEE Signal Process. Mag.* **2013**, *30*, 137–148. [CrossRef]
476. Luo, L.; Xie, Y.; Zhang, Z.; Li, W.-J. Support matrix machines. In Proceedings of the the 32nd International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; pp. 938–947.
477. Xu, W.; Liu, J.; Lian, H. Distributed Estimation of Support Vector Machines for Matrix Data. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 6643–6653. [CrossRef]
478. Tao, D.; Li, X.; Hu, W.; Maybank, S.; Wu, X. Supervised tensor learning. In Proceedings of the 5th IEEE International Conference on Data Mining, Houston, TX, USA, 27–30 November 2005; pp. 450–457.
479. Lian, H. Learning rate for convex support tensor machines. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 3755–3760. [CrossRef]
480. Guo, W.; Kotsia, I.; Patras, I. Tensor learning for regression. *IEEE Trans. Image Process.* **2012**, *21*, 816–827. [CrossRef]
481. He, L.; Lu, C.-T.; Ding, H.; Wang, S.; Shen, L.; Yu, P.S.; Ragin, A.B. Multi-way multi-level kernel modeling for neuroimaging classification. In Proceedings of the IEEE IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6846–6854.
482. He, L.; Lu, C.-T.; Ma, G.; Wang, S.; Shen, L.; Yu, P.S.; Ragin, A.B. Kernelized support tensor machines. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 1442–1451.
483. He, L.; Kong, X.; Yu, P.S.; Yang, X.; Ragin, A.B.; Hao, Z. DuSK: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. In Proceedings of the 2014 SIAM International Conference on Data Mining (SDM), Philadelphia, PA, USA, 24–26 April 2014; pp. 127–135.
484. Tao, D.; Li, X.; Hu, W.; Maybank, S.; Wu, X. Supervised tensor learning. *Knowl. Inf. Syst.* **2007**, *13*, 1–42. [CrossRef]
485. Chen, C.; Batselier, K.; Yu, W.; Wong, N. Kernelized support tensor train machines. *Pattern Recogn.* **2022**, *122*, 108337. [CrossRef]
486. Kotsia, I.; Patras, I. Support tucker machines. In Proceedings of the CVPR 2011, Colorado Springs, CO, USA, 20–25 June 2011; pp. 633–640.
487. Kour, K.; Benner, P.; Dolgov, S.; Stoll, M. Efficient structure-preserving support tensor train machine. *J. Mach. Learn. Res.* **2023**, *24*, 1–22.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.